


ORIGINAL ARTICLE

Korean TableQA: Structured data question answering based on span prediction style with S³-NET

Cheoneum Park^{1,2}  | Myungji Kim³ | Soyeon Park³ | Seungyoung Lim³ | Jooyoul Lee³ | Changki Lee²

¹AIR Lab, HYUNDAI MOTOR COMPANY, Seoul, South Korea

²Computer Science, Kangwon National University, Chuncheon, South Korea

³AI Research, LG CNS, Seoul, South Korea

Correspondence

Changki Lee, Computer Science, Kangwon National University, Chuncheon, South Korea.

Email: leeck@kangwon.ac.kr

Funding information

This research was supported by LG CNS under the Question Answering project for formatted documents and Korea Electric Power Corporation. (Grant number:R18XA05)

The data in tables are accurate and rich in information, which facilitates the performance of information extraction and question answering (QA) tasks. TableQA, which is based on tables, solves problems by understanding the table structure and searching for answers to questions. In this paper, we introduce both novice and intermediate Korean TableQA tasks that involve deducing the answer to a question from structured tabular data and using it to build a question answering pair. To solve Korean TableQA tasks, we use S³-NET, which has shown a good performance in machine reading comprehension (MRC), and propose a method of converting structured tabular data into a record format suitable for MRC. Our experimental results show that the proposed method outperforms a baseline in both the novice task (exact match (EM) 96.48% and F1 97.06%) and intermediate task (EM 99.30% and F1 99.55%).

KEYWORDS

TableQA, structured data, question answering, question understanding, span prediction, S³-NET

1 | INTRODUCTION

Many studies have recently been conducted on machine reading comprehension question answering (MRQA) [1-4]. Machine reading comprehension (MRC) implies a machine's ability to understand a given context and use it for reasoning. Question answering (QA) is the task of correctly answering a given question. For example, MRQA should be able to understand the context "... Hangul was completed in 1443 and published in 1446 along with a 33-page manual titled Hunmin Jeong-eum, explaining what the letters are as well as the philosophical theories and motives behind them..." and find the correct answer "Hunmin Jeong-eum" to the question "What type of writing system did Sejong the Great use in 1443?" in the context.

MRQA tasks include the Children's Book Test (CBT) dataset [1] created by Facebook, the Stanford Question Answering

Dataset (SQuAD) [2], WikiQA [3], and the Microsoft Machine Reading Comprehension (MSMARCO) dataset [4] created by Microsoft. The CBT [1] is a cloze-style QA task that consists of understanding a given context and question to effectively select the correct answer to questions from among candidate answers. SQuAD [2] is a span prediction task consisting of producing a correct answer given accurate understanding of a passage and question. WikiQA [3] is a sentence detection task that consists of understanding a given paragraph and question to identify the sentence in the paragraph that contains the answer to the question. MSMARCO [4] is a task in which a number of passages and a question are provided as input, and the system generates an answer sequence in response to the question based on its understanding of the passages.

The unstructured data of an MRQA task are composed mainly of plaintext, such as natural language text in

paragraphs taken from news articles or Wikipedia. An example of plaintext is “Sejong the Great was the fourth king of the Joseon dynasty of Korea,” which is quoted from the Wikipedia article about Sejong the Great. However, most documents requiring QA consist of various forms of data, including structured data, alongside plaintext. In recent years, MRQA methods for plaintext have performed comparably to humans [2,5], but QA methods for tables have not, although tabular-structured data facilitate reasoning similar to that enabled by plaintext data. Furthermore, a table can be easily understood when the tabular structure is utilized as features.

Unlike MRQA, which uses only plaintext, TableQA is a task that performs QA based on tabular data having form and structure. To find the correct answer to a given question in a table, we must build a knowledge base by constructing the table as a triple. This approach can be problematic because of the high maintenance costs that result from adding or changing fields. When a database is created for table data, the given question must be written in a structured query language (SQL) query to find the answer. Moreover, finding contents within a large table composed of various columns and cells requires a significantly complex SQL [6].

In this study, we performed a QA task for structured data, such as TableQA, and we present the following solutions to the problems mentioned above. (a) We propose a deep learning model for finding the correct answer to a given question from tabular data. The proposed model predicts the correct answer using the span prediction style of MRQA in an end-to-end method. (b) We built a new dataset for the TableQA task and attempted to solve the task using the proposed model. This dataset consisted of 100 000 instances of fruit- and Olympic Games-themed novice tasks and 100 000 instances of intermediate tasks on the topic of telephone services.

2 | RELATED WORK

2.1 | Table question answering

WikiTableQuestions [7] encodes the relationship and entities of semi-structured Hypertext Markup Language (HTML) tables into a knowledge graph, generates a graph query by parsing the knowledge graph with questions based on the semantic parsing system, and finally, searches for an answer to the given question by ranking the graph query. The Neural Enquirer [8] uses randomly generated Olympic Games-themed table data, creates the embedding for a given question and table as an end-to-end model, and returns the probability of each cell in the table. TabNet [9], which is a model for classifying Web table data types, encodes a table using recurrent neural networks (RNNs) [10] and convolutional neural networks (CNNs) [11]

to extract feature vectors. Similarly to TabNet, TabVec [12] has been studied for crawling Web table data, preprocessing (such as stop-word removal and normalization) of the crawled data, constructing a word vector space, and classifying tables by clustering. Using the Allen Institute for Artificial Intelligence (Ai2)’s Aristo Tablestore dataset, a log-linear model was proposed based on 11 features, including term frequency-inverse document frequency (TF-IDF) and word inclusion ratios [13]. Data were generated by crowdsourcing 15 000 questions for 150 Wikipedia documents, and QA was performed on infobox using a multi-channel CNN [14]. Structured data were constructed in triples, that is, rows, columns, and values [15], and the QA was solved using the end-to-end memory network [16]; however, this system was not deemed usable because of the low test accuracy.

2.2 | Machine reading comprehension for span prediction

Span prediction MRC is a task that consists of finding the answer to a question in a passage with a span, as exhibited by the SQuAD style. To solve this task, r-net [17], BiDAF [18], fusion-net [19], SAN [20], Unet [21], and S³-NET [22] have been studied. Deep learning models for MRC frequently use attention pooling for questions and passages, such as BiDAF, to encode two sequences so that they understand each other, as well as a method for rearranging the encoded passage with the self-attention mechanism used in r-net. For each method encoding the low-, middle-, and high-understanding layers, fusion-net uses all the encoding information when calculating the attention score of the model, SAN employs the ensemble effect to create various output results using an RNN in the output layer, and S³-NET, which encodes passages at word- and sentence-level, calculates an attention score for the output. Many studies have been conducted on QA in the English language using various table datasets, but no appropriate dataset exists in the Korean language, and no studies have been conducted on Korean QA. Thus, the contributions of this study can be summarized as follows:

1. *Korean TableQA dataset*: We constructed a structured QA dataset for TableQA in the Korean language.
2. *End-to-end deep learning model to solve TableQA*: We used an end-to-end deep learning model to solve TableQA. To achieve this, we used our proposed new format for table planning, where table data are defined as a record unit. We used S³-NET, which performs well in Korean MRC, to solve TableQA transformed into records.
3. *Features for TableQA*: If the tabular data in the record unit are changed, the structure of the table collapses. To solve this problem, we propose the extraction of five features for a table and two features for MRC.

3 | STRUCTURED DATA FOR TABLEQA

In this study, we constructed a Korean TableQA dataset to perform MRQA on structured tabular data. The Korean TableQA dataset consists of 100 000 novice tasks on fruit- and Olympic Games–themed topics and 100 000 intermediate tasks on the topic of telecom services.

3.1 | Dataset generation

We present two versions of synthetic datasets with different levels of difficulty for the QA task involving tabular documents. We first describe the data generation process for the novice data and then extend it to the intermediate version with a more complex task. Both datasets consist of table (T), question (Q), and answer (A) triples, where the size of each table T^i is denoted by (n^i, m^i) . The answer A_j^i denotes a cell that answers the question Q_j^i . Here, i is the index for data instances and j is the index for question and answer pairs.

3.2 | Novice task: Syntactic table generation

The novice task targets tables on two subjects: the Olympic Games and fruit. To generate a table, we first set its size. The numbers of rows and columns in the table were randomly set at between 5 and 10. Then, we drew m^i fields and $n^i - 1$ values for each field from the predefined set outlined in Table 1. Cell values were sampled without replacement by default, and sampling with replacement was performed when the number of possible values was less than the number of values that must be drawn. The cell values were created by sampling natural language tools and vocabulary that could be used for Level 1 to Level 3 questions. Ten thousand tables were generated for each subject; an example of a table is provided in Table 2.

3.3 | Natural language question generation

For the novice task, the system is required to solve three levels of questions, as displayed in Table 3. In Table 3, Q denotes question, C denotes condition, and A denotes answer position. For each table, 10 questions were created with templates that were preset with syntactic variation. In total, we generated 60 000, 20 000, and 20 000 questions for Levels 1, 2, and 3, respectively.

3.4 | Intermediate task

The data for the novice tasks were created with rules such that questions were asked following a certain

TABLE 1 Possible numbers of values for each field and its respective type

Olympics		
Field name	Count	Type
Country	204	Cat
City	184	Num
Year	87	Num
Season	2	Num
No. of participants	50	Cat
No. of cheerleaders	800	Num
Advertising revenue	8000	Num
Popular sport	38	Cat
Audience's response rate	101	Num
Stadium area	9000	Num
Ticket selling rate	41	Num
Medal-winning country	204	Cat
Ranking	50	Num
Athletic ranking	50	Num
Asian participation rate	39	Num
Gold medal event	38	Cat
TV station	4	Cat
Viewer ratings	60	Num
Main MC	37	Cat
Singer	29	Cat
Fruits		
Field name	Count	Type
Fruit	16	Cat
Country of origin	204	Cat
City of origin	184	Cat
Production month	12	Num
Production amount	49 000	Num
No. of likes	9900	Num
Shipping season	4	Cat
Price	8000	Num
Size	99	Num
Quality level	8	Num
Preference	99	Num
Brand	5	Cat
Channel	8	Cat
Selling amount	970	Num
Selling rate	300	Num

template and were therefore unnatural and grammatically incorrect. Consequently, we created the intermediate dataset with more natural questions, targeting the tables on telephone services, as shown in Table 4. The process for creating the table was the same as that for

TABLE 2 Example of a table generated in the fruit domain. As shown, letters representing the unit can be stored in numeric columns

Fruit	Country of origin	Price	Size	Preference	Selling rate	Shipping season
Orange	Belgium	1329 won	9	Rank 32	12.9%	Spring
Apple	Korea	4302 won	58	Rank 9	0.7%	Winter
Cherry	US	8520 won	14	Rank 89	20.0%	Winter
Mango	Jamaica	6341 won	57	Rank 3	2.3%	Spring

TABLE 3 Example of the three levels of question templates generated for a novice task. The field names or values corresponding to the parentheses were sampled to generate questions according to the template; columns shown in blue can be extracted only from numeric fields**Lv 1. Retrieve {Col 1} value that {Col 2} has {Value 2}**

(KR) Q: 과일 오렌지의 가격은 얼마입니까?

(gwail olenjiui gagyeg-eun eolmaibnikka?)

(EN) Q: What is the price of the fruit Orange?

C: {Col 1: Price}, {Col 2: Fruit}, {Value: Orange}

A: (1, 3)

Lv2. Retrieve {Col 1} value with its {min/max}

(KR) Q: 가장 높은 판매율은 무엇입니까?

(gajang nop-eun panmaeyul-eun mueos-ibnikka?)

(EN) Q: What is the highest selling rate?

C: {Col 1: Selling rate}, {max}

A: (3, 6)

Lv3. Retrieve {Col 1} value where {Col 2} value is {min/max}

(KR) Q: 선호도가 가장 낮은 가격은 얼마입니까?

(seonhodoga gajang naj-eun gagyeg-eun eolmaibnikka?)

(EN) Q: What is the price when the preference is the lowest?

C: {Col 1: Price}, {Col 2: Preference}, {min}

A: (3, 3)

the novice task; however, to provide greater realism, the first and second columns specified the product name and price, respectively, as shown in Table 5. It was necessary to include the “Name” and “Price” fields in every table to reflect reality, but the remaining fields were randomly selected and placed in various orders. We created the table form by referring to the product description on the actual telephone service Website. The intermediate task table was designed such that each field contains a variety of values for a rich vocabulary and a variety of numbers. The number of possible values for each column is accurately indicated in the list in Table 4. All the cell values in the table were randomly selected within their respective ranges.

To generate questions, we imposed lexical variations so that the field name was not directly used in the question and provided more diverse syntactic variations than

TABLE 4 Data fields for telephone services randomly utilized for the intermediate task. Field names were extracted from actual Korean carrier Websites

Field name	Count	Type
Name	600	Cat
Price	91	Num
Services	21	Cat
Terms of agreement	4	Cat
No. of Messages	101	Num
Model	40	Cat
Calls available	76	Num
Freebies	21	Cat
Data plan	246	Num
Available groups	6	Cat
Internet discount	101	Num
Payment method	6	Cat
Mobile discount	101	Num
Combined services	4	Cat
Combined discount	19	Num
Carrier	3	Cat
Tax	10	Num
Membership	5	Cat
Data speed limit	3	Cat

in the novice task. Furthermore, we added two more complex levels of queries, as outlined in Table 6. For the intermediate task, we generated 10 000 tables and 20 000 questions for each level. Finally, various vocabulary words and phrases were used by capitalizing on the characteristics of the Korean language. For example, we considered colorful sentences, such as honorific and casual speech, and varied vocabulary for synonyms, such as 월 얼마 내야 해? (How much should I pay in a month?), 요금은 얼마인가요? (How much is a fee?).” Additionally, there were some questions that did not directly mention field names, such as “얼마야? (How much?)” and “누가 광고해? (Who advertises?),” and therefore, inference as to which column the subject of the question corresponds in the given table was necessary. In Table 6, Q denotes question, C denotes condition, and A denotes answer position in the table.

TABLE 5 Example of a table generated for the telephone service domain. As shown, letters representing the unit can be stored in numeric columns

Name	Price	Data plan	Carrier	Tax	Model	No. of Messages
Rainbow kids-56	53 000 won	0.8 GB	A-telecom	2650 won	Inna You	980
Compact-45	33 000 won	13.7 GB	A-telecom	3630 won	Chaewon Moon	520
Advanced-56	63 000 won	17.8 GB	B-telecom	3780 won	Seongjae Yook	580
Rainbow army-C	49 000 won	20.3 GB	A-telecom	7350 won	Astro	100

4 | KOREAN TABLEQA USING S³-NET

The Korean TableQA dataset consists of questions (Q), tables (T), and answers (Y). Each question contains m words, that is, $Q = \{q_1, q_2, \dots, q_m\}$, and each table consists of r rows and c columns, that is, $T = \{t_1, 1, t_1, 2, \dots, t_{(1,c)}, \dots, t_{(r,c)}\}$. The correct answer for each question can be identified by row and column, such as $Y = \{P_{\text{row}}, P_{\text{column}}\}$. To use the S³-NET model to solve the Korean TableQA, we performed training and inference by modifying the tabular data into an unstructured format. Table T was changed into a record unit to be formed into sentences, and the records created were concatenated to form one passage (P). The created passages comprised n words, that is, $P = \{p_1, p_2, \dots, p_n\}$, and the positions of the correct answers, which are indicated by rows and columns, were also changed corresponding to the index of the modified passages. The new correct answer positions were then defined as the start and end boundaries of the correct answers, as in the span prediction style of SQuAD, defined as $Y = \{P_{\text{start}}, P_{\text{end}}\}$. The method for changing tabular data into record units is as follows:

For a given table, as exemplified in Figure 1, the two-dimensional tabular data are composed of rows and columns. Each cell in the row is concatenated with the head in the same column to form a “head cell,” which is transformed into a record unit as follows:

Changing tabular data into a record unit:

KR: [“최다 메달 국가 터키 계절 동계 관객 호응도 0.1”, “최다 메달 국가 이탈리아 계절 하계 관객 호응도 0.9”].

EN: [“Most medal country Turkey season winter audience response 0.1”, “Most medal country Italy season summer audience response 0.9”].

4.1 | Features

As tabular data, Korean TableQA extracts features according to TableQA. The record example derived from the tabular data in Figure 1, that is, “Most medal country Turkey season winter audience response 0.1,” is also an example of feature extraction.

- **Row position:** A word in a record is a feature that can indicate the position of the row in the table. For example, “row-1 row-1 row-2 row-1 row-2 row-1 row-1 row-2.”

- **Column position:** A word in a record is a feature that can indicate the position of the column in the table. For example, “col-1 col-1 col-1 col-1 col-2 col-2 col-3 col-3 col-3.”
- **Boundary feature:** Records use head-B, head-I, cell-B, and cell-I tags to distinguish each head and cell boundary. For example, “head-B head-I head-I cell-B head-B cell-B head-B head-I cell-B.”
- **Head embedding:** Head embedding is based on the word dictionary. It consists of the morphemes of the head shown in the table, according to the training performed.
- **Min/max feature:** The min/max feature distinguishes the minimum and maximum numeric values for the same attribute in a given table.

Additionally, we used features such as a character CNN, exact match (EM), term frequency, and aligned question embedding [17]. They are detailed as follows:

- **Exact match:** EM is a binary feature that outputs 1 if each word in the passage is included in the question and 0 otherwise.
- **Term frequency:** Term frequency (tf) is a feature that calculates the frequency of each word for a passage or question. tf is normalized to the length of the inputted passage in question.
- **Aligned question embedding:** The question sentence is encoded and then an alignment vector is generated.

The weighted sum with the encoding-hidden state of the question sentence is then calculated to create a question vector \mathbf{q} . The question vector \mathbf{q} is given by (1) and is used to output the correct answer in the output layer.

$$\mathbf{q} = \sum_j b_j \mathbf{u}_j^Q, \quad (1)$$

$$b_j = \frac{\exp(\mathbf{w} \cdot \mathbf{u}_j^Q)}{\sum_{j'} \exp(\mathbf{w} \cdot \mathbf{u}_{j'}^Q)}. \quad (2)$$

4.2 | Model: S³-NET

S³-NET is the MRC model of a hierarchical module using self-attention matching based on a simple recurrent unit

TABLE 6 Values shown in orange can be extracted only from fields sampled with replacement and in blue only from numeric fields. As shown, the field name was queried with different words and syntax variants

Lv 1. Retrieve {Col 1} value that {Col 2} has {Value 2 | Value}

(KR) Q: 월 요금 63,000원인 상품이 뭔가요?
 (wol yogeum 63,000won-in sangpum-i mwongayo?)
 (EN) Q: What is the product that costs 63,000 won per month?
 C: {Col 1: Data plan}, {Col 2: Price}, {Value: 63,000}
 A: (3, 1)

Lv 2. Retrieve {Col 1} value is {min/max}

(KR) Q: 가장 센 요금제는 얼마 내야 되니?
 (gajang sen yogeumjeneun eolma naeya doeni?)
 (EN) Q: How much does the most expensive plan cost?
 C: {Col 1: Price}, {max}
 A: (4, 2)

Lv 3. Retrieve {Col 1} value where {Col 2} value is {min/max}

(KR) Q: 데이터 가장 적게 줄 때 광고는 누가해요?
 (deiteo gajang jeogge jul ttae gwang-goneun nugahaeyo?)
 (EN) Q: Who does the ads when you give the least data?
 C: {Col 1: Model}, {Col 2: Data plan}, {min}
 A: (1, 6)

Lv 4. Retrieve {min/max} of {Col 1} where {Col 2} value meets {Value 2 | Range}

(KR) Q: 5,000원 이하의 부가 서비스 중에서 최저 요금은 얼마입니까?
 (5,000 won ihai buga seobiseu jung-eseo choejeo yogeum-eun eolmaibnikka?)
 (EN) Q: What is the lowest rate among services with a surtax of 5,000 won or less?
 C: {min}, {Col 1: Tax}, {Col 2: Price}, {Range: <= 5,000}
 A: (2, 2)

Lv 5. Retrieve {Col 1} value where {Col 2} is {min/max} and {Col 3} value meets {Value 3 | Range}

(KR) Q: 부가세가 가장 높을 때 80 개의 메시지를 보낼 수 있는 이동통신사는 무엇입니까?
 (bugasae-ga gajang nop-eul ddae 80 gaeui mesijileul bonael su-issneun idong tongsinsaneun mueos-ibnikka?)
 (EN) Q: Which carrier can send up to 80 messages where the tax is the highest?
 C: {Col 1: Carrier}, {Col 2: Tax}, {max}, {Col 3: Messages}
 {Range: <= 80}
 A: (4, 7)

(SRU) [23]. In this study, we used S³-NET, as illustrated in Figure 2, to solve the Korean TableQA of structured tabular data.

S³-NET receives the word and feature embedding as input and concatenates them together to form \bar{P} and \bar{Q} , respectively. Subsequently, passage and question encoding are

performed using bidirectional SRU (BiSRU) in each hidden layer at the word level. For the record level, record embedding is created from the inputted record, and the record encoding is performed in the hidden layer. The record-hidden state then performs and models the attention mechanism between the record and the question encoding to create a question-aware record representation that knows the question information. Passage encoding is performed and the attention mechanism is modeled with the question-aware and record-hidden state to create a record-aware passage representation that knows the information generated previously, such as question and record. Finally, modeling is performed again through the self-matching network, and the start and end positions of the correct answer span are outputted using the pointer network [24].

Both passage encoding (U^P) and question encoding (U^Q) are generated using BiSRU. Here, P denotes passage and Q denotes question. Record embedding creates a hidden state P^R using a CNN for each record based on the word embedding of the input word. The record-hidden state U^R is created by encoding based on the generated record embedding.

In the question-record matching layer, we use a gated attention-based RNN to create a record-hidden state with question information.

$$V^R = BiSRU(f_{gate}([U^R; C^R])), \quad (3)$$

$$C^R = U^Q f_{attn}(U^Q, U^R). \quad (4)$$

The context vector C^R including the question information is calculated by the attention weights for U^Q and U^R and the weighted sum U^Q . The attention weight function $f_{attn}(\cdot)$ uses a bilinear sequence scoring method. The additional gate $f_{gate}(\cdot)$ [17] is a function that creates a gated weight by applying a sigmoid to the input and then produces a result vector by an element-wise product between the input and the gated weight. Thus, the result vector generated through $f_{gate}(\cdot)$ is modeled by applying BiSRU to create V^R . In the record-passage matching layer, the context vector is created in the same manner as above, and the modeling of the context vector is performed by applying BiSRU.

$$V^P = BiSRU(f_{gate}([U^P; C^P])), \quad (5)$$

$$C^P = V^R f_{attn}(V^R, U^P). \quad (6)$$

S³-NET applies the self-attention mechanism based on the previously generated passage encoding-hidden state V^P in the self-matching layer and finally performs modeling again.

$$H^P = BiSRU(f_{gate}([H^P; C^P])), \quad (7)$$

$$C^P = V^P f_{attn}(V^P, V^P). \quad (8)$$

	Column 1	Column 2	Column 3
Row 1 / Head	최대 메달 국가 (Country with most medals)	계절 (Season)	관객 호응도 (Audience response)
Row 2	터키 (Turkey)	동계 (Winter)	0.1 (0.1)
Row 3	이탈리아 (Italy)	하계 (Summer)	0.9 (0.9)

FIGURE 1 Visualization of tabular data

Based on the hierarchical pointer network, S³-NET outputs the start and end indexes, P_{start} and P_{end} , of the correct answer corresponding to the question within the input passage.

$$P_{start}(k) \propto \exp(v_{t,k}^R W_s^R q) \exp(h_k^P W_s^P q), \quad (9)$$

$$P_{end}(k) \propto \exp(v_{t,k}^R W_e^R q) \exp(h_k^P W_e^P q). \quad (10)$$

We used the bilinear sequence from the attention score method in the above equation; the attention mechanism generates the final score by calculating the records and passage scores and multiplying them together. The hidden state of the record modeling is $v_{t,k}^R$, the question vector is q , and the hidden state of the passage modeling is h_k^P . A word index is denoted by k , and a record index including the word k is denoted by t .

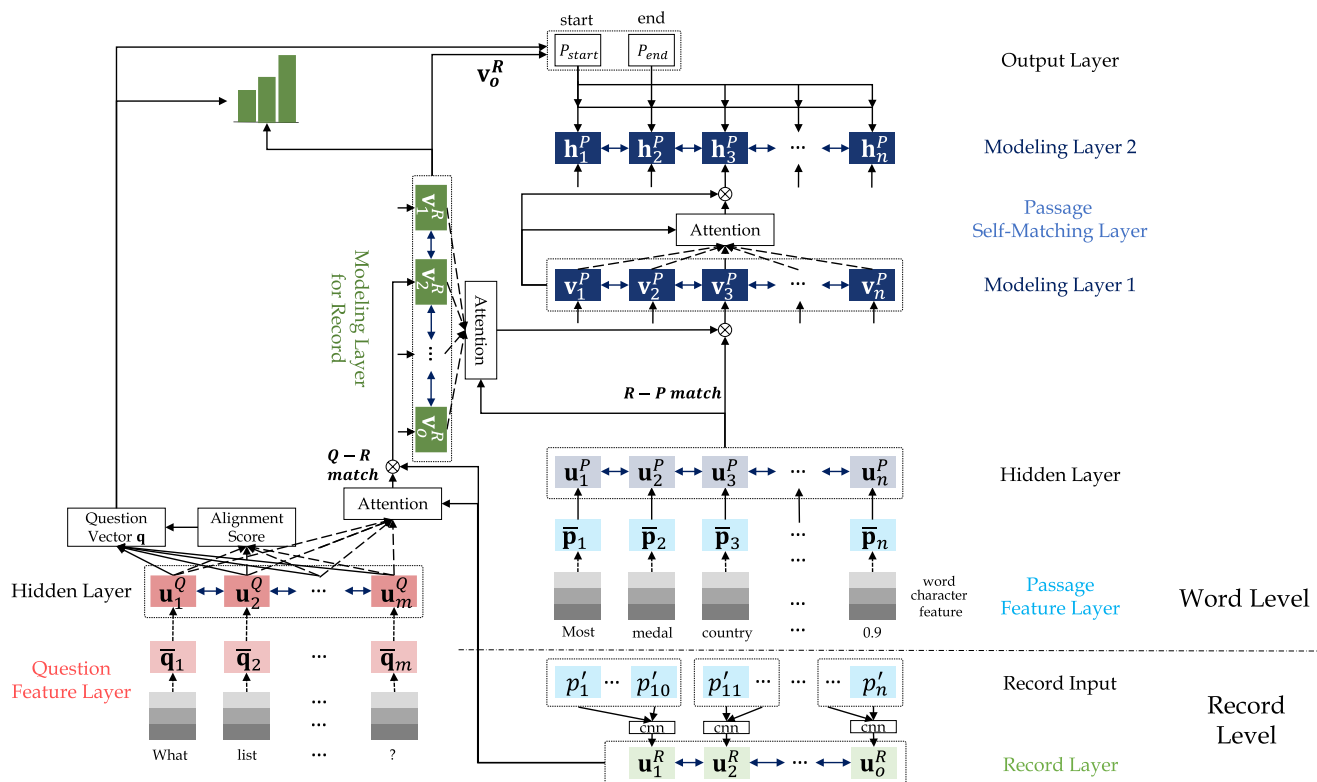


FIGURE 2 S³-NET structure for record format of TableQA

5 | EXPERIMENTS

Our experiments were performed on a computer with an Intel Core i7-4790 3.60-GHz CPU, 32 GB of RAM, and an NVIDIA TITAN X (Pascal) graphics card. The S³-NET test code was developed using PyTorch.

5.1 | Dataset

The dataset used in the experiment was the Korean TableQA of LG CNS, which was divided into novice and intermediate tasks. Each task consisted of 100 000 total data. The novice task consisted of 60 000 Level 1 (Lv 1), 20 000 Level 2 (Lv 2), and 20 000 Level 3 (Lv 3) tasks, where the level indicates the difficulty of the question. The intermediate task was divided into five levels (Lv 1 to Lv 5), where each level consisted of 20 000 data. The Korean TableQA of LG CNS divides the entire dataset according to an 8:2 ratio into training and test sets per question difficulty for training and evaluation, respectively. Table 7 outlines the statistics of the data per task.

5.2 | Settings

The experimental environment for Korean TableQA using S³-NET is described below. The activation function for both the RNN hidden layer and attention layer was tanh, and all

RNN layers used BiSRU. The dropout of the embedding layer was fixed at 0.5, and the dropout of all hidden layers was set at 0.2. The numbers of dimensions of the character embedding, word embedding, and hidden layers were 50, 100, and 80, respectively. We performed optimization for the above hyper-parameters for each task of the TableQA. The word embedding used in this study was trained on 354 499 news articles and 299 768 social community data collected by LG CNS using continuous bag of words [25]. The character embedding employed a CNN with a filter size of (2, 3, 4, 5, 6), where the number of dimensions of each filter was set to 30. Record embedding also used a CNN with a filter size of (3, 4, 5), where the number of dimensions of each filter was set to 60. We used Adam [26] for training and set the learning rate to 0.001. The placement size of the mini-batch was set to 150, and the optimal model was obtained by performing the evaluation with each development set for each epoch. We used the EM and F1 scores as the performance measures [2].

5.3 | Novice task

The novice task targeted tables on two subjects: the Olympic Games and fruit. We first identified the best features of the novice task through feature ablation. Table 8 presents the F1 score and feature ablation results for the novice task in Korean TableQA. The F1 score of S³-NET using all the proposed features was 97.06%. In terms of feature ablation, removal of head and question embedding reduced the performance by 0.08% to 96.98%. Removal of the boundary feature further decreased the performance by 0.10% to 96.96%. The min/max feature is significant because its removal led to a 1.01% performance decrease. Removal of the column position feature dramatically increased the performance degradation to 10.11%, leading to an F1 score of 86.95%. Similarly, EM & term frequency and row position removal initiated 10.81% and 12.66% decreases in the F1 score to 86.25% and 84.40%, respectively, demonstrating the significance of these features in terms of solving TableQA. Finally, the removal of character

TABLE 7 Numbers of tables and questions generated for each data item

	Novice	Intermediate
No. of Tables	10 000	10 000
No. of Questions	100 000	100 000
No. of Lv 1	60 000	20 000
No. of Lv 2	20 000	20 000
No. of Lv 3	20 000	20 000
No. of Lv 4	N/A	20 000
No. of Lv 5	N/A	20 000

embedding (char CNN) generated the greatest difference, 15.40%, which is possibly due to the inclusion of the numerical value in the table data causing a large ratio of unknown words.

Table 9 lists the performances according to question difficulty for S³-NET on TableQA. At Level 1, the performance scores of EM and F1 were 97.65% and 98.06%, respectively. For Level 2, the EM and F1 scores were 94.95% and 95.53%, respectively, and for Level 3 the scores were 94.53% and 95.58%, respectively. Thus, the results indicate that an increase in the question complexity level decreases the performance scores, supporting the logical assumption that question complexity is proportionally related to the difficulty the machine encounters in understanding the question.

We performed an additional experiment on the passage length, as illustrated in Figure 3, to examine the change in performance in the novice task for various passage lengths of S³-NET. Table 10 provides statistics for the ranges of passage lengths when the table was transformed into record format for the novice task test set. Given the length of the longest and shortest passages were 408 and 303 words, respectively, we evaluated the performance in terms of passage length in 20-word increments, beginning with 300 words. The performance for short passages, specifically documents of less than 340 words, was the best, and the performance decreased with the increasing passage length. The performance for passage lengths ranging from 0 to 320 words was low as a result of insufficient training data.

5.4 | Intermediate task

The intermediate task contained more natural questions than the novice task and targeted tables related to telephone services. Our experiments on the intermediate task subsequently provided optimizations for the number of hidden layer dimensions, the number of stack layers for question, passage encoder, and modeling layers, the dropout for RNN,

TABLE 8 Feature ablation of the novice task (%)

Model	F1	Δ
S ³ -NET for TableQA	97.06	N/A
- head embedding	96.98	-0.08
- question embedding	96.98	-0.08
- boundary feature	96.96	-0.10
- min/max feature	96.05	-1.01
- column position	86.95	-10.11
- EM & term frequency	86.25	-10.81
- row position	84.40	-12.66
- char CNN	81.66	-15.40

TABLE 9 Performance according to question level for the novice task (%)

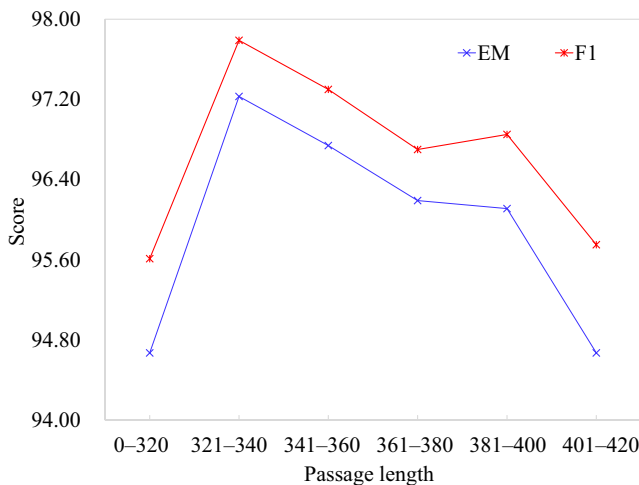
Question level	EM	F1
All Levels	96.48	97.06
Lv 1	97.65	98.06
Lv 2	94.95	95.53
Lv 3	94.53	95.58

and RNN type. We also performed feature ablation. The optimization of the hidden layer dimension number is characterized in Table 11. As a result, EM was 99.24% and F1 was 99.52%, which are the highest performances achieved.

The optimization of the number of stack layers for each question, passage encoder, and modeling layer depends on the optimization of the number of hidden layer dimensions. As evidenced in Table 11, 80 dimensions yielded the highest performance. The optimization of the number of stack layers is described in Table 12. When the numbers of question encoders, passage encoders, and modeling layers were 3, 3, and 2, respectively, the performance peaked with an EM of 99.30% and F1 of 99.55%.

The dropout optimization of the hidden layer of S^3 -NET is characterized in Table 13, given 80 hidden layer dimensions, 3-stack question encoders, 3-stack passage encoders, and 2-stack modeling layers. The experimental results showed the best performance for a dropout of 0.2.

Table 14 gives the performance of each RNN type of S^3 -NET. The S^3 -NET employed in this study is based on SRU; when SRU was used, the EM and F1 scores peaked at 99.30% and 99.55%, respectively. The training time for SRU is the fastest at 696 seconds, approximately 1.26 and 1.62 times faster than that for the gated recurrent unit (GRU) [27] and long short-term memory (LSTM) [28] models, respectively.

**FIGURE 3** Performance on the novice task for varying passage lengths

Similarly, the test time for SRU was fastest at 127 seconds, approximately 1.54 and 1.49 times faster than for GRU and LSTM, respectively.

In Table 15, we present the results of feature ablation on the previously optimized S^3 -NET. With removal of both question and head embedding, decreases of 0.01 and 0.03 were observed, respectively. The removal of row and column position features returned 0.15% and 0.31% decreases, respectively. Furthermore, the min/max feature was 0.44% lower than that of S^3 -NET. Elimination of the EM & term frequency, boundary feature, and hierarchical module returned decreases in performance of 0.78%, 0.98%, and 1.01%, resulting in scores of 98.77%, 98.57%, and 98.54%, respectively. The latter three features were of greater significance to the task than the former features. Elimination of all the related table features, such as head embedding, row and column position, and min/max feature, generated a 1.18% reduction in performance. Thus, we suggest that the features proposed in this paper significantly assist in solving the TableQA. Finally, when character embedding was removed, a dramatic decrease in performance of 9.75% was seen.

Table 16 shows the performance according to the question difficulty of the intermediate task when TableQA was performed with S^3 -NET. Levels 1–3 appear to solve the problem with reduced difficulty, each with performance scores over 99.80%. Because Levels 4 and 5 are more complicated, their performances were reduced, yielding an EM and F1 of 98.53% and 99.28% for Levels 4 and 98.70% and 98.98% for Level 5, respectively.

We conducted an experiment using each passage length, as illustrated in Figure 4, to examine the change in S^3 -NET's performance for various passage lengths in the intermediate task. Table 17 provides data statistics for ranges of passage lengths when the intermediate task dataset was changed from table to record format. The training set's longest and shortest passage lengths were 408 and 283 words, respectively, and the test set's longest and shortest passages were 397 and 273 words, respectively. For passages of word length 300 or less, the EM and F1 scores peaked at 99.57% and 99.69%, respectively. Passage lengths ranging from 341 to 360 words demonstrated the lowest EM, 99.18%, while passage lengths ranging from 361 to 380 words returned the lowest F1, 99.45%.

5.5 | Model analysis

We performed a model analysis, as outlined in Table 18, to confirm that the S^3 -NET model applied in this study conforms to TableQA. We tested both the novice and intermediate tasks. The models used for performance comparison were DrQA [3], BiDAF, and S^2 -NET [29]. The hyperparameters of the models

TABLE 10 Data statistics for passage length ranges in the novice task

Length range	No. of training set	No. of test set
0–320	N/A	300
321–340	900	3750
341–360	10 800	5550
361–80	31 700	7400
381–400	33 000	2700
401–420	3600	300

TABLE 11 Optimization of the number of hidden layer dimensions in S^3 -NET (%)

Number of dimensions	EM	F1
80	99.24	99.52
100	98.65	99.16
150	96.92	98.08
200	98.89	99.30

Bold values mean the highest performance, one of the others.

TABLE 12 Optimization of the number of stack layers for each module (%)

Question encoder	Passage encoder	Modeling layer	EM	F1
2	2	1	99.24	99.52
3	3	1	99.19	99.50
3	3	2	99.30	99.55
4	4	2	98.73	99.19
5	5	2	99.17	99.47

Bold values mean the highest performance, one of the others.

TABLE 13 Optimization of dropout in the hidden layer of S^3 -NET (%)

Dropout rate of hidden layer	EM	F1
0.1	96.86	97.42
0.2	99.30	99.55
0.3	97.44	98.36
0.4	95.76	97.35
0.5	90.04	93.59

Bold values mean the highest performance, one of the others.

were all allocated their optimized values as determined earlier in the study. Both DrQA and BiDAF trained the model using LSTM. In both, the encoder and modeling layer, one stack layer was designated for the BiDAF. The baseline model of the experiment was DrQA. For the novice task, the DrQA baseline returned an EM of 79.08% and F1 of 83.91%. These results are

TABLE 14 Performance for RNN types of S^3 -NET

Question encoder	EM (%)	F1 (%)	Training time (s)	Test time (s)
SRU	99.30	99.55	696	127
GRU	98.20	98.88	880	195
LSTM	97.03	98.07	1128	190

Bold values mean the highest performance, one of the others.

TABLE 15 Feature ablation of S^3 -NET on the intermediate task (%)

Model	F1	Δ
S^3 -NET for TableQA	99.55	N/A
– question embedding	99.54	–0.01
– head embedding	99.52	–0.03
– row position	99.40	–0.15
– column position	99.24	–0.31
– min/max feature	99.11	–0.44
– EM & term frequency	98.77	–0.78
– boundary feature	98.57	–0.98
– hierarchical module	98.54	–1.01
– all table features	98.37	–1.18
– char CNN	89.80	–9.75

TABLE 16 Performance according to the question level of the intermediate task (%)

Question level	EM	F1
All Levels	99.30	99.55
Level 1	99.78	99.83
Level 2	99.80	99.91
Level 3	99.80	99.84
Level 4	98.53	99.28
Level 5	98.70	98.98

explained by the DrQA model's use of only the static alignment score, such as aligned question embedding, instead of the attention mechanism between the question and passage. BiDAF employing a bi-attention flow function exhibited an EM score of 80.58% and F1 score of 85.19%, which are higher values than those achieved by DrQA. BiDAF's higher performance appears to be due to the trainable attention function of BiDAF that reflects the relationship between the given passage and the question. As for S^2 -NET, an EM of 94.16% and F1 of 95.89% were achieved. S^2 -NET's higher performance is considered to be a result of the model's application of the attention mechanism for the passage and question, as in match-LSTM [30], and the calculation of attention weights using the self-attention mechanism. Finally, the S^3 -NET model applied in this study performed

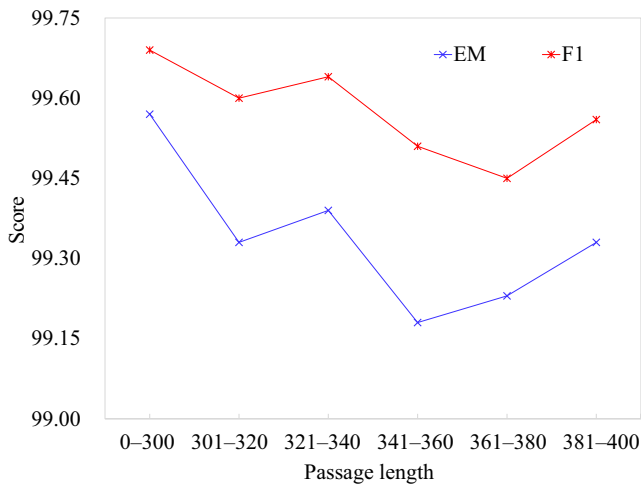


FIGURE 4 Performance on the intermediate task for varying passage lengths

best on the novice task, achieving an EM of 96.48% and F1 of 97.06%, which constitute increases of 17.40% and 13.20%, respectively, as compared to the baseline DrQA. Based on the hierarchical structure, S³-NET's performance is sufficient, because the model adds more weight to the record with the correct answer than the other models. On the intermediate task, the DrQA baseline achieved an EM of 81.52% and F1 of 88.01%. Because aligned question embedding was employed, a static alignment score was used as a feature, and DrQA performed less well than the other models on this task. BiDAF, which computes the attention weight of the context and question with the bi-attention flow function, returned an EM of 96.81% and F1 of 97.91%, exhibiting a significantly high performance on the intermediate task. Thus, the attention mechanism evidently helps solve the task. The S²-NET model achieved the second highest EM, 97.72%, and F1, 98.54%, as well as the top performance with an EM of 99.30% and F1 of 99.55%. As described above, the attention and self-attention mechanisms of question-record and record-passage in S³-NET can understand the relationship between each component based on passage, record, and question information. Further, because of the hierarchical model's ability to find a record close to the correct answer and weight

TABLE 17 Data statistics for passage length ranges in the intermediate task

Length range	No. of training set	No. of test set
0-300	650	1400
301-320	3600	3450
321-340	11 250	3600
341-360	18 150	5250
361-380	34 050	4800
381-400	9600	1500
401-420	2700	N/A

TABLE 18 Model analysis for each task (%)

Model	EM	F1
Novice task		
DrQA [3] (baseline)	79.08	83.91
BiDAF [18] (our implementation)	80.58	85.19
S ² -NET [30] (our implementation)	94.16	95.89
S ³ -NET (our implementation)	96.48	97.06
Intermediate task		
DrQA [3] (baseline)	81.52	88.01
BiDAF [18] (our implementation)	96.81	97.91
S ² -NET [30] (our implementation)	97.72	98.54
S ³ -NET (our implementation)	99.30	99.55

the attention score, S²-NET is considered the best performer for all tasks of TableQA.

6 | CONCLUSION

In this paper, we defined Korean TableQA, which stores structured data in tabular form, and introduced novice and intermediate tasks. To solve the TableQA task as an end-to-end model, we applied S³-NET and proposed five features suitable for TableQA. We transformed all types of tables in TableQA to record format, performed preprocessing for all records as one passage, and extracted the features from the table to solve TableQA using S³-NET. Our experimental results indicate that the proposed method significantly improves performance on the novice task, where an EM of 96.48% and F1 of 97.06% were achieved, as well as on the intermediate task, where an EM of 99.30% and F1 of 99.55% were achieved. We confirmed the significance of the proposed features through feature ablation for each TableQA task. In future work, we will improve the performance by fine-tuning the pre-trained network BERT [31]. We will also define the unstructured data as a single document, consisting of plaintext and structured data composed of tables, as in natural questions [32], and devise a solution for finding the answer to a question in a document.

ORCID

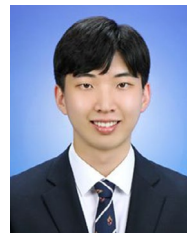
Cheoneum Park  <https://orcid.org/0000-0001-5386-0483>

REFERENCES

1. F. Hill et al., *The goldilocks principle: reading children's books with explicit memory representations*, arXiv preprint arXiv:1511.02301, 2015, pp. 1–13.
2. P. Rajpurkar et al., *Squad: 100,000+ questions for machine comprehension of text*, arXiv preprint arXiv:1606.05250, 2016, pp. 1–10.

3. D. Chen et al., *Reading wikipedia to answer open-domain questions*, arXiv preprint arXiv:1704.00051, 2017, pp. 1–10.
4. P. Bajaj et al., *Ms marco: A human-generated machine reading comprehension dataset*, arXiv preprint arXiv:1611.09268, 2016, pp. 1–11.
5. P. Rajpurkar, R. Jia, and P. Liang, *Know what you don't know: Unanswerable questions for squad*, arXiv preprint arXiv:1806.03822, 2018, pp. 1–9.
6. V. Zhong, C. Xiong, and R. Socher, *Seq2sql: Generating structured queries from natural language using reinforcement learning*, arXiv preprint arXiv:1709.00103, 2017, pp. 1–12.
7. P. Pasupat and P. Liang, *Compositional semantic parsing on semi-structured tables*, arXiv preprint arXiv:1508.00305, 2015, pp. 1–11.
8. Z. Lu, H. Li, and B. Kao, *Neural enquirer: learning to query tables in natural language*, IEEE Data Eng. Bull. **39** (2016), no. 3, 63–73.
9. K. Nishida et al., *Understanding the semantic structures of tables with a hybrid deep neural network architecture*, in Proc. Thirty-First AAAI Conf. Artif. Intell. (San Francisco, CA, USA), Feb. 2017, pp. 168–174.
10. A. Graves, A. Mohamed, and G. Hinton, *Speech recognition with deep recurrent neural networks*, in Proc. IEEE Int. Conf. Acoustics, Speech Signal Process. (Vancouver, Canada), May, 2013, pp. 6645–6649.
11. Y. Kim, *Convolutional neural networks for sentence classification*, arXiv preprint arXiv:1408.5882, 2014, pp. 1–6.
12. M. Ghasemi-Gol and P. Szekely, *Tabvec: Table vectors for classification of web tables*, arXiv preprint arXiv:1802.06290, 2018, pp. 1–9.
13. S.K. Jauhar, P. Turney, and E. Hovy, *Tables as semi-structured knowledge for question answering*, in Proc. Annu. Meeting Association Comput. Linguistics (Berlin, Germany), Aug. 2016, pp. 474–483.
14. A. Morales et al., *Learning to answer questions from wikipedia infoboxes*, in Proc. Conf. Empirical Methods Natural Language Process. (Austin, TX, USA), Nov. 2016, pp. 1930–1935.
15. S. Vakulenko and V. Savenkov, *Tableqa: Question answering on tabular data*, arXiv preprint arXiv:1705.06504, 2017, pp. 1–5.
16. S. Sukhbaatar, J. Weston, and R. Fergus, *End-to-end memory networks*, in Proc. Adv. Neural Inf. Process. Syst. (Montreal, Canada), Dec. 2015, pp. 2440–2448.
17. W. Wang et al., *Gated self-matching networks for reading comprehension and question answering*, in Proc. Annu. Meeting Association Comput. Linguistics (Vancouver, Canada), July 2017, pp. 189–198.
18. M. Seo et al., *Bidirectional attention flow for machine comprehension*, arXiv preprint arXiv:1611.01603, 2016, pp. 1–13.
19. H.-Y. Huang et al., *Fusionnet: Fusing via fully-aware attention with application to machine comprehension*, arXiv preprint arXiv:1711.07341, 2017, pp. 1–20.
20. X. Liu et al., *Stochastic answer networks for machine reading comprehension*, arXiv preprint arXiv:1712.03556, 2017, pp. 1–11.
21. F. Sun et al., *U-net: Machine reading comprehension with unanswerable questions*, arXiv preprint arXiv:1810.06638, 2018, pp. 1–9.
22. C. Park et al., *S³-net: Korean machine reading comprehension using sru-based sentence and self matching networks*, Proceeding of KSC (2017), 649–651.
23. T. Lei, Y. Zhang, and Y. Artzi, *Training rnns as fast as cnns*, arXiv preprint arXiv:1709.02755, 2017.
24. O. Vinyals, M. Fortunato, and N. Jaitly, *Pointer networks*, in Proc. Adv. Neural Inf. Process. Syst. (Montreal, Canada), Dec. 2015, pp. 2692–2700.
25. T. Mikolov et al., *Efficient estimation of word representations in vector space*, arXiv preprint arXiv:1301.3781, 2013, pp. 1–12.
26. D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, arXiv preprint arXiv:1412.6980, 2014, pp. 1–5.
27. J. Chung et al., *Empirical evaluation of gated recurrent neural networks on sequence modeling*, arXiv preprint arXiv:1412.3555, 2014, pp. 1–9.
28. S. Hochreiter and J. Schmidhuber, *Long short-term memory*, Neural computation **9** (1997), no. 8, 1735–1780.
29. C. Park et al., *S²-net: Machine reading comprehension with sru-based self-matching networks*, ETRI J. **41** (2019), no. 3, 371–382.
30. S. Wang and J. Jiang, *Machine comprehension using match- lstm and answer pointer*, arXiv preprint arXiv:1608.07905, 2016, pp. 1–11.
31. J. Devlin et al., *Bert: Pre-training of deep bidirectional transformers for language understanding*, arXiv preprint arXiv:1810.04805, 2018, pp. 1–16.
32. T. Kwiatkowski et al., *Natural questions: a benchmark for question answering research*. Trans. Association Comput. Linguistics. **7** (2019), 453–466.

AUTHOR BIOGRAPHIES



Cheoneum Park received his BS, MS, and PhD degrees in Computer Science from Kangwon National University, Chuncheon, Rep. of Korea, in 2014, 2016, and 2020, respectively. He is now a researcher at Hyundai Motor Company, Seoul, Rep. of Korea. His research interests include natural language processing, natural language understanding, question answering, and deep learning.



Myungji Kim received her BS degree in Statistics from Korea University, Seoul, Rep. of Korea, in 2014. She has been a researcher at LG CNS, Seoul, Rep. of Korea since 2014. Her research interests include natural language understanding, question answering, information retrieval, and deep learning.



Soyeon Park received the BS and MS degrees in Mathematical Science from Seoul National University, Seoul, Rep. of Korea, in 2013 and 2015, respectively. She was a big data engineer from 2015 to 2018 at LG CNS, Seoul, Rep. of Korea and has been an AI researcher since 2018. Her research interests include natural language processing, natural language understanding, question answering, and deep learning.



Seungyoung Lim received the BS and MS degrees in Economics and Statistics, respectively, from Ewha Woman's University, Seoul, Rep. of Korea, in 2016, and 2018, respectively. She has been an AI researcher at LG CNS, Seoul, Rep. of Korea since 2018.

Her research interests include natural language processing and understanding, question answering, and deep learning.



Jooyoul Lee received the BS and MS degrees in Physics from Yonsei University, Seoul, Rep. of Korea, in 1996 and 1998, respectively. He was a software engineer from 2006 to 2015, an AI researcher from 2016 to 2018, and has been leading the AI research

department at LG CNS, Seoul, Rep. of Korea since 2019. His research interests include question answering and neural architecture search.



Changki Lee received his BS degree in Computer Science from the Korea Advanced Institute of Science and Technology, Daejeon, Rep. of Korea, in 1999. He received his MS degree and PhD in Computer Engineering from POSTECH, Pohang, Rep. of Korea, in

2001 and 2004, respectively. From 2004 to 2012, he was a researcher with the Electronics and Technology Research Institute, Daejeon, Rep. of Korea. Since 2012, he has been a professor of Computer Science at Kangwon National University, Chuncheon, Rep. of Korea. His research interests include natural language processing, machine learning, and deep learning.