INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL
Online ISSN 1841-9844, ISSN-L 1841-9836, Volume: 16, Issue: 2, Month: April, Year: 2021
Article Number: 4115, https://doi.org/10.15837/ijccc.2021.2.4115

CCC Publications

AGORA UNIVERSITY PRESS

# Extract Executable Action Sequences from Natural Language Instructions Based on DQN for Medical Service Robots

F. D. Zhao, Z. K. Yang, X. S. Li, D. D. Guo, H. T. Li

**Fengda Zhao**
1. School of Information Science and Engineering
Yanshan University
Qinhuangdao 066004, China
2. Xinjiang University of Science and Technology
Korla 841000, China
3. The Key Laboratory for Software Engineering of Hebei Province
Yanshan University
Qinhuangdao 066004, China
zfd@ysu.edu.cn

**Zhikai Yang, Dingding Guo, Haitao Li**
1. School of Information Science and Engineering
Yanshan University
Qinhuangdao 066004, China
2. The Key Laboratory for Software Engineering of Hebei Province
Yanshan University
Qinhuangdao 066004, China
ysuyangzhikai@stumail.ysu.edu.cn
guodingding@ysu.edu.cn
xjlht@ysu.edu.cn

**Xianshan Li***
1. School of Information Science and Engineering
Yanshan University
Qinhuangdao 066004, China
2. The Key Laboratory for Software Engineering of Hebei Province
Yanshan University
Qinhuangdao 066004, China
*Corresponding author: xjlxs@ysu.edu.cn

**Abstract**

The emergence and popularization of medical robots bring great convenience to doctors in treating patients. The core of medical robots is the interaction and cooperation between doctors and robots, so it is crucial to design a simple and stable human-robots interaction system for medical robots. Language is the most convenient way for people to communicate with each other, so in this paper, a DQN agent based on long-short term memory (LSTM) and attention mechanism is proposed to enable the robots to extract executable action sequences from doctors' natural language instructions. For this, our agent should be able to complete two related tasks: 1) extracting action names from instructions. 2) extracting action arguments according to the extracted action names. We evaluate our agent on three datasets composed of texts with an average length of 49.95, 209.34, 417.17 words respectively. The results show that our agent can perform better than similar agents. And our agent has a better ability to handle long texts than previous works.

**Keywords:** medical robots, human-robots interaction, DQN agent, attention mechanism, long-short term memory.

## 1 Introduction

In recent years, medical robots have become a hot spot for the development of the robotics industry and the medical industry, they are able to help doctors treat patients more efficiently. For example, surgical robots can be used for surgical image guidance and minimally invasive surgery, rehabilitation robots can be used to recover from limb injuries, assist the disabled to complete actions and to improve their athletic ability, auxiliary robots can be used to assist doctors to diagnose the patient's condition and make a real-time observation on the health condition of the elderly [1], service robots can be used to move medical supplies and clean up medical waste in isolation wards to reduce the risk of infection among medical staff. One of the core technologies is the interaction between humans and robots, but traditional robots with keyboard-and-mouse or touch-screen interfaces are not easy to do this. Thus, it is critical to provide a simple way for doctors to interact with and control robots.

People use language every day to direct behavior, ask for help, and communicate with others. Thus, more and more researchers are devoted to enable robots to interact with users by natural language. One of the bases of it is to let robots understand the natural language instructions given by users. Over the past five years, a large number of methods based on semantic framework and sequence to sequence models were applied to this task [2, 3].

Reinforcement learning (RL) [4] has been received much attention because of its excellent performance in areas such as mobile robots navigation [5, 6], decision-making system [7, 8]. Recently, reinforcement learning has been a viable and powerful approach in the area of natural language processing (NLP). In 2018, Feng et al. [9] presented an intelligent agent named EASDRL to extract executable action sequences from free and long natural language instructions. However, EASDRL is a Deep Q-Network (DQN) agent based on convolutional neural networks (CNN), resulting in the difficulty for the agent to understand the complex context of the actions in instructions and it becomes unstable when the instruction is too long because it is not able to solve the long-range dependencies problem of texts.

In this paper, we propose a novel DQN agent (as shown in Figure 1) based on bidirectional LSTM (BiLSTM) [10] and attention mechanisms to solve the above problems. LSTM is proposed to solve the problem of long-range dependencies in text. It has a better ability to remember long-term information than traditional neural networks, so LSTMs are more suitable for processing long texts than traditional neural networks.

Although LSTMs have the ability to process long text information, it seems not enough because of their forgetfulness. In this paper, the models are trained on natural language instructions composed of hundreds of words and phrases. Thus, we apply attention mechanisms to our DQN agent to reduce the impact of the loss of features on our experimental results. As a core technology, attention mechanisms have been widely used in the field of natural language processing (NLP). They are first used in machine translation tasks to solve the problem of long-range dependencies of traditional seq2seq models by using global text information to compute the output of the encoder.

We evaluate the performance of our agent on three datasets, i.e., "Microsoft Windows Help and Support" (WHS) documents [11], and two datasets "WikiHow Home and Garden"(WHG) and "CookingTutorial"(CT) labeled by [9]. The results of the experiments show that our DQN agent based on BiLSTM and attention mechanism performs better than existing similar approaches.

Our major contributions are as follows:

- Compared with the deep learning-based methods, our method based on deep reinforcement learning has a better ability to extract action sequences from the text that contains multi actions.

- We build our agent based on BiLSTM and attention mechanisms in order to solve the problem of long-range dependencies of long texts and fully utilize the term context relations in text. Thus, our agent has a better ability to extract action sequences from long texts than the agents presented by previous works.

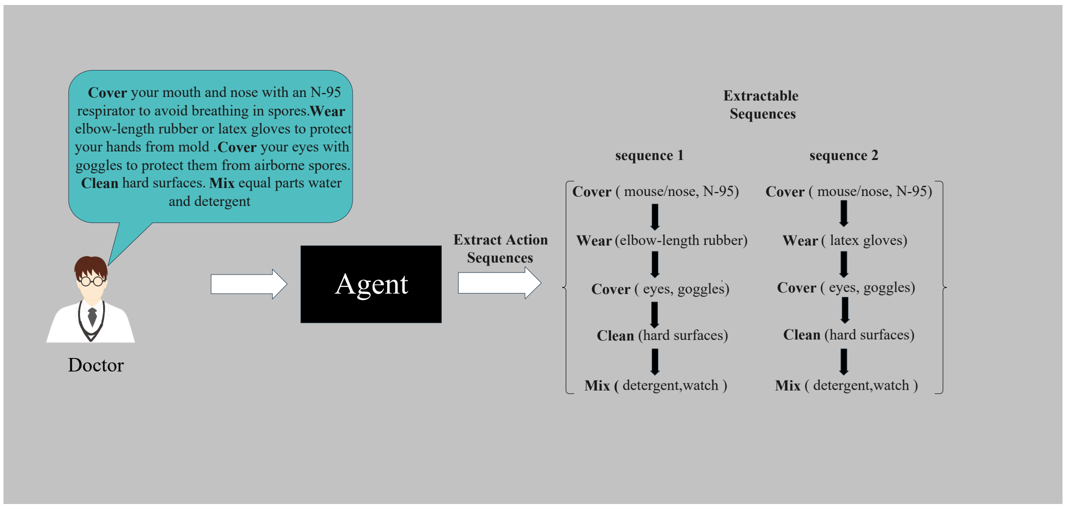

Cover your mouth and nose with an N-95 respirator to avoid breathing in spores.Wear elbow-length rubber or latex gloves to protect your hands from mold .Cover your eyes with goggles to protect them from airborne spores. Clean hard surfaces. Mix equal parts water and detergent
Doctor
Agent
Extract Action Sequences
Extractable Sequences
sequence 1
sequence 2
Cover ( mouse/nose, N-95)
Wear (elbow-length rubber)
Cover ( eyes, goggles)
Clean (hard surfaces)
Mix ( detergent,watch )
Cover ( mouse/nose, N-95)
Wear ( latex gloves)
Cover ( eyes, goggles)
Clean (hard surfaces)
Mix ( detergent,watch )

Figure 1: Our agent extracts action sequences from natural language instructions, including action names and their arguments (between parentheses). There are two candidate sequences extracted by the instruction given by the user in the figure because the action "Wear(elbow-length rubber)" and "Wear(latex gloves)" is *exclusive*, meaning that the agent could extract only one of them.

## 2 Related Work

In this section, we review the literature on *deep learning-based* and *reinforcement learning-based* methods.

### 2.1 Deep Learning-Based Methods

Deep learning is widely used in the areas such as behavioral recognition [13], medical assistance [14] and computer version [15] by the way of building deep neural networks. Using recurrent neural networks for semantic slot filling is one of the most popular deep learning-based methods to extract action sequences. In 2015, Bastianelli et al. [2] proposed a corpus named *Human-Robot Interaction Corpus* (HuRIC) based on frame semantics [12]. In 2018, they presented a method to semantically parse natural language robotic commands from the HuRIC corpus using a multi-layer LSTM network with attention layers.

Recently, language models with an encoder-decoder structure such as Transformer [16] and BERT [17] are used in this task. In 2020, Li et al. [18] proposed a Transformer-based phrase tuple extraction model to extract action phrase tuples from natural language instructions and a Transformer-based grounding model to connect the action phrase tuples extracted by previous works to executable actions. However, the core of these *semantic framework-based* methods is designing an action framework for each action that robots can execute, resulting in restrictions on the functions of robots.

### 2.2 Reinforcement Learning-Based Methods

Reinforcement learning (RL) is the study of how an agent can interact with its environment to learn a policy that maximizes expected cumulative rewards for a task. Recently, RL for NLP has experienced great attention because of the excellent results in areas such as dialogue systems [19], neural machine translation [21] and text generation [22].

In 2009, Branavan et al. [11] presented a reinforcement learning approach for mapping natural language instructions contained in Microsoft Windows Help and Support documents to sequences of executable actions and executed them in the Windows 2000 environment. In 2020, Zhuo et al. [20] presented a novel deep reinforcement learning framework named *Federated Deep Reinforcement Learning* (FedRL) that is able to also work well when the feature space of the agent is small and the training data is limited. The FedRL framework is applied to the task presented by [9] and had a better performance.

## 3 Our Agent

Our task is to train a DQN agent based on BiLSTM and attention mechanisms for the purpose of extracting executable action sequences from natural language instructions. In this paper, word *tag* is used for taking apart of the meaning of "action" in reinforcement learning. Given a sequence of words $X = (x_1, x_2..., x_n)$ , our goal is to predict a sequence of tags $Tag = (tag_1, tag_2, ..., tag_n)$ on $X$, where $tag \in [select, eliminate]$. When predicting action names or arguments, $tag_i = select$ means that $x_i$ is predicted as an action name or action argument, and $tag_i = eliminate$ means that $x_i$ isn't predicted as an action name or action argument.

Our aim is to learn two models to extract action names and their arguments. The two models are

$$\mathcal{F}_1(Tag|X;\omega_1), X \in \mathcal{D} \tag{1}$$

$$\mathcal{F}_2(Tag|X;a;\omega_2), X \in \mathcal{D} \tag{2}$$

where $X$ is the instruction contained in our training set $\mathcal{D}$, $\omega_1$ and $\omega_2$ are parameters to be learned for predicting action names and action arguments, $a$ is an action name predicted by $\mathcal{F}_1$.

Our task can be described as a *Markov Decision Process* (MDP) and defined as the tuple $(S, Tag, T, \gamma, R)$, $S$ is the state space, $Tag$ is the action space of our intelligent agent, $T$ is a transition matrix, $\gamma$ is a discount factor used for measuring the importance of the feature and immediate rewards. The goal of our work is to find a policy $\pi : S \to Tag$ to select a tag according to the current state. We can extract the action sequences from given texts by choosing tags for the words in the given text according to the current state represented by the text. After applying the tag to the current state, a new state will be generated. This process will be executed repeatedly until each word of the given text is tagged. The core of our works is two Q-functions: $\mathcal{Q}(s, tag)$ for action names extraction task and $\mathcal{Q}(\widehat{s}, tag)$ for action arguments extraction task, we can achieve the above-mentioned process by updating these Q-functions according to the Bellman equation:

$$\mathcal{Q}_{i+1}(s_t, tag_t) = E[r + \gamma max_{tag'} \mathcal{Q}_i(s_{t+1}, tag_{t+1})], s_t \in \mathcal{S}, tag_t \in Tag \tag{3}$$

$$\mathcal{Q}_{i+1}(\widehat{s}_t, tag_t) = E[r + \gamma max_{tag'} \mathcal{Q}_i(\widehat{s}_{t+1}, tag_{t+1})], \widehat{s}_t \in \mathcal{S}, tag_t \in Tag \tag{4}$$

In this section, firstly, we introduce how to process the words that compose the text. Then, the approach of building *state* and predicting $Tag$ for each word given text $X$ is proposed. Finally, our deep Q-networks and how to predict $Tag$ for the $X$ are introduced.

### 3.1 Word Embedding

Word embeddings are widely-adopted technologies for NLP tasks by the way of representing a word by a fix-length vector. In this paper, we represent a text $X$ by a sequence of vectors $(w_1, w_2, ..., w_n)$, where $w_i \in \mathcal{R}^{50}$ is a 50-dimension real-valued vector, trained by *Word2Vec* model, representing the $i$th word in $X$.

In addition, a $\{word : index\}$ dictionary with the length $n+2$ is used for storing each different word in our datasets, where $n$ is the number of different words in our datasets. The first word in the dictionary is *PAD* and the last word in the dictionary is *UNK* which represents the word not contained in the vocabulary of the *Word2Vec* model.

## 3.2 State

In this subsection, the approach of representing the state $s$ from texts is introduced. For action names extraction, we define the state $s$ as a tuple ( *Word_ind*, *Word _pos*, *tag*) , where *Word_ind* is the index of the word according to the $\{word : index\}$ dictionary defined by previous work and *Word _pos* is the part-of-speech tagged by *StanfordPOSTagger*. For example, considering the text "Cover your nose with a respirator...", after selecting $tag_1 = Select$ on state $s_t$ and a state $s_{t+1}$ will be generated. The transition of state $s$ from time step $t$ to time step *t+1* is as shown in Figure 2. We initialize the *tag* of each word to *Null*, which means the word has not been assigned a tag by the agent.

For action arguments extraction, we use $x_a$ to denote the word which represents the action name, and we define the state as a tuple ( *Word_ind*, *Dis*, *Word_pos*, *tag*), where $Dis = (\ d_1,\ d_2, ...,\ d_n)$, where $d_i = |i - a|$ is used for representing the distance between *x_a* and other words in text.

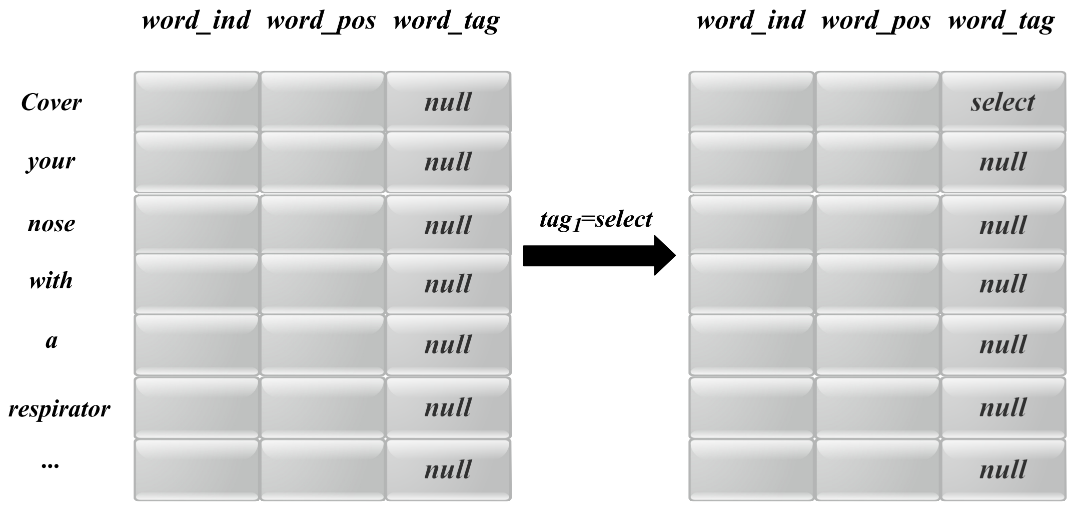


Figure 2: The transition of state from time step *t* to time step *t+1* for action name extraction

## 3.3 Our Deep Q-network

In our work, two deep neural networks based on BiLSTM and attention mechanisms $\mathcal{Q}(s_t, tag_t, \omega_1)$ and $\mathcal{Q}(\widehat{s}_t, tag_t, \omega_2)$ are used for representing the Q-functions $\mathcal{Q}(s, tag)$ and $\mathcal{Q}(\widehat{s}, tag)$ approximately.

The goal of our work is to extract action sequences from free and long texts, it requires our agent to be able to solve the long-range dependencies problem and understand the complex context of actions, but the traditional deep q-networks based on CNN or RNNs are unable to do anything about these. Thus, we use LSTM-based network to build our action names extraction model $\mathcal{F}_1$ and action arguments extraction model $\mathcal{F}_2$. Our model for action names extraction $\mathcal{F}_1$ is shown as Figure 3. Assuming that we have already known the *tag* of *x_1*, *x_2*, ..., *x_i-1*, we introduce how to predict *tag_i*, the *tag* of *x_i*. First, we use two 50-dimension real-valued vectors to represent the *tag* (tag_embedding in Figure 3) and the part-of-speech (pos_embedding in Figure 3) of $x_i$, and they are concatenated with the *word embedding* of $x_i$ as the input of forward LSTM layer and backward LSTM layer.

$$Input = Concat(word\_embedding, pos\_embedding, tag\_embedding) \tag{5}$$

Second, the outputs of the forward LSTM layer and backward LSTM layer are fed into the attention layer separately and the outputs of attention layers are concatenated as a vector.

$$\begin{aligned} \overrightarrow{\mathcal{L}} &= \overrightarrow{LSTM}(Input) \\ \overleftarrow{\mathcal{L}} &= \overleftarrow{LSTM}(Input) \end{aligned} \tag{6}$$

$$\mathcal{L} = Concat(Attention(\overrightarrow{\mathcal{L}}), Attention(\overleftarrow{\mathcal{L}})) \tag{7}$$

Where $\overrightarrow{\mathcal{L}}$ is the output of forward LSTM layer, $\overleftarrow{\mathcal{L}}$ is the output of backward LSTM layer.

Third, we apply two *Conv*1D layers to extract features from the vector. And we get the output of the network, which contains the expectation of reward value after performing every possible tag according to the current state.

$$Output = Dense(Conv(\mathcal{L})) \tag{8}$$

Finally, we use an *Argmax* function to choose $tag_{i+1}$ according to the output of the network.

$$tag_{t+1} = Argmax_{tag}(Output) \tag{9}$$

For action arguments extraction, a 50-dimension real-valued vector is used for representing the distance between action name $x_a$ and other words, and the input of our action arguments extraction model $\mathcal{F}_2$ is

$$Input = Concat(word\_embedding, dis\_embedding, pos\_embedding, tag\_embedding) \tag{10}$$

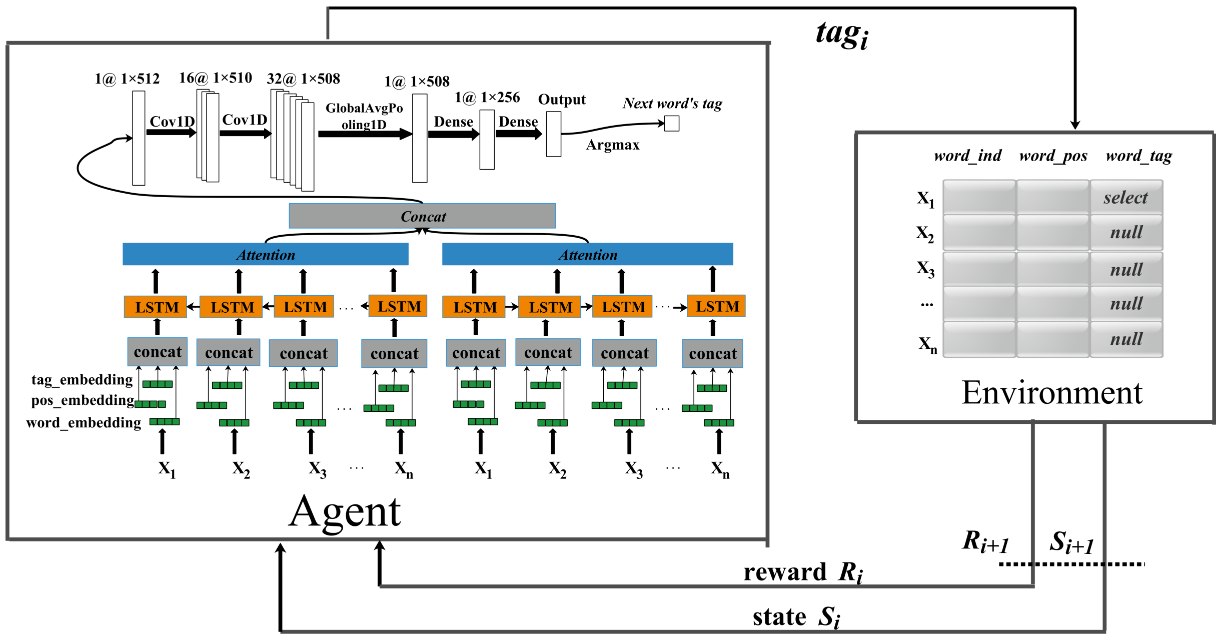


Figure 3: Our LSTM-based DQN agent for action names extraction

## 3.4 Reward

We use the same approach as [9]. The *Reward* of our agent is composed of two parts: a basic reward and an additional reward. The basic reward is given when the agent predicts a correct tag for a word in the given text, and the additional reward is given when the agent predicts a correct *tag* for the word that represents a special action type. The reward of our agent is calculated as the following equation:

$$R = \begin{cases} \alpha r_b + r_a & \text{if current word represents a special action type} \\ \alpha r_b & \text{otherwise} \end{cases} \tag{11}$$

where $r_b$ in the equation represents the basic reward and the $r_a$ in the equation represents the additional reward. The $\alpha$ in the equation is a factor that is different for different types of action.

## 3.5 Training

To learn the parameters $\omega_1$ and $\omega_2$ of the two models, we first store transitions $\langle s, tag, r, s' \rangle$ and $\langle \widehat{s}, tag, r, \widehat{s'} \rangle$ in replay memory $\Omega$ and $\widehat{\Omega}$ respectively and exploit a mini-batch strategy. where $s$ is the state representation for action names extraction model $\mathcal{F}_1$ and $\widehat{s}$ is the state representation for action arguments extraction model $\mathcal{F}_2$.

Algorithm 1 shows the process that how to train the action names extraction model $\mathcal{F}_1$. If we want to train the action arguments extraction model $\mathcal{F}_2$, we only need to replace $s$, $\Omega$ and $\omega_1$ with $\widehat{s}$, $\widehat{\Omega}$ and $\omega_2$.

We are able to build the Q-function $Q(s, tag; \omega_1)$, an expectation of reward value to be obtained after the tag is selected at state s with Algorithm 1. We update our Q-function by the Temporal-Difference (TD) algorithm during training :

$$\mathcal{Q}(s_t, tag_t, \omega_1) \leftarrow \mathcal{Q}(s_t, tag_t, \omega_1) + \alpha(r_{t+1} + \gamma \max_{tag} \mathcal{Q}(s_{t+1}, tag, \omega_1) - \mathcal{Q}(s_t, tag_t, \omega_1)) \tag{12}$$

where the right-side $\mathcal{Q}(s_t, tag_t, \omega_1)$ is the current tag-value, $\alpha$ is learning ratio, $\gamma$ is a discount factor, and $\max_{tag} \mathcal{Q}(s_{t+1}, tag, \omega_1)$ is a maximum estimated tag-value for state *s_t+1*. First, our agent predicts a sequence of $tag = (tag_1, tag_2, ..., tag_n)$ for a text by maximizing the Q-function. Then we can use this tag sequence to generate action names and use the action names to build $Q(\widehat{s}, tag; \omega_2)$ with the replay memory $\widehat{\Omega}$ and the same approach as Algorithm 1. Finally, we can use the words extracted by these models to compose the action sequences that robots can execute.

**Algorithm 1** Algorithm for extracting action names from text

```
Input: training set $\mathcal{D}$
Output: the weights $\omega_1$
  Initialize replay memory $\Omega$
  Initialize evaluate network with random weights $\omega_1$
  Initialize the target network with weights $\omega_1^- = \omega_1$
  Initialize the number of epoch to be $H$
  Initialize the number of time step of every epoch to be $N$
  Initialize the number of time step that reset the target network's weights to be $C$
  for $epoch = 1$ to H do
    for each text $X \in \Phi$ do
      Generate the initial state $s_1$ based on $X$
      for $\tau = 1$ to N do
        With probability $\varepsilon$ select a random tag $tag_\tau$
        Otherwise select $tag_\tau = argmax_{tag} Q(s_\tau, tag; \omega_1)$
        Perform $tag_\tau$ on $s_\tau$ to generate $s_{\tau+1}$ and calculate reward $r_\tau$
        Store transition $(s_\tau, tag_\tau, r_\tau, s_{\tau+1})$ in $\Omega$
        Sample a minibatch of transitions $(s_j, tag_j, r_j, s_{j+1})$ from $\Omega$
        Set
        $y_j = \begin{cases} r_j & \text{if s\_j+1 is terminal state} \\ r_j + \gamma \max_{tag'} Q(s_{j+1}, tag'; \omega_1) & \text{otherwise} \end{cases}$
        Update $\omega_1$ based on loss function $L(\omega_1) = E_{(s\_t, tag\_t, r\_t+1, s\_t+1) \in \Omega}[(y_j - \mathcal{Q}(s_t, tag_t; \omega_1))^2]$
        Every C steps reset $\omega_1^- = \omega_1$
      end for
    end for
  end for
  return The weights $\omega_1$
```

## 4 Experiments

We evaluated our agent on three datasets, i.e., "Microsoft Windows Help and Support" (WHS) documents, and two datasets labeled by [9]. The details of these datasets are presented in Table 1.

For evaluation, we first feed texts to models to predict a sequence of $Tag = (tag_1, tag_2, ..., tag_n)$. We then compare the predicted tag of each word to its corresponding annotation. We use $\#Num\_Truth$ for representing the number of words which are action names or action arguments according to the annotations in datasets, $\#Num\_Tagged$ for representing the number of words tagged as an action name or action argument by our models, and $\#Num\_Right$ for representing the number of words tagged correctly by our models. Finally, we compute metrics: $precision = \frac{\#Num\_Right}{\#Num\_Tagged}$, $recall = \frac{\#Num\_Right}{\#Num\_Truth}$, $F_1 = \frac{2 \times precision \times recall}{precision + recall}$.

Table 1: Details of our datasets

| | WHS | CT | WHG |
|---|---|---|---|
| Num. texts | 154 | 116 | 150 |
| Average length of texts | 49.98 | 209.34 | 417.17 |
| Vocab size | 791 | 2885 | 7367 |
| Action name rate(%) | 19.47 | 10.37 | 7.61 |
| Action argument rate (%) | 15.45 | 7.44 | 6.30 |

In order to evaluate the effect of the attention mechanism on our experimental results, we use LSTM networks with self-attention mechanism, feed-forward mechanism, and no attention mechanism to build our action names extraction model and action arguments extraction model. We use the following symbols to represent these approaches.

- Agent-N: An agent builds its action names extraction model and action arguments extraction model based on BiLSTM only.
- Agent-F: An agent builds its action names extraction model and action arguments extraction model based on BiLSTM and feed-forward attention mechanism [23].
- Agent-S: An agent builds its action names extraction model and action arguments extraction model based on BiLSTM and self-attention mechanism.

Then we compare these models with EASDRL, a DQN agent presented by [9] that builds its action names extraction model and action arguments extraction model based on CNN.

The input dimension for action names extraction model is set to be $(100 \times 4)$ and for action arguments extraction model is set to be $(400 \times 3)$, the batch size to be 32. The output shape of the forward-LSTM layer and backward-LSTM layer are set to be 256. And two CNN layers with kernels of size $1 \times 3$ and two fully-connected layers with the size of $512 \times 256$ and $256 \times 2$ are used. We adopt the *NAdam* optimizer [24] with learning rate 0.001 and the *ReLU* activation for all layers. The size of replay memory was set to be 50000, discount factor $\gamma$ to be 0.9.

As experimental results shown in Table 2, we can see that our agent can perform better than the EASDRL, a DQN agent based on CNN, on both action names extraction task and action arguments extraction task. We can also see that the agent with attention mechanism performs better than all tasks especially for the action arguments extraction task of the Cooking dataset and the Wikihow dataset. As the average length of texts shown in Table 1, we can know that our agent with attention mechanism can process long texts better than the agent without attention mechanism. Besides, we use the *Word_ind* to represent a word in the instructions instead of the word vector used in the state representation of [9], which reduces the cost of time in processing training data.

In order to observe the influence of model complexity on our experimental results, we vary the output shape of the LSTM layer from 128 to 512 with other parameters fixed. Results are shown in Figure 4. As shown in Figure 4 (a), we can see that the performance of the action names extraction model can be improved obviously when the output shape of the LSTM layer is increased from 128 to 256, especially for the Wikihow dataset. And it has a little improvement when the output shape of

Table 2: $F_1$ scores of different models in extracting action names and action arguments

| | Action Names | | | Action Arguments | | |
|---|---|---|---|---|---|---|
| Method | WHS | CT | WHG | WHS | CT | WHG |
| EASDRL | 93.46 | 84.18 | 75.40 | 95.07 | 74.80 | 75.02 |
| Agent-N | 97.25 | 86.31 | 76.76 | 95.47 | 79.88 | 74.67 |
| Agent-F | 97.75 | 86.44 | 79.09 | 95.55 | 80.49 | **75.95** |
| Agent-S | **98.37** | **88.98** | **81.15** | **95.59** | **81.65** | 75.77 |

the LSTM layer is increased from 256 to 512. For the action arguments extraction task, increasing the output shape of the LSTM layer has little effect on the results as shown in Figure 4 (b).

Gated recurrent units (GRU) [25] is a novel approach that is proposed to solve the problem of long-range dependencies. Compared with LSTM, GRU is able to get similar results with LSTM but it is computationally cheaper. In order to observe the performance of the GRU-based agent in our experiment, we build an agent based on BiGRU and self-attention mechanism and compared the performance with the agent based on BiLSTM and self-attention mechanism. F1 scores of the two agents are shown in Table 3.

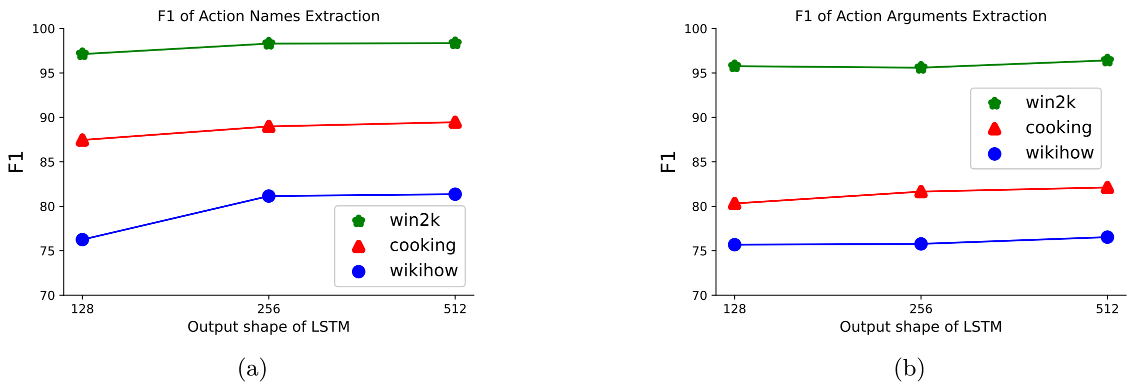
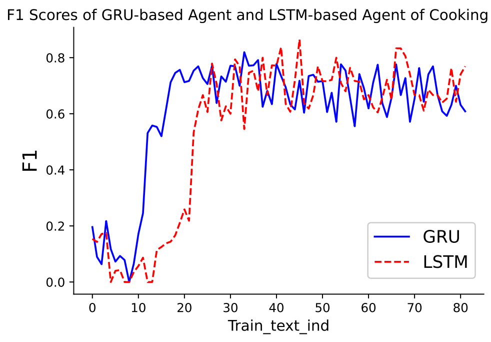


Figure 4: F1 Scores of Action Names Extraction Task and Action Arguments Extraction Task of Three Datasets

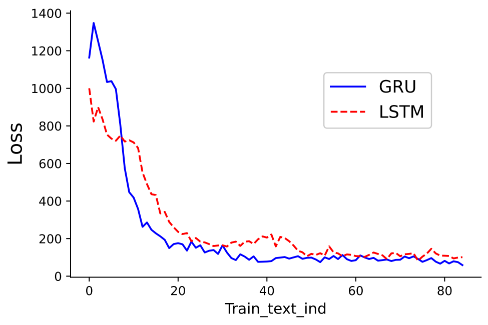
Table 3: $F_1$ scores of LSTM-based and GRU-based models in extracting action names and action arguments

| | Action Names | | | Action Arguments | | |
|---|---|---|---|---|---|---|
| Method | WHS | CT | WHG | WHS | CT | WHG |
| BiLSTM+Self-Attention | 98.37 | 88.98 | 81.15 | 95.59 | 81.65 | 75.70 |
| BiGRU+Self-Attention | 97.78 | 86.04 | 76.67 | 95.64 | 79.72 | 76.47 |

We also compare the F1 scores and loss values of extracting action names task for the Cooking dataset. The results are shown in Figure 5. The *Train_text_ind* in Figure 5 (a) and Figure 5 (b) is the number of the instructions that have been fed in the action names extraction model. According to Figure 5 and Table 3, we can see that the GRU-based models can be trained more quickly than the LSTM-based models in our experiments, but as the F1 scores of action names extraction task shown in Table 3, we can see that the LSTM-based agent is able to perform better than the GRU-based agent when the instruction is long. For the action arguments extraction task, it doesn't have an obvious difference between the LSTM-based agent and GRU-based agent.



(a) F1



(b) Loss

Figure 5: F1 Scores and Loss Values of Action Names Extraction Task of Cooking Dataset

## 5 Conclusion

Barrier-free communication with doctors is an important development trend of medical robots in the future. Thus, it is crucial for medical robots to understand the natural language instructions given by doctors and execute them correctly. In this paper, a DQN agent based on LSTM and attention mechanism is presented to extract executable action sequences for the medical service robots. An action names extraction model and an action arguments extraction model with the same structure are built, and the whole action sequences are composed of the action names and action arguments extracted by them. The experimental results show that the agent presented by this paper can perform better than the similar agents proposed by previous work. Especially because of the application of LSTM and attention mechanism, our agent has a better ability to learn the long-range dependencies. Thus, compared with the previous works, the performance of the agent presented in this paper has a significant improvement when the given instructions are long.

Pre-training language models such as ConvBERT and ELECTRA presented recently have shown a significant improvement in language understanding than traditional methods based on LSTM or RNN with attention mechanisms. So in future work, we will attempt the combination of deep reinforcement learning and pre-training language models to improve the performance of our agent.

## Acknowledgment

This work was supported by the Key Research and Development Program of Hebei Province Grant No.18210336 and Qinhuangdao City Science, Technology Research and Development Plan Grant No.202003B043 and Xinjiang Uygur Autonomous Region University Scientific Research Project (Key Natural Science Project), XJEDU2021I029. The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the paper.

## References

[1] BĂJENARU, L.; Marinescu, I. A.; Dobre, C.; DRĂGHICI, R.; Herghelegiu, A. M.; Rusu, A. (2020). Identifying the Needs of Older People for Personalized Assistive Solutions in Romanian Healthcare System. *Studies in Informatics and Control*, DOI: https://doi.org/10.24846/v29i3y202009. ISSN 1220-1766, 29(3), 363-372, 2020.

[2] Bastianelli, E.; Castellucci, G.; Croce, D.; Iocchi, L.; Basili, R.; Nardi, D. (2014). HuRIC: a Human Robot Interaction Corpus. *Proceedings of the 9th International Conference on Language Resources and Evaluation*, 4519-4526, 2014.

[3] Mensio, M.; Bastianelli, E.; Tiddi, I.; Rizzo, G. (2020). Mitigating Bias in Deep Nets with Knowl-

edge Bases: the Case of Natural Language Understanding for Robots. *AAAI Spring Symposium: Combining Machine Learning with Knowledge Engineering*, 1, 1-9, 2020.

[4] Kaelbling, L. P.; Littman, M. L.; Moore, A. W. (1995). An introduction to reinforcement learning. *The Biology and Technology of Intelligent Autonomous Agents.* Springer, Berlin, Heidelberg, 90-127, 1995.

[5] Shi, H.; Li, X.; Hwang, K. S.; Pan, W.; Xu, G. (2016). Decoupled visual servoing with fuzzyQ-learning. *IEEE Transactions on Industrial Informatics*, 14(1), 241-252, 2016.

[6] Shi, H.; Shi, L.; Xu, M.; Hwang, K. S. (2020). End-to-End Navigation Strategy With Deep Reinforcement Learning for Mobile robots. *IEEE Transactions on Industrial Informatics*, 16(4), 2393-2402, 2020.

[7] Shi, H.; Lin, Z.; Hwang, K. S.; Yang, S.; Chen, J. (2018). An adaptive strategy selection method with reinforcement learning for robotic soccer games. *IEEE Access*, 6, 8376-8386, 2018.

[8] Shi, H.; Lin, Z.,;Zhang, S.; Li, X.; Hwang, K. S. (2018). An adaptive decision-making method with fuzzy Bayesian reinforcement learning for robot soccer. *Information Sciences*, 436, 268-281, 2018.

[9] Feng, W.; Zhuo, H. H.; Kambhampati, S. (2018). Extracting action sequences from texts based on deep reinforcement learning. *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 4064-4070, 2018.

[10] Hochreiter, S.; Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8), 1735-1780, 1997.

[11] Branavan, S. R.; Chen, H.; Zettlemoyer, L. S.; Barzilay, R. (2009). Reinforcement Learning for Mapping Instructions to Actions. *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, 82-90, 2009.

[12] Fillmore, C. J. (2006). Frame semantics. *Cognitive linguistics: Basic readings*, 34, 373-400, 2006.

[13] Qin, L.; Yu, N.; Zhao, D. (2018). Applying the Convolutional Neural Network Deep Learning Technology to Behavioural Recognition in Intelligent Video. *Tehnicki vjesnik-Technical Gazette*, DOI: 10.17559/TV-20171229024444. 25(2), 528-535, 2018.

[14] Afify, H. M.; Mohammed, K. K.; Hassanien, A. E. (2020). Multi-Images Recognition of Breast Cancer Histopathological via Probabilistic Neural Network Approach. *Journal of System and Management Sciences*, 1(2), 53-68, 2020.

[15] Dumitrescu, C. M.; Dumitrache, I. (2019). Combining Deep Learning Technologies with Multi-Level Gabor Features for Facial Recognition in Biometric Automated Systems, *Studies in Informatics and Control*, DOI:https://doi.org/10.24846/v28i2y201910, 28(2), 221-230, 2019.

[16] Vaswani A; Shazeer N; Parmar N, et al. (2017). Attention is all you need. *Advances in neural information processing systems*, 5998-6008, 2017.

[17] Devlin, J.; Chang, M. W.; Lee, K.; Toutanova, K (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of the 2019 Conference of the NorthAmerican Chapter of the Association for Computational Linguistics: Human Language Technologies*, 1, 4171–4186, 2019.

[18] Li, Y.; He, J.; Zhou, X.; Zhang, Y.; Baldridge, J. (2020). Mapping Natural Language Instructions to Mobile UI Action Sequences. *arXiv preprint arXiv:2005.03776*, 2020.

[19] Azaria, A.; Srivastava, S.; Krishnamurthy, J.; Labutov, I.; Mitchell, T. M. (2020). Mitchell. An agent for learning new natural language commands. *Autonomous Agents and Multi-Agent Systems*, 34(1), 6, 2020.

[20] Zhuo, H. H.; Feng, W.; Xu, Q.; Yang, Q.; Lin, Y. (2019). Federated reinforcement learning. *arXiv preprint arXiv:1901.08277*, 2019.

[21] Kumar, G.; Foster, G.; Cherry, C.; Krikun, M. (2019). Reinforcement Learning based Curriculum Optimization for Neural Machine Translation. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 1, 2054-2061, 2019.

[22] Shi, Z.; Chen, X.; Qiu, X.; Huang, X. (2018). Toward diverse text generation with inverse reinforcement learning. *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 4361-4367, 2018.

[23] Raffel, C.; Ellis, D. P. (2016). Feed-forward networks with attention can solve some long-term memory problems. *International Conference on Learning Representations*, 2016.

[24] Dozat, T. (2016). Incorporating Nesterov Momentum into Adam. *International Conference on Learning Representations*, 1, 2013–2016, 2016.

[25] Chung, J.; Gulcehre, C.; Cho, K.; Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR abs/1412.3555*, 2014.

Copyright ©2021 by the authors. Licensee Agora University, Oradea, Romania.
This is an open access article distributed under the terms and conditions of the Creative Commons Attribution-NonCommercial 4.0 International License.
Journal's webpage: http://univagora.ro/jour/index.php/ijccc/

C O P E
Member since 2012
JM08090

This journal is a member of, and subscribes to the principles of,
the Committee on Publication Ethics (COPE).
https://publicationethics.org/members/international-journal-computers-communications-and-control

*Cite this paper as:*
Andonie, R.; Dzitac, I. (2021). How to Write a Good Paper in Computer Science and How Will It Be Measured by ISI Web of Science, *International Journal of Computers Communications & Control*, 15(5), 4115, 2021.
https://doi.org/10.15837/ijccc.2021.2.4115