# A Neural Question Answering Model Based on Semi-Structured Tables

**Hao Wang**[1], **Xiaodong Zhang**[1], **Shuming Ma**[1], **Xu Sun**[1], **Houfeng Wang**[1,2], **Mengxiang Wang**[3]
[1] MOE Key Lab of Computational Linguistics, Peking University, Beijing, 100871, China
[2] Collaborative Innovation Center for Language Ability, Xuzhou, Jiangsu, 221009, China
[3] Teachers' College of Beijing Union University, Beijing, 100011, China
{hhhwang, zxdcs, shumingma, xusun, wanghf}@pku.edu.cn
41326498@qq.com

## Abstract

Most question answering (QA) systems are based on raw text and structured knowledge graph. However, raw text corpora are hard for QA system to understand, and structured knowledge graph needs intensive manual work, while it is relatively easy to obtain semi-structured tables from many sources directly, or build them automatically. In this paper, we build an end-to-end system to answer multiple choice questions with semi-structured tables as its knowledge. Our system answers queries by two steps. First, it finds the most similar tables. Then the system measures the relevance between each question and candidate table cells, and choose the most related cell as the source of answer. The system is evaluated with TabMCQ dataset, and gets a huge improvement compared to the state of the art.

## 1 Introduction

Question answering is a practical task, and there are many related challenges. Due to the complexity of natural language, those challenges are hard to solve. A QA model requires knowledge, which contains facts, and fetches the answer from the knowledge. For most QA systems, raw text and structured knowledge graph are used as their knowledge. However, raw text corpora are hard to understand for machine. Besides, most knowledge graphs contain too much noise and still need manual intervention. In contrast, the semi-structured data are more flexible to be comprehended than raw text corpora, and much easier to build from text automatically than knowledge graphs. Besides, there are many documents which can be easily converted to semi-structured tables, such as announcements published by institutions and knowledge in science books, etc. We would like to utilize these kinds of knowledge in QA systems.

There are many works on QA based on other knowledge. Some of them are based on raw text corpora, such as Miller et al. (2016), Min et al. (2017), Yin et al. (2016). Besides, QA models based on knowledge graphs are improving rapidly. Freebase (Bollacker et al., 2008) and Wikidata (Vrandecic and Krötzsch, 2014) are well-known structured knowledge bases, which are built almost manually by their users. Although there are some studies on automatic knowledge graph construction, like Sateli and Witte (2015), it is not good enough. There are also many studies on those structured data, like Yih et al. (2015), Yao and Durme (2014). Some of them focused on text description of each item, while others, such as Zhang et al. (2016), try to obtain embeddings of the items.

Although there is limited work on semi-structured tables, there has already been research of QA based on simple tables. HILDB (Dua et al., 2013) is a QA system which converts questions in natural language to SQL queries. Vakulenko and Savenkov (2017) offer another data structure of tables, and introduce a QA system based on tabular knowledge. However, those methods are limited to the specific structure of tables, and cannot take advantage of semi-structured data.

In previous work, candidate table selection is based on manually selected features, such as TF-IDF, which makes it hard to transfer to other environment. Besides, all columns in each row are considered equally, but for some questions, some of columns in the table are completely unrelated to question, and those cells can mislead the model to give a wrong answer.





*Proceedings of the 27th International Conference on Computational Linguistics*, pages 1941–1951
Santa Fe, New Mexico, USA, August 20-26, 2018.

Resource Type Organism

| RESOURCE for organisms | | RESOURCE TYPE | RESOURCE SOURCE | | DIRECT OR INDIRECT? | | ORGANISM CLASS | |
|---|---|---|---|---|---|---|---|---|
| sunlight | is | light energy | from the Sun | that is | directly | required by | plants | to survive |
| sunlight | is | light energy | from the Sun | that is | indirectly | required by | all living things | to survive |
| water | is | matter | available on Earth | that is | directly | required by | all living things | to survive |
| air | is | matter | available on Earth | that is | indirectly | required by | all living things | to survive |
| air | is | matter | available on Earth | that is | directly | required by | animals | to survive |
| air | is | matter | available on Earth | that is | directly | required by | plants | to survive |
| food | is | matter | created using water, air, and sunlight | that is | directly | required by | animals | to survive |
| food | is | chemical energy | created using water, air, and sunlight | that is | directly | required by | animals | to survive |
| habitat | is | an environment | on Earth | that is | directly | required by | animals | to survive |

What is an example of light energy indirectly required by all living things?
A. Air B. habitat C. food D. sunlight

Figure 1: The structure of semi-structured table knowledge and one of its corresponding MCQs.

In this paper, we will propose a neural system for answering multiple choice questions (MCQs) based on semi-structured tabular data. Our model contains two parts: 1) *candidate table selection*: for each query, tables are ranked according to relevance to the question, and the model fetches candidate cells from the most relevant tables, 2) *cell scoring and answer selection*: the model measures the relevance of query and each candidate cell, and finally the model obtains an answer based on the most relevant cells to the query.

Our first contribution is offering an end-to-end model to select candidate tables before scoring and ranking the answers, instead of using hand-crafted features in previous work. According to our evaluation, recall of the model on the top three tables is more than 98.9%, which leads the system to be efficient and accurate.

Another contribution of our work is a new relevance scorer between table cells and text. We use an attention layer to make the model focus on useful information of the table. With effort of the new scorer, our system has better performance than state-of-the-art on the task. Besides, the model has fewer parameters, which makes it easier to be trained.

Base on our evaluation, our system gets a high performance on the answer scoring task of TabMCQ. Our system achieves an accuracy of 79.0% on test set, while the state-of-the-art system TabNN (Jauhar, 2017) only answers 56.8% of questions correctly.

## 2 Task Description

The knowledge contains some semi-structured tables $\mathbb{T} = \{T_i | i \in 1...n\}$. And each case contains a MCQ query. There is an example of knowledge and queries in Figure 1. There are several semi-structured tables in its knowledge base.

Each semi-structured table has a title, like *Resource Type Organism* in Figure 1. Different from structured tables, semi-structured tables have some columns used to link other cells, which makes each row is a complete sentence, and those columns are not tagged. Except for those columns, each column in the tables has a tag.

Each query contains a question and three or four candidate choices, i.e. $Q = (q, c_1, c_2, c_3, c_4)$. Our task is to predict the correct choice $C$ of query $Q$ based on $\mathbb{T}$. We assume that each case has one correct answer. In training set, each case contains a MCQ query $Q$, its correct choice $C$, and source cells of answer $\left\{T_i(j,k) | i \in 1...n, j \in 1...d_r^{T_i}, k \in 1...d_c^{T_i}\right\}$, where $T_i(j,k)$ is the cell at $i$-th row and $j$-th column of $T_i$, and $d_r^{T_i}$ and $d_c^{T_i}$ are the numbers of rows and columns of $T_i$. The system is evaluated by accuracy of correct selections.

## 3 System Architecture and Training

To narrow down the range of candidate tables, our system uses an end-to-end model which has high recall to filter out unrelated tables. After that, our model searches in candidate tables, and obtains a list

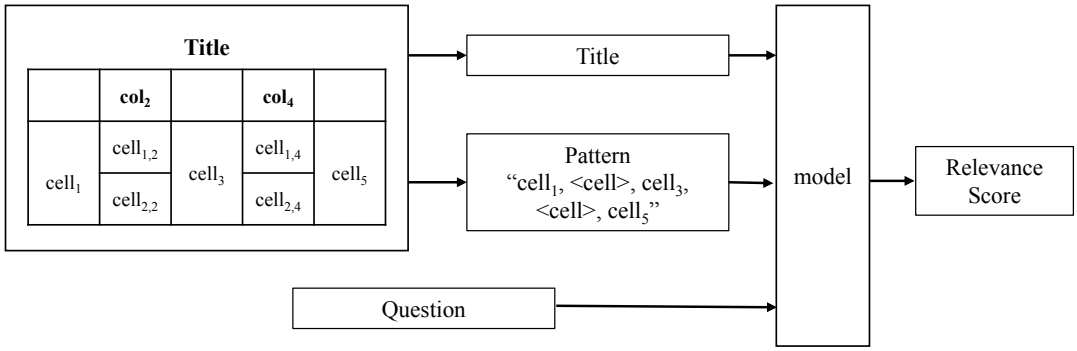


Figure 2: The system structure to select candidate tables.

of candidate cells. Then an attentive recurrent model will measure the relevance of all these cells with the query.

### 3.1 Candidate Table Selection

In this section, we will introduce the model to select related tables of a given question. The structure is shown in Figure 2. We use DiSAN (Shen et al., 2017) to encode the tables and questions separately, and then measure relevance. The tables with highest relevance scores are selected as input of answer selection model.

**Directional Self-Attention Network (DiSAN)**: DiSAN is a network to learn sentence representations, instead of convolution neural networks (CNN) and recurrent neural networks (RNN), it is based on the attention mechanism. DiSAN is made up by directional self-attention (DiSA) blocks. For each block, assume the input sequence is $x = [x_1, x_2, ..., x_n]$, we have

$$\boldsymbol{h} = \sigma_h(W^h \boldsymbol{x} + b^h), \tag{1}$$

where $\sigma_h$ is an activation function. Then given masks $M$, we can use masks to encode temporal order information into the output. In DiSA, the masks are the forward mask and the backward mask, i.e.

$$M_{ij}^{fw} = \begin{cases} 0 & i < j, \\ -\infty & \text{otherwise.} \end{cases}, \tag{2}$$

$$M_{ij}^{bw} = \begin{cases} 0 & i > j, \\ -\infty & \text{otherwise.} \end{cases}, \tag{3}$$

Define $p(z_k = i|x, q)$ as the probability of the k-th feature of token $x_i$ contributes important information to $q$, and $\mathbf{1}$ represents all-one vector. In the multi-dimensional attention layer, we have a feature-wise score vector instead of a single scalar score, that is

$$f(h_i, h_j) = c \cdot \tanh\left([W^{(1)} h_i + W^{(2)} h_j + b^{(1)}]/c\right) + M_{ij}\mathbf{1}, \tag{4}$$

$W^{(1)}, W^{(2)} \in \mathbb{R}^{d_h \times d_h}$ and $b^{(1)} \in \mathbb{R}^{d_h}$ are parameters. When $M = M^{fw}$, only the relation of later $j$ and earlier $i$ will be in attention, and when $M = M^{bw}$, only earlier $j$ and later $i$ will be focused. Now, we can obtain the importance weight of each feature $k$ in each token $i$, i.e.

$$P(z_k = i|\boldsymbol{h}, h_j) = \text{softmax}(f(h_i, h_j)). \tag{5}$$

Then, the output of $x_j$ can be written as

$$\boldsymbol{s} = [\mathbb{E}_{i \sim p(z_k|\boldsymbol{h}, h_j)} x_{ki}]_{k=1}^{d_e} = \sum_{i=1}^{n} p(z.|\boldsymbol{h}, h_j) \odot x_i. \tag{6}$$

| | CHARACTERISTIC Physical characteristic of an animal or human | | INHERITED? Is the characteristic inherited, learned, or acquired? |
|---|---|---|---|
| An | facial scar | is | acquired |
| | blue eyes | are | inherited |
| | long hair | is | acquired |
| A | broken leg | is | acquired |
| | strong muscles | are | acquired |
| | telling a story | is | learned |

Figure 3: An example of the tables with a few link words.

The final output $u$ of a DiSA block is gotten by input $h$ and output $s$ of the block by a fusion gate.

$$F = sigmoid(W^{(f1)}h_i + W^{(f2)}h_j + b^{(f)}), \tag{7}$$

$$\boldsymbol{u} = F \odot \boldsymbol{h} + (1 - F) \odot \boldsymbol{s}, \tag{8}$$

$W^{(f1)}, W^{(f2)} \in \mathbb{R}^{d_h \times d_h}$ and $b^{(f)} \in \mathbb{R}^{d_h}$ are parameters to be learned. DiSAN firstly applies forward blocks and backward blocks, and their outputs $\boldsymbol{u}^{fw}, \boldsymbol{u}^{bw} \in \mathbb{R}^{d_h \times n}$ are concatenated and sent to another self-attention block called source2token block, which compresses $[\boldsymbol{u}^{fw}, \boldsymbol{u}^{bw}]$ into a single vector, i.e.

$$f(u_i) = W^T(W^{(1)}u_i + b^{(1)}) + b, \tag{9}$$

$$s_{\text{disan}} = [\mathbb{E}_{i \sim p(z_k|u)} u_{ki}]_{k=1}^{d_e} = \sum_{i=1}^{n} p(z.|\boldsymbol{u}) \odot u_i \tag{10}$$

Compared to other methods, DiSAN contains fewer parameters and takes less time to train, and it works well on encoding tables and questions separately. In table selection, there are many query-table pairs, and we need to score them accurately and time-efficiently. Inspired by this method, when tables and questions are encoded, they are self-attentioned without extra information, which can reduce computation complexity of this part.

**Candidate Table Selection Model**: For each (query, table) pair, we will score the relevance between them. Consider a table $[x, T]$ and a question $q = \{w_{i,q}\}_{i=1}^{n}$, where $x$ is the title of a table, $T$ is content of the table, and $w_{i,q}$ is the $i$-th word in $q$.

We first convert words in questions into representations $I_q = \mathbb{R}^{d_w \times n}$, which are concatenation of the word-level embeddings of the words $\{e_{w_{i,q}}\}_{i=1}^{n}$ and their POS-tag embeddings $\{e_{p_{i,q}}\}_{i=1}^{n}$. The embeddings of POS tags are randomly initialized and are trained together with parameters, and the word-level embeddings are initialized by GloVe (Pennington et al., 2014) vectors. We then use a DiSAN layer described above to encode each question $q$:

$$E(q) = \text{DiSAN}(I_q). \tag{11}$$

Then we convert tables into their vector representations. A table contains a title and some structured data, and we encode them respectively. In some tables, there are some columns filled with link words, and each row in the table is a complete sentence, while others have few or even no link words (like Figure 3). We put those columns, which link the cells, and tags of other columns into a DiSAN encoder, and then, we measure the relevance between each table and the question by their representations, i.e.

$$E([x, T]) = W[\text{DiSAN}(I_x), \text{DiSAN}(T)] + b, \tag{12}$$

$$o(q, [x, T]) = \cos\left(E(q), E([x, T])\right). \tag{13}$$

For each question, we select the $l$ most relevant tables. Then, rows in those tables are selected by their intersection of text with the query. Those rows are sent into the cell scoring model.

Question: Which period of daylight is the summer solstice related?

| | ORBITAL EVENT | | PERIOD OF DAYLIGHT | | PERIOD OF NIGHT | |
|---|---|---|---|---|---|---|
| The | summer solstice | is the day with the | longest | period of daylight and the | shortest | period of night |
| The | winter solstice | is the day with the | shortest | period of daylight and the | longest | period of night |
| The | spring equinox | is the day with the | midrange | period of daylight and the | midrange | period of night |
| The | fall equinox | is the day with the | midrange | period of daylight and the | midrange | period of night |

Figure 4: The table contains the answer of "Which period of daylight is the summer solstice related?", relevant rows and columns, and their intersection are marked.

### 3.2 Cell Scoring and Answer Selection

After finding candidate rows, we introduce a model that focuses on useful parts of tables and tries to score each answer by its relevance with the query. For most queries, answers can be selected based on cells from candidate tables, so we score each cell directly. As Figure 4 shows, for question "Which period of daylight is the summer solstice related?", we can find a row in table, which tells us summer solstice has longest period of daylight and shortest period of night, but "shortest period of night" has nothing to do with answering the query, and may mislead model to make a wrong decision. Therefore, to focus on useful information of the table, a soft attention layer is on the top of table encoder. The structure of our answer selection model is shown in Figure 5.

**Soft Attention**: The $r$-th row of table T contains several cells $\{T(r,i)\}_{i=1}^{d_c}$. The attention layer needs to find out useful cells in the candidate row (like those marked in Figure 4). When we obtain attention weights, rows in the table will be transformed to the unstructured sentences. Suppose we have $m$ words in $q$ and $n$ words in $T(c)$, for each query $Q$ and candidate row, we build a matrix $S_{q,T(r,\cdot)} = \{\text{sim}(i,j)\}_{m\times n}$, $\text{sim}(i,j)$ is cosine similarity between the vector representation of the $i$-th word of $q$ and $j$-th word of sent$(T(r,\cdot))$. Then, attention weights $W_a$ are calculated by a CNN, which have $S$ as its input. Convolution layers are used to measure the similarity of n-grams between $q$ and sent$(T(r,\cdot))$, and a pooling layer on the dimension with variable length is applied before we obtain $W_a$, i.e.

$$W_a = \tanh(\text{pool}(\text{conv}(S))). \tag{14}$$

Here, we add an activation layer to make sure the weights are in a small range.

**Cell Scoring**: In most cases, the answer candidate cells in each row are not components of a question. Therefore, for each $(q, T(r,c))$, input of the encoder does not contain the candidate answer cell $T(r,c)$, and it is replaced by a tag "⟨SPACE⟩". For some rows in same table, if we remove the answer cell, they are in same pattern (such as, in the table "Country Hemispheres", both "China" and "Japan" are in the north hemisphere, if we remove "country" cells, those two rows are both in pattern "⟨SPACE⟩ / is in / north hemisphere"). For these cells, we merge them into one group. Also, a row in the table is hard to be encoded directly, we convert each row into a word sequence, in which different cells in a row are separated by sign "||".

Suppose we have a question $q$ with $n$ words, and a row $T(r)$ with $m$ words, let $I_q \in \mathbb{R}^{d_w\times n}$ and $I_{T(r,c)} \in \mathbb{R}^{d_w\times m}$ denote the embeddings of words in question $q$ and row $T(r)$, where $d_w$ is the dimension of word representations. Question encoding $E(q)$ and $E(q,T(r,c))$ can be obtained through a bidirectional Gated Recurrent Unit (Bi-GRU) Network:

$$E(q) = \text{Bi-GRU}(I_q), \tag{15}$$

$$E(q,T(r,c)) = \text{Bi-GRU}(W_a \cdot I_{T(r,c)}). \tag{16}$$

We concatenate the final states of two directions of Bi-GRU and take it as the representation of the question and the selected cells. Then for each $(q,T(r,c))$, we put $E(q)$ and $E(q,T(r,c))$ into a fully-connected layer and then obtain the score of each pair, i.e.

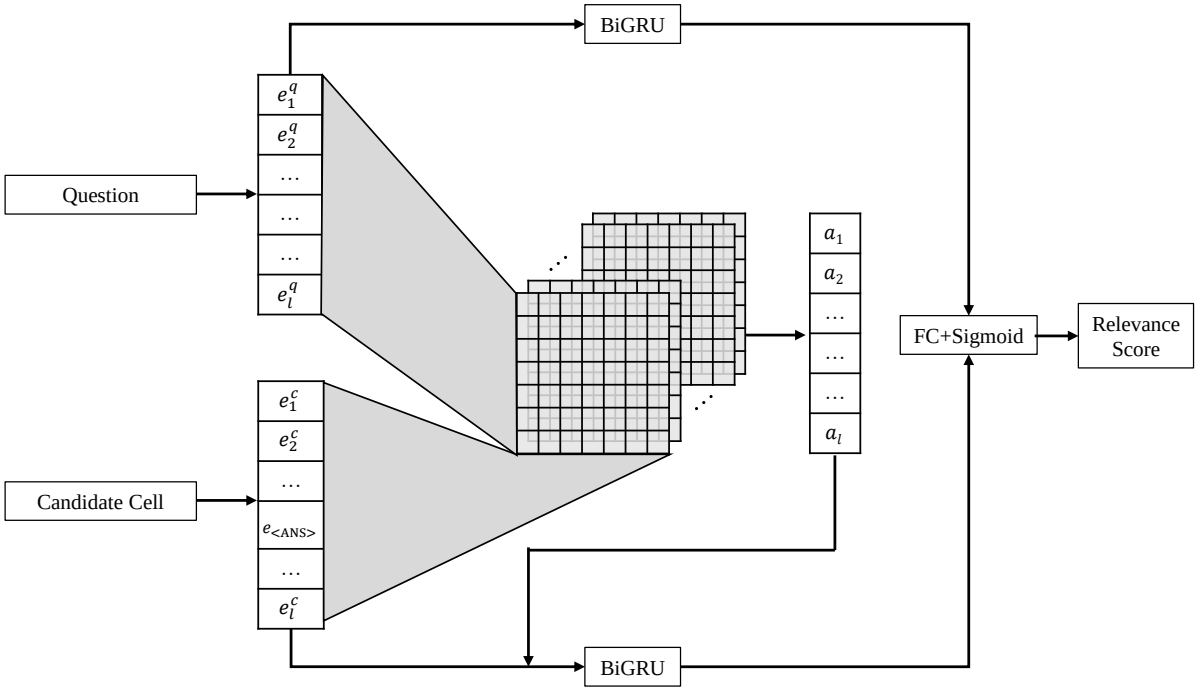


Figure 5: The structure of the answer scoring model.

$$s = rel(q, T(r, c)) = \text{sigmoid}(W \cdot \text{FC}(E(q), E(q, T(r, c))) + b). \tag{17}$$

**Answer Selection from the candidate cells**: After we score each rows, we need to select the best answer from the three or four candidates. Because of some queries have a choice like "none of above", we should also determine whether there are true answers in all choices. For each query $Q$, we order all patterns by their scores, and for each pattern, we will find the matched words between selected cells and the choices. For those queries can not be determined by the most relevant patterns, the next pattern will be used to select choices and get its intersection with the answer set. The process finishes when only one choice is remained, or no answer is left in the answer set.

Our model is a scorer model, which can answer MCQs with multiple answers. There are two different kinds of those queries. In some of those queries, the answers are in same pattern set, and it is easy to find them out. For those cases that answers are in different set, a straightforward solution is selecting candidates of which the score is above a threshold as answers. However, there is no such case in both the training and testing set, we did not explore the way to measure the threshold.

### 3.3 Training

Instead of measuring the system by an absolute score for relevant and irrelevant cells, we train the model to score the best cell with the highest score. For each query, we select some wrong cells from the candidate tables of each query randomly, and train the model based on triples $(q, T(r_+, c_+), T(r_-, c_-))$. We measure the loss of each triple by:

$$L(q, T(r_+, c_+), T(r_-, c_-)) = \max(0, \alpha - s_+ + s_-) \tag{18}$$

In (18), $s_+$ and $s_-$ are relevance scores between question and $T(r_+, c_+)$, $T(r_-, c_-)$, and $\alpha$ is a hyper-parameter to control the gap between the two scores.

## 4 Experiment

In this section, we will evaluate our QA system based on semi-structured knowledge. Firstly, we will introduce our dataset, evaluation metrics and setup. Then, we compare our system with other work.

Finally, we analysis the result of evaluation.

### 4.1 Dataset

We use the TabMCQ (Jauhar et al., 2016) dataset to evaluate our system. TabMCQ contains 9092 manually annotated multiple choice questions (MCQs) with their answers, and 63 tables as its knowledge. Tables in this task are semi-structured tables, rows in each table are sentences with well-defined recurring filler patterns. For those tables built by sentences, some of the tables contain some link words to make the sentence complete, while in other tables, all cells are meaningful. Same as simple tables, analogies between rows of tables also exist.

The target domain for the tables is the *4th grade science exam*, and most tables are constructed based on the topic. MCQs in the dataset are created by information of the row containing the target cell, and the other choices of the question should be built according to other cells of the same table. Therefore, for all MCQs in the dataset, we know the source cell to answer the MCQs, which makes it easier to train based on the dataset.

Jauhar (2017) also evaluates his model on Elementary School Science Questions (ESSQ) dataset. However, the structure of those two datasets are quite different. Length of questions in TabMCQ is much shorter than ESSQ, and in ESSQ, many questions can not be solved by searching, but analogy and reasoning. Our system is not aimed at working on these questions. Besides, ESSQ contains only 108 MCQs in its training set, which is hard to train a model. Therefore, ESSQ is not in the sources of evaluation.

### 4.2 Evaluation Metric

For each query, there are three or four choices. We measure the systems by their accuracy on answering questions. The cases will be set to wrong when the correct group is selected, but the model returns the wrong choice. On table selection task, the goal is to get more correct recall in limited return, so we evaluate the models by their mean average precision (acc@$n$), which means the proportion of correct table of the query in $n$ most relevant tables returned by the model.

### 4.3 Experiment Setup

When preprocessing the corpus, we use tokenizer from Natural Language Toolkit (NLTK) (Bird, 2006). The representations of words are pre-trained by GloVe (Pennington et al., 2014), and all these embeddings are fine-tuned in the training process. The dimension of word representations in all components is 300.

Two parts of the system are trained separately. For the DiSAN layer, we have a dropout layer (Srivastava et al., 2014) with dropout rate being 0.2, and the activation function is ELU (Clevert et al., 2015). The model is trained as a classifier, whose loss is measured by cross entropy with softmax, and we minimize the loss by using Adagrad (Duchi et al., 2011) with learning rate set to 0.05. When getting attention weights, we apply padding and masking to make input into fix length. Candidate cells of the answer scoring model are from table selection model we describe in section 3.1. We use the three tables with the highest scores as its source. The size of the GRU hidden layer is 100, and input and output of each GRU layer have a dropout rate of 0.2. We train the model on batches of 32 queries, and for each query, a correct answer and a wrong answer are used to construct a training sample. Attention layer is trained together with the whole model, and the loss of the answer scoring model is minimized by Adadelta (Clevert et al., 2015) with learning rate set to 0.08. Representation of words in our system is trainable, and we evaluate our model on both pre-trained word embeddings and randomly initialized word embeddings.

To compare with other work, we split queries from TabMCQ into three parts, 10% of queries are in testing set, 10% of the rest queries are in validation set, and the remaining queries are in training set. We train our system on the training set, and finish training when accuracy on the validation set stops increasing. Besides, we also evaluate our system without pre-trained vector and attention.

### 4.4 Baselines

To evaluate our system, we compare our system with other work. We take some experiments to show the influence of each components of our system. For systems that need to be trained (Bi-LSTM, TabNN, and our model), models are learned by using training set and validation set, and evaluations on all models are based on test set.

**Random**: The answer is selected from all choices randomly. It's used to measure whether the model gains the result on this task. Because there are a few questions with only three choices, the accuracy of random selection is a little higher than 0.25.

**Bi-LSTM**: Each row is converted into a sequence, and put into a bidirectional LSTM network. The hidden size of the LSTM layer is 100 or 200. The model is trained by Adadelta. The model's output is the relevance score of query and row. Candidate rows of the query is obtained from the model in 3.1, and the final choice is selected by matched words.

**Bi-GRU**: Structure of the system is similar with Bi-LSTM, but the recurrent layer is changed to bidirectional GRU. The hidden size of the model is 120 or 200. The model is trained by Adadelta.

**Lucene**: Lucene is a tool for retrieving unstructured information, and it is well known as a search engine. We convert semi-structured tables into unstructured sentences, which can be indexed and searched by Lucene. For each query, the top hit returned by Lucene is regarded as the answer, and the sentence is compared with the choices, and the choice with most coincidence is selected.

**TabNN**: TabNN, proposed by (Jauhar, 2017), represents the state-of-the-art on the task. It is composed of two scorers: a scorer between questions and rows and the other one between choices and columns. In each component, the basic unit of table encoder is cell-LSTM, which is an LSTM layer to encode cells. Output of cell-LSTM can be used by the following row and column encoders as their input. In the evaluation, TabNN-fixed is the model with fix word representations, while TabNN-learned is the model with trainable word representations.

### 4.5 Results

For the whole system, accuracy on the test set of TabMCQ is shown in Table 1. Count of parameters does not include those from word representations. As we can see from Table 1, our system performs better than existing work by a large margin. Because our training is based on (query, relevant cell, improper cell) triples, and does not train by all wrong cells directly, test accuracy of our system is very close to train accuracy. Training accuracy of our model is 79.27%, while TabNN has accuracy of 68.0% on the training set, and 56.8% on the test set.

Besides, our model is less complicated than TabNN. As we can see from Table 1, our scorer has only half of the parameters, compared to the number of TabNN. In most deep learning system trained by a small dataset, model with fewer parameters can be trained more easily than complicated ones, and is less likely to overfit.

For different components of our system, soft attention is significant. As we can see from the Table 1, the system with Bi-GRU as its scorer works poor in the task, although it has similar settings in other parts with our model. That is because ordinary models are hard to classify whether the information in the tables is useless or not, but a soft attention layer can help filter out the cells, which helps ordinary models work much better. Our system benefits from using pre-trained word embedding to initialize our system.

### 4.6 Analysis of Candidate Selection

In section 3.1, we offer a table selection model. To evaluate the effort of the model, we compare it with some baselines, which is shown on Table 2. In our experiment, LSTM with attention has a hidden size of 120, and CNN has filters in size $(n, d_w)(n = 2, 3, 5)$ and a fully-connected layer with activation. Each of them is trained by AdaGrad optimizer.

As we can see from the result, LSTM takes a poor performance on selecting the most relevant table of the question. On acc@1, both our model and CNN work well, but the correct tables achieve higher score

| Systems | Hidden Size | # Parameters | Test Accuracy |
|---|---|---|---|
| Random | - | - | 25.1% |
| Lucene | - | - | 52.2% |
| Bi-LSTM | 100 | 807,005 | 32.9% |
| Bi-LSTM | 200 | 1,930,205 | 32.2% |
| Bi-GRU | 120 | 755,005 | 35.4% |
| Bi-GRU | 200 | 1,445,405 | 34.7% |
| TabNN-fixed (Jauhar, 2017) | 160 | 1,131,843 | 31.9% |
| | 320 | 3,902,083 | 30.9% |
| TabNN-learned (Jauhar, 2017) | 160 | 1,131,843 | 56.8% |
| | 320 | 3,902,083 | 50.8% |
| Ours w/o pre-trained | 100 | **605,846** | 75.1% |
| Ours | | | **79.0%** |

Table 1: Comparison between systems.

| Model | Acc@1 | Acc@2 | Acc@3 |
|---|---|---|---|
| LSTM with attention | 89.9% | 95.7% | 96.3% |
| CNN | 94.2% | 96.0% | 97.1% |
| Ours (based on DiSAN) | 96.4% | 98.4% | 99.0% |

Table 2: Comparison between models on table selection.

on our model, which means our model selects fewer tables on the same recall of the accurate tables, and the system can obtain the correct answer more easily than using CNN.

Besides, our model runs faster than LSTM, and has almost the same running time with CNN for each batch of data. Therefore, DiSAN works well on this task in our model. There are much more relevant tables when using the three most relevant tables than the top two, and the proportion of relevant tables is not selected is small (acc@4 of our model is 99.0%, and acc@5 is 99.2%, the gap between acc@3 and acc@5 is smaller than those shown in table 2, and the improvement is little on using more candidate tables. Therefore, for each query, we have three candidate tables sent into answer scoring.

### 4.7 Error Analysis

Although our system is much better than existing work, there is also a big room to improve. We randomly analysis some queries that our system can not select the correct choice, including those from the training set and the test set.

Some of the errors are due to candidate selection. Although we choose the three most relevant tables as candidates, there are still a few queries matched with wrong tables. Most of those queries' corresponding tables contain only few columns, and has little feature on their patterns. An example is shown on Figure 3. Besides, in some queries, words in different patterns cannot be recognized by the system. For example, the correct choice is "USA", while the representation in the relevant cell is "the united states". It's hard to recognize by our model, and it can be solved by using large knowledge to get better representations of those words. Otherwise, for some question, we have obtained the correct group with several cells, but our answer selection method is not accurate enough, the wrong choice is selected based on other cells in the same group.

There are also some faults of tagging in the dataset. Some queries in the training set are labeled with wrong cells as their "relevant cell", such as the question "Yucatan is a state located in ?", is asking for the country of Yucatan, the "relevant cell" should be "country", but the question is marked as "subdivision". Besides, there are a few questions with two correct choices. For question "Which of the following is a carnivore?", both "Andean Cat" and "Arctic fox" are in choices, and only one of them is marked as correct.

## 5 Conclusion

In this paper, we present a question answering system based on semi-structured tabular data. Our system is composed of two parts, which are used to select candidate tables and give out answers. Then we take some experiments with different settings on the system, and our model is compared with previous work by their performance. Besides, we evaluate the effect of the components of the model. Due to the effect of attention layer and efficient model, our system performs better than state-of-the-art work on TabMCQ dataset.

In further research, there are still much work to be done. MCQs with multiple correct choices can be added to the dataset, and then the system can solve answer MCQs with multiple answers. Besides, recall of table selection model still can be improved. Making decision on choice selection more accurate is another important point. Answering questions with induction and reasoning is also a point to improve. Otherwise, representation of cells are also important. An embedding with both text and structure information can be helpful to improve the model. Otherwise, the manners to convert raw text and structured-tables to semi-structured tables can also be explored.

## Acknowledgements

We would like to thank for anonymous reviewers for their helpful feedback. Our work is supported by National Natural Science Foundation of China under Grant No.61333018 and the National Key Research and Development Program of China under Grant No.2017YFB1002101. The corresponding author of this paper is Houfeng Wang.

## References

Steven Bird. 2006. NLTK: the natural language toolkit. In *ACL 2006, 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, Sydney, Australia, 17-21 July 2006*.

Kurt D. Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, pages 1247–1250.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2015. Fast and accurate deep network learning by exponential linear units (elus). *CoRR*, abs/1511.07289.

Mohit Dua, Sandeep Kumar, and Zorawar Singh Virk. 2013. Hindi language graphical user interface to database management system. In *12th International Conference on Machine Learning and Applications, ICMLA 2013, Miami, FL, USA, December 4-7, 2013, Volume 2*, pages 555–559.

John C. Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Sujay Kumar Jauhar, Peter D. Turney, and Eduard H. Hovy. 2016. Tables as semi-structured knowledge for question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Sujay Kumar Jauhar. 2017. *A Relation-Centric View of Semantic Representation Learning*. Ph.D. thesis, Carnegie Mellon University.

Alexander H. Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1400–1409.

Sewon Min, Min Joon Seo, and Hannaneh Hajishirzi. 2017. Question answering through transfer learning from large fine-grained supervision data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 2: Short Papers*, pages 510–517.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543.

Bahar Sateli and René Witte. 2015. Automatic construction of a semantic knowledge base from ceur workshop proceedings. In *Semantic Web Evaluation Challenge*, pages 129–141. Springer.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2017. Disan: Directional self-attention network for rnn/cnn-free language understanding. *CoRR*, abs/1709.04696.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Svitlana Vakulenko and Vadim Savenkov. 2017. Tableqa: Question answering on tabular data. *CoRR*, abs/1705.06504.

Denny Vrandecic and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with freebase. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 956–966.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1321–1331.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2016. ABCNN: attention-based convolutional neural network for modeling sentence pairs. *TACL*, 4:259–272.

Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 353–362. ACM.