IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 3 June 2022, accepted 23 June 2022, date of publication 27 June 2022, date of current version 1 July 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3186760

RESEARCH ARTICLE

Building a Natural Language Query and Control Interface for IoT Platforms

ZHIPENG XU[1], HAO WU[1], XU CHEN[1], YONGMEI WANG[1], AND ZHENYU YUE[1,2]
[1]School of Information and Computer Science, Anhui Agricultural University, Hefei 230036, China
[2]Anhui Provincial Engineering Laboratory for Beidou Precision Agriculture Information, Hefei 230036, China

Corresponding author: Yongmei Wang (plainfebruary@gmail.com)

This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 62102004.

ABSTRACT The rapid development and popularity of IoT technology has reshaped the way people interact with the real world. Many researchers have attempted to build natural language interfaces for IoT platforms, but have not produced much progress in parsing natural language commands that contain multiple operations and more complex logical structures. In this paper, we propose IoT-NLI, a natural language query and control interface for popular IoT platforms, which uses hierarchical semantic parsing algorithms and directed edge-tagged graph structures to efficiently parse natural language commands input by users, enabling them to perform multiple operations contained in one complex natural language command. Experiments in three domains, agriculture, industry, and smart home, show that IoT-NLI has excellent performance and reasonable response time. Finally, a IoT-NLI application was developed on the Android platform and integrated with the AliCloud platform. It enables users to query and control devices on Android phones through chat windows similar to instant messaging software.

INDEX TERMS Natural language interface, human-computer interaction, natural language processing, IoT platform.

I. INTRODUCTION

Benefiting from the rapid development of smart hardware and mobile computing as well as the improvement of people's living standards, Internet of Things (IoT) technology has been applied in a large number of fields such as smart home, medical, automotive, and agriculture [1]. Real-time information exchange between human and computer through the Graphical User Interface (GUI) provided by the IoT platform is a common human-computer interaction method of the current IoT platform [2]. GUIs have contributed greatly to the popularity of computers, but with the increasing number of IoT devices, the design of GUI has become more and more complex, which increases the learning and usage costs for users. The development of natural language processing technology makes natural language interface [3] a new solution for the interaction method of IoT platform, which means that users can use natural and direct way for human-computer interaction. The outstanding performance of virtual assistants in mobile applications, represented by Apple Siri and Google Assistant [4], also proves the user-friendliness and efficiency of natural language interfaces.

The task of building natural language interfaces for IoT platforms has received the attention of many researchers due to its practical relevance in real-life situations. The fundamental task of the interface is to analyze the meaning of the commands in order to identify the user's goals and send them in a certain form to the device and lead to a series of operations by the controlled device [5]. Keyword-based interaction approaches are the most popular among related studies: Austerjost *et al.* [6] constructed an intelligent virtual assistant for a chemical laboratory that looks for key phrases in the command text that match a pre-constructed lexicon of device actions to trigger specific intentions leading to the device's actions. Baby *et al.* [7] performed word segmentation as well as stop word filtering on natural language command text and then matched the words in the command with words in the device dictionary and action dictionary to determine the control target. Mahnoosh *et al.* [8] constructed a natural language interface for a smart home scenario,

The associate editor coordinating the review of this manuscript and approving it for publication was Maurizio Tucci.

which used NLP tools to do semantic annotation of natural language commands to extract elements such as location, device, and action from them, which is more goal-oriented than the above approaches, but still focuses only on the meanings expressed by individual words in the commands and ignores the logical relationships contained in the natural language command statements. Several researchers have explored grammar-based approaches: Noura *et al.* [9] proposed the VISH grammar, which uses multiple rules to extract multiple types of entities from command for analyzing the intent in natural language commands and controlling target devices in smart home scenarios. Wu *et al.* [10] proposed a parsing algorithm based on dependent syntactic analysis, which uses a dependent syntactic tree and defined natural language semantic parsing rules to semantically extract action-location-device triplets, making the triads corresponding to each operation logically independent, which solves the problem of parsing commands containing multiple intents to some extent. However, dependent syntactic analysis tools are very sensitive to the length of the command sequence, making it difficult to analyze complex commands with long lengths effectively. In recent years, several researchers have experimented using deep learning-based approaches: Gui *et al.* [11] first classify natural language commands for intent, then convert natural language commands into logical form using a sequence-to-sequence model [12], and finally further compile them into the form of JavaScript to send requests to sensors and return responses. Park *et al.* [13] used a named entity recognition method to annotate the words in the commands so as to extract one logical form in the form of Key-Value in each natural language command. Ye *et al.* [14] vectorize the instruction text, compare it with the commands in the command dictionary for similarity, and perform the operation corresponding to the instruction with the highest similarity to it. These approaches above have made effective attempts to build natural language interfaces for IoT platforms, but most of the related work can only parse natural language commands that contain a single operation. A common example of a real-world scenario is that when a user wants to start a device and then adjust its parameters, two separate commands are needed to perform two operations (e.g., if the users wants to start the air conditioner first and then adjust the temperature, they need to start the device by entering "请先帮我打开客厅里的空调的开关" ("Please turn on the air conditioner in the living room for me first"), wait for the command to be processed, and then enter "再将温度设置到24度"("set the temperature to 24 degrees") to adjust the parameters). Such a process does not have the coherence and ease of use of natural human-computer interaction. Amazon's internal user command data shows that 53% of natural language commands contain multiple actions [15], indicating that users prefer to use commands with multiple operations (e.g., "请先帮我打开客厅里的空调的开关, 再将温度设置到24度"("Please turn on the air conditioner in my living room and set the temperature to 24 degrees")) when performing more complex operations. In this paper, we consider the command parsing problem as a construction process of a directed acyclic graph structure and use the command structure features to obtain the intent of each operation.

We have drawn some inspiration from the design of natural language interfaces for relational databases [16], code generation tasks [17], knowledge bases [18], and web navigation [19]. Xu *et al.* [20] proposed SQLNet, which uses a Seq2set model and a dependency graph-based column attention mechanism for SQL statement synthesis, effectively solving the order-matters problem of the traditional Seq2Seq model on related tasks. Su *et al.* [21] proposed a modular sequence-to-sequence model for building an interactive natural language interface that uses fine-grained user interactions to enhance user experience and interface usability. Mazumder *et al.* [22] proposed a continuous learning framework for knowledge graph construction and updating by extracting knowledge triples from conversational information with users. In addition, some studies on entity extraction, relationship extraction, and joint extraction have also inspired us in semantic parsing methods. Wei *et al.* [24] introduced a new entry point for the triplet extraction task by proposing the use of a cascaded binary tagger for entity tagging, which elegantly solves the common but intractable overlap problem in relation extraction. Jie *et al.* [25] proposed an effective way to incorporate dependent syntactic information into the BiLSTM-CRF model [26] for the named entity recognition task on which excellent results were achieved.

In conclusion, building natural language interfaces for IoT platforms has a positive effect on improving the human-computer interaction experience in IoT scenarios. However, the existing natural language interfaces are weak in parsing commands that contain multiple operations and complex logic, making the interaction process incoherent. The contributions of this paper are as follows:

- In this paper, we propose a hierarchical semantic parsing framework for efficient structured parsing of Chinese natural language commands in the form of directed edge-tagged graphs. It is able to overcome the drawback that traditional natural language interface construction methods for IoT platforms can only parse natural language commands that contain a single operation.
- Experimental results on the publicly available HCIC dataset show that our approach can effectively parse commands containing multiple operations, making user interactions more natural and coherent. And it has good domain adaptability in common IoT scenarios such as industrial, agricultural and smart home.
- We developed a user-conversational natural language interaction application on Android platform in the form of popular instant messengers such as WeChat, WhatsApp, and LINE. We integrate it with AliCloud IoT platform and apply the proposed approach to real scenarios.

https://iot.cn-shanghai.platform.com/?Action=QueryDevicePropertyData
&Iotid=$Iotid&Identifier=$Identifier
&PageSize=10&Asc=1
(a) Query API template

https://iot.cn-shanghai.platform.com/?Action=SetDeviceProperty
&Iotid=$Iotid&Items={"$Identifier":$TargetValue}
(b) Control API template

FIGURE 1. Query and control API.

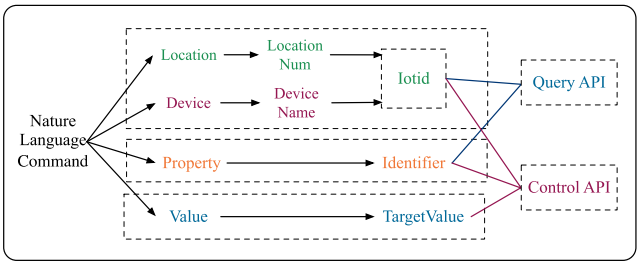

FIGURE 2. Parameter dependencies.

II. PRELIMINARY

A. INTERFACE DESIGN REQUIREMENTS

Data collection, data query, device control and parameter adjustment are the main functions of the GUI interface of the IoT platform. In the actual application scenario, the end-user makes use of the GUI interface provided by the IoT platform to query device data and control operations by clicking on the page controls. In terms of technical principle, this approach is to obtain key information through user interaction with the GUI interface to determine the API to be called, and then send HTTPS/HTTP GET or POST requests to the API server address, and the IoT platform performs operations such as device manipulation or data query based on the processing of the request. Our IoT-NLI has the same design idea as GUI, but we hope to provide a more user-friendly interaction for users: the interface can parse the natural language commands spoken by users, obtain key information and confirm the intent of the commands, select the appropriate API, auto-populate the key information in it, and then call the API to complete the operation in the same way. In this paper, we take the data query and device control APIs in the AliCloud IoT platform API suite as an example. It should be noted that our core technology can also be used in other popular IoT platforms such as Tencent Cloud and Things Cloud, but we only focus on the Ali Cloud IoT platform in this work, and users can migrate according to their needs. Figure 1 shows the form of query and control API involved in this paper, the variable name after '$' indicates that it is the parameter that needs to be populated, Action represents the operation used by this API, PageSize as well as Asc are the variables that do not need to be concerned in this paper. iotid is the device in the IoT platform Identifer means the identifier corresponding to the attribute name, and in the control API, the identifier also needs to get its corresponding target value TagetValue to set the value of the property. The approach of extracting entities from natural language commands loses a large amount of semantic information, and when multiple entities of the same type appear in one command, order problems among the entities lead to confusion in semantic relations. For this reason, a structured logical form for natural language commands is needed, which constrains the semantic relations between entities while extracting the entities in the commands. We define specific structured logical forms for IoT-NLI tasks, consisting of four parts: location, device, property, and value, and each structured logical form represents an action that can be executed. Given the natural language command "请帮我打开大棚里的恒温器的电源,再查看温室里水箱的温度"("Please help me turn on the power to the thermostat in the shed and check the temperature of the water tank in the greenhouse"), the corresponding logical form is 'location': '大棚' ('shed'), 'device': '恒温器' ('thermostat'), 'property': '电源' ('power'), 'value': '打开' ('turn on') and 'location': '温室' ('greenhouse'), 'device': '水箱' ('water tank'), 'property': '温度' ('temperature'), 'value': '查看' ('check'). The necessary contextual information in each structured logical form, can be populated into the API according to the parameter dependencies shown in Figure 2. Parameter dependencies can be obtained from the IoT platform through requests. The script for obtaining parameter dependencies from the AliCloud platform and the related request code can be found at https://github.com/CodaChan/IoT-NLI. Depending on the length of the command, the expression of the intent, and the number of operations contained in the command (i.e., the number of structured logical forms), commands are classified into basic natural language commands and complex natural language commands, which are defined as shown in Table 1.

B. SYNTACTIC DEPENDENCY

Dependency parsing is one of the key techniques in natural language processing. The basic process is to determine whether the composition of the input sentence is grammatically correct or not, and to analyze the syntactic structure of the sentence to construct a syntactic dependency tree. IoT-NLI addresses the semantic understanding of basic natural language commands by parsing the syntactic dependencies in the commands. To better explain the semantic parsing algorithm based on dependency parsing that we will propose in Section 3.3, this subsection introduces syntactic dependency relations and dependency parsing, and demonstrates the process of dependency parsing. In DuCTB [27], a Chinese syntactic dependency tree library constructed by Baidu, the dependencies in the syntactic dependency tree are defined as 14 annotation relations. Examples of several common syntactic dependency relations HED, VOB, ATT, and COO used in the semantic parsing algorithm are given in Table 2.

Dependency parsing of natural language commands will be performed using the Chinese dependency parser tool DDParser [28]. Taking the natural language command "请帮我查看客厅里空调的温度" ("Please help me check the temperature of the air conditioner in the living

TABLE 1. Classification of natural language command.

Type	Description	Example
Basic natural language commands	Sequence length less than or equal to 21 characters. &Target value is verb &The number of structured logical forms contained is less than 3.	打开客厅和厨房的灯的电源。 （Please turn on the lights in the living room and kitchen）
Complex natural language commands	Sequence length greater than 21 characters. Or The number of structured logical forms contained is greater than or equal to 3.	将大棚和温室里的灯光颜色变为红色，再查看一下实验田里的土壤温度。 (Change the color of the lights in the shed and greenhouse to red, and check the soil temperature in the experimental field)

TABLE 2. Syntactic dependencies.

Type	Description	Example
HED	The core of the entire sentence	打开温室里的空调的电源（ROOT，打开，HED） (Turn on the power of the air conditioner in the greenhouse)
VOB	The relationship between object and predicate	查看大棚里的土壤的温度（查看，温度，VOB） (Check the temperature of the soil in the greenhouse)
ATT	The relationship between attributive and headword	查看客厅里的主灯的颜色（主灯，颜色，ATT） (Check the color of the main light in the living room)
COO	The relationship between words of the same type	打开客厅和书房的空调的电源（客厅，书房，COO） (Turn on the power of the air conditioners in the study and living room)

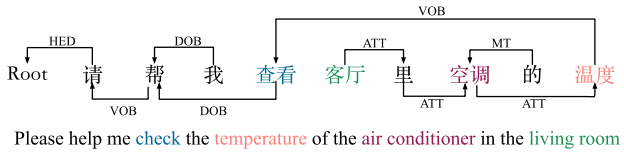

FIGURE 3. Syntactic dependency graph.

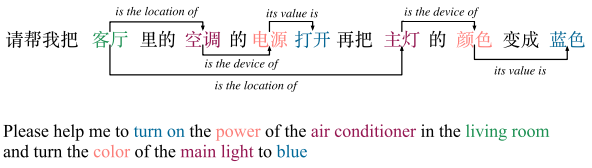

FIGURE 4. Semantic relations between the entities.

room'') as an example, DDParser will return the syntactic dependency tree and lexical annotation results in the form of a list. The syntactic dependency tree constructed by DDParser is shown in Figure 3. In this example, the noun "温度" ("temperature") corresponds to the verb "查看" ("check") in the command, and the corresponding relationship is a verb-object relationship (VOB), which means that the noun "温度" is the object of the verb "查看". The noun "空调" ("air conditioner") corresponds to the word "温度", and the corresponding relationship is ATT, which means that the noun "空调" is used as a gerund to modify "温度". The noun "客厅" ("living room") is in ATT relationship with the preposition "里" ("inside"), and the preposition "里" is also in ATT relationship with the noun "空调", so it can be assumed that the noun "客厅" modifies the noun "空调" through the preposition "里".

C. SEMANTIC RELATIONS IN COMMANDS

Semantic relations describe the semantic connections between the components of a sentence [29]. For a specific task, the correct semantic relations can be obtained by parsing the dependencies and formulating extraction rules. The original corpus of the dataset in this paper contains natural language query and control commands in three IoT application scenarios: agriculture, industry, and smart home. By summarizing the semantic relations existing in natural language query and control commands, this paper defines three semantic relations between entities in structured logical forms: the inclusion relation between location and device ("is the location of"), the inclusion relation between device and property ("is the device of"), and the key-value relation between property and property value ("its value is"). It is worth noting that the three semantic relations defined here denote semantic logical connections between entities. Specifically, if the entities e_a and e_b form a semantic relation r, then e_a and e_b belong to the same structured logical form. Given the natural language command "请帮我把客厅里的空调的电源打开, 再把主灯的颜色变成蓝色"("Please help me turn on the power of the air conditioner in the living room and turn the color of the main light to blue"), the semantic relations

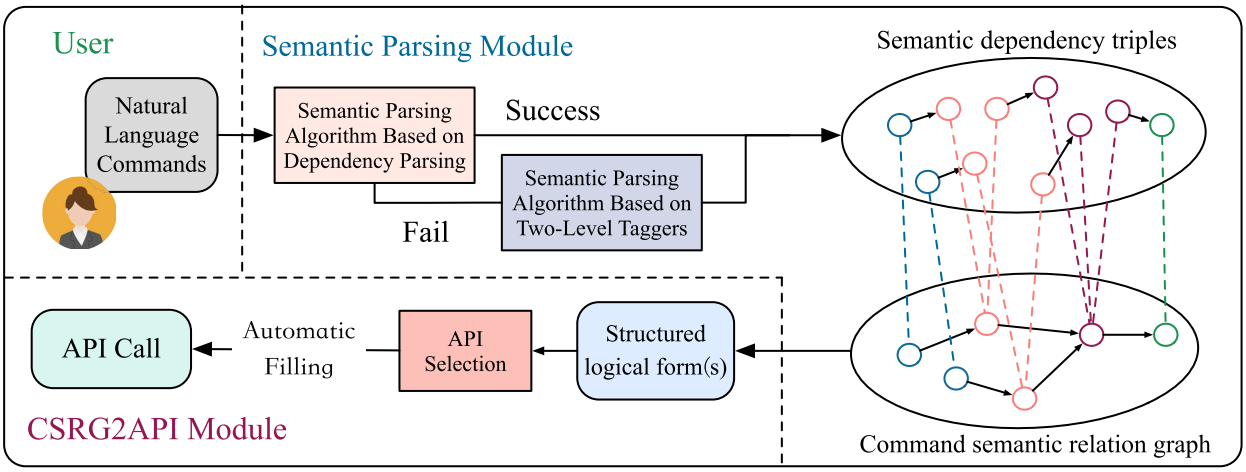

FIGURE 5. IoT-NLI equivalent abstraction workflow.

between the entities is shown in Figure 4. For the above example, the entity "living room" is in "is the location of" relationship with "空调" ("air conditioner"), and the entity "客厅" ("living room") is also in "is the location of" relationship with "主灯" ("main light"), which means that "空调" and "主灯" are both devices in the "客厅". The entity "空调" and "电源" ("power") form "is the device of" relationship, and the entity "主灯" and "颜色" ("color") form "is the device of" relationship, which means that in this command, "电源" is the property of "空调", "颜色" is the property of "主灯", the two are independent of each other. Constraints via semantic relationships avoid semantic confusion, and such constraints also exist in key-value relationships between attributes and attribute values. In this paper, the semantic relations in the commands are represented using a triple of the form (h, r, t), where h is the head entity, r is the relationship between entities, and $t$is the tail entity.

III. METHODOLOGY

A. IoT-NLI EQUIVALENT ABSTRACTION WORKFLOW

IoT-NLI is a natural language interface for IoT platform query and control APIs based on hierarchical semantic parsing algorithms, and its equivalent abstraction workflow is given in this subsection. Its equivalent abstraction workflow is shown in Figure 5.

The user takes the natural language command as input, and the semantic parsing module of IoT-NLI first performs a preliminary semantic parsing of the command using a semantic parsing algorithm based on dependency parsing. If the parsing is successful, the set of semantic relation triples of the natural language command is generated. Commands that fail will be parsed using semantic parsing algorithm based on two-level taggers to achieve the same goal. The set of semantic relation triples will be mapped into the command semantic relation graph (CSRG) by defining mapping rules. Finally, the CSRG2API module generates structured logical forms from CSRG and determines their intent, and then completes the automatic filling of the API according to the parameter dependencies.

B. INTERMEDIATE LANGUAGE FRAMEWORK

The use of graph structures for structured representation of text has many advantages over relational models and a range of NoSQL alternatives [29]: graph-type text structures provide concise forms for text and reduce the noise interference in discrete natural text. Complex and continuous relationships between entities are captured through edges and paths in graphs [30] and allow data to be stored in a more flexible way [31]. Inspired by frame semantics [32] and related work on knowledge graphs [29], we propose a graph-structured command intermediate language framework and call it command semantic relation graph (CSRG) for structural parsing of commands using semantic triple extraction. CSRG is the form of directed edge-labelled graph, which takes entities in commands as nodes and makes directed links through semantic relations between entities to store necessary information in natural language commands. The process of forming CSRG from the semantic relation triad is shown in Figure 6, where the blue, purple and brown directed edges represent the three semantic relations "is the location of", "is the device of" and "its value is" respectively. This process is similar to the construction of knowledge graph. The same entities in different triples will be mapped to the same node in the CSRG and the semantic relationships on the edges will be preserved. It is important to note that in this process, the position number of the entity in the command is stored in the CSRG to prevent ambiguity.

C. SEMANTIC PARSING ALGORITHM BASED ON DEPENDENCY PARSING

This subsection proposes a low-consumption semantic parsing algorithm to accomplish the semantic parsing of basic

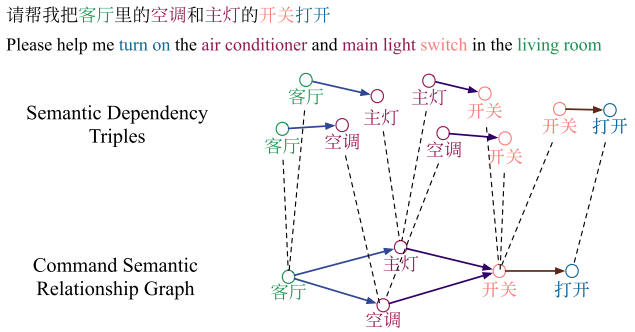

FIGURE 6. Construction of the command semantic relationship graph.

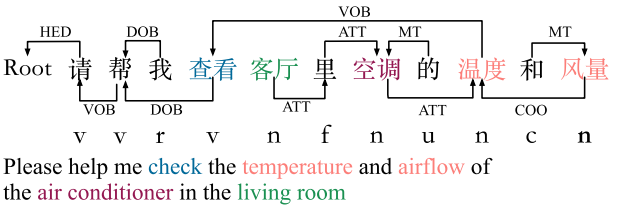

FIGURE 7. Dependency parsing results for the natural language command " Please help me check the temperature and airflow of the air conditioner in my living room ".

natural language commands based on syntactic dependencies. The algorithm first uses DDParser to perform dependency parsing of natural language commands to obtain the syntactic dependency tree of the commands, and then converts the syntactic dependencies to semantic relations by defining multiple rules. In this paper, we propose the following rules for this process, each of which can be used multiple times during the parsing process in a single natural language command.

Rule 1: When a verb that forms verb-object relationship (VOB) with a noun exists in the command, the verb indicates the action that will be performed on the noun. The noun can be considered as an entity of "property" type, and the verb as the corresponding value. The VOB relationship can be used to extract the "its value is" relation triple.

Rule 2: Words in COO relationship with the same noun w_1 can be considered to belong to the same set of entities $W = \{w_1, w_2 \ldots w_n\}$, and the words in the set W share the same semantic relationship with w_1.

Rule 3: The nouns that form ATT relationship with the entities of "property" type extracted by Rule 1 can be extracted as entities of "device" type, and the "is the device of" relationship triple is obtained.

Rule 4: The noun that forms ATT relationship with an entity of the "device" type can be extracted as an entity of the "location" type, and the relationship triple "is the location of" is obtained.

Rule 5: ATT relationship can be passed through prepositions. Let the noun w_a form an ATT relationship with the preposition w_{f1}. If there exists a noun w_b that forms ATT relationship with w_{f1}, w_a still modifies w_b, which can be used for rules 3 and 4. The process can be repeated if the word that forms an ATT relation with w_{f1} is also a preposition.

Based on the above rules, we take the natural language command "请帮我查看客厅里空调的温度和风量" ("Please help me check the temperature and airflow of the air conditioner in my living room") as an example, and parse it as follows. Firstly, we use DDParser to obtain the syntactic dependency graph and lexical annotation results. As in this example, the syntactic dependencies are shown in Figure 7. The second step is to find the nouns that form verb-object relations (VOB) with other words according to the parsing results and, with the help of rule 1, to extract the entity of the "property" type and the corresponding value in the natural language command. At the same time, according to rule 2, the nouns that form COO relations with entity of "property" type belong to the same set of entities and share the same syntactic relations. In this example, we first find the word "温度" ("temperature"), which forms VOB relationship with the verb "查看" ("check"), and we can get the semantic relationship triple ("温度", its value is, "查看") according to rule 1. Meanwhile, the entity "温度" and the noun "风量" ("airflow") form COO relationship, and according to rule 2, we can get the triplet ("风量", its value is, "查看").

Next, the command will be extracted by rule 3: in this example, the set of entities of the "property" type of the command has been found according to rule 1 and rule 2, and the noun "空调" ("air conditioner") forms ATT relationship with the entity "温度" of the "property" type, so the entity "空调" of the "device"n type can be extracted. In addition, according to rule 2, the entity "空调" modifies both "温度" and "风量". Then we can get the triple ("空调", is the device of, "温度") and ("空调", is the device of, "风量").

Finally, according to rule 4 and rule 5, the noun "卧室" ("living room") modifies the entity "空调" of device type, so we can extract "卧室" as the entity of "location" type and get the triple ("卧室", is the location of, "卧室"). With semantic parsing algorithm based on dependency parsing, natural language commands are parsed into semantic relations triples, which are then stored in the CSRG intermediate language framework.

D. SEMANTIC PARSING ALGORITHM BASED ON TWO-LEVEL TAGGERS

If the semantic parsing algorithm based on dependency parsing fails (when the set of any relation triples is empty and the existing relations cannot be mapped to a complete CSRG), the command is considered as a complex natural language command, and IoT-NLI uses semantic parsing algorithm based on two-level taggers to complete the parsing of complex natural language commands. This method consumes more resources but is more flexible in parsing natural language commands and has stronger generalization capability. Common joint extraction models are not suitable for the task of IoT-NLI due to the overlapping problem of entities and the uncertainty of the number of relationships. Therefore, we use a tagger-based

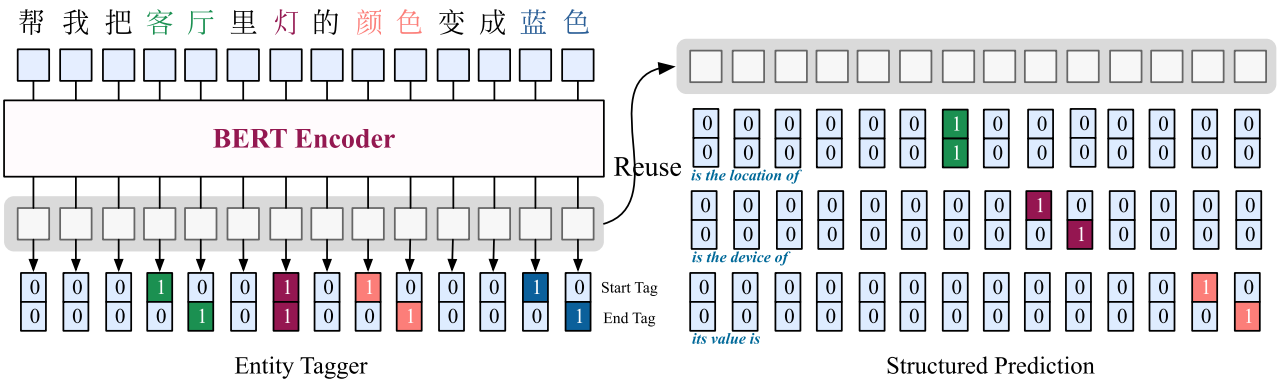

FIGURE 8. The structure of semantic parsing algorithm based on two-level taggers.

prediction model to accomplish the extraction of semantic relationship triples. Its structure is shown in Figure 8.

In the example in Figure 8, the entity tagger extracts "客厅" ("living room"), "灯" ("lamp"), "颜色" ("color"), and "蓝色" ("blue") from the natural language commands, and then performs structured prediction of each relationship for each entity to generate semantic relations triples. For example, when predicting the entity "客厅" using the tagger of the "is the location of" relation, the entity "灯" of the "device" type is actually predicted based on the entity "客厅" and the relation "is the location of". The semantic relation triple ("客厅", is the location of, "灯") is obtained by the above process. We will describe the various parts of this process in detail.

1) BERT ENCODER

BERT[33] learns linguistic representation by masking or replacing random words during training and making predictions through context. We use BERT as encoder to encode natural language commands. We denote the Transformer Block as $Trm(x)$, and the operations of BERT at each layer can be represented as

$$h_0 = \text{Trm}\left(WO_c + E_s + E_p\right) \tag{1}$$
$$h_n = \text{Trm}\left(h_{n-1}\right) \tag{2}$$

where h_n represents the hidden state vector of n-th layer. We use $BERT(x)$ to represent the BERT model. The operation process can be expressed as

$$h = BERT\left(E_t + E_s + E_p\right) \tag{3}$$

where $E_(t)$ is token embedding, E_s is segmentation embedding, and E_p is position embedding. O_c and W represents the one-hot embedding of the input words and the BERT stored.

2) ENTITY TAGGER

We use two identical binary taggers to extract all entities in natural language commands, as shown in Figure 8. When the start tagger or end tagger marks the current character as 1, we call the current character with a "start tag" or "end tag", indicating that the current character is the start or end position of an entity, thus completing the extraction of the entity. The process of using tagger for prediction is as follows.

$$p_i^{start} = \text{sigmoid}\left(W_{start} v_i + b_{\text{start}}\right) \tag{4}$$
$$p_i^{end} = \text{sigmoid}\left(W_{\text{end}}\, v_i + b_{\text{end}}\right) \tag{5}$$

where p_i^{start}, p_i^{end} represent the probability of the i-th character as the start character or end character of the entity in the input natural language command sequence, and v_i is the vector representation of the i-th character in the command sequence after encoding by BERT. W_{start}, W_{end} represent trainable weights, respectively, and b_{start}, b_end represent bias. When p^{start} or p^{end} is greater than the threshold value, taggers mark the start tag or end tag as 1. Here we set the threshold value to 0.6.

The minimum character span principle is used for multiple entity detection, where characters with start tag are paired towards the nearest character with end tag. All end tags before the first start tag are ignored to ensure the integrity of each entity. Let's take the example shown in Figure 8: the starting character "颜" is predicted by the start tagger, and we extract the entity "颜色" ("color") by taking only the ending character "色" with the smallest character span (character span of 1) as the ending character of the entity, while the forward ending character "灯" and the ending character "色" with a span of 4 will not be paired with the current starting character

3) STRUCTURED PREDICTION

A structure similar to the entity tagger is used to structurally predict the semantic relations triples using the header entities already extracted by the entity tagger. In this process, each of the three relationships defined in this paper will be assigned a pair of taggers to locate the tail entity that forms the specific relationship with the head entity. The prediction process for the start tag and the end tag of the tail entity is as follows. The semantic parsing algorithm based on two-level taggers is not affected by the overlap problem because of the unique

structure of tagger.

$$p_i^{start'} = \text{sigmoid}\left(W_{start'}\left(v_i + v_w^k\right) + b_{start'}\right) \quad (6)$$

$$p_i^{end'} = \text{sigmoid}\left(W_{end'}\left(v_i + v_w^k\right) + b_{end'}\right) \quad (7)$$

where v_w^k represents the vector representation of the k-th entity captured by the entity tagger. When the entity consists of multiple characters, v_w^k is the average vector of each character contained in the k-th entity. The entity tagger and structured prediction are trained as a whole.

E. CSRG2API MODULE

After the parsing of the above two algorithms, natural language commands can be stored in the form of CSRG. However, suitable algorithms are still required to generate structured logical forms by the intermediate language framework and to convert the necessary parameters into suitable APIs based on parameter dependencies. For the first step, based on the directedness and edge labeling of the CSRG, we perform a depth-first search of the CSRG starting from the Location node to obtain several structured logical forms in the graph. We describe this process with the CSRG in Figure 6 as an example: there is only one location node ''living room'' in this graph, and the search starts from this node based on the directed relationship of edges. The search results are ''客厅'' (''living room''), ''主灯'' (''main light''), ''开关'' (''switch''), ''打开'' (''turn on'') and ''客厅'' (''living room''), ''空调'' (''air conditioner''), ''开关'' (''switch''), ''打开'' (''turn on''). This is consistent with the composition of structured logical forms. Also, since the search depth of a node corresponds to the node type, the node type (entity type) can be obtained by the search depth. By the above process, the following structured logical form can be obtained ''’location’: ''客厅'', ’device’: ''主灯'', ’property’: ''开关'', ’value’: ''打开'' and ’location’: ''客厅'', ’device’: ''空调'', ’property’: ''开关'', ’value’: ''打开''.'' The next step is to select the appropriate API for each structured logical form, which is actually the process of classifying the intent of the structured logical form. In the process of parsing commands, we find that the intent information in structured logical forms is usually reflected in the ''value'' parameters. In query commands, the ''value'' is often the verb associated with the query action, such as ''查看'' (''view''), ''查询'' (''query''), ''告诉'' (''tell''), etc. In control commands, the ''value'' parameter is usually the value that will be set for the property, such as ''打开'' (''open''), ''蓝色'' (''blue''), etc. Based on this feature, we propose a special method to select the appropriate API for structured logic forms. Given a structured logical form, we first obtain the ''value'' parameter and a pre-constructed list of query verbs, and use Word2Vec to embed both the ''value'' parameter and the verbs in the list. Then we calculate the cosine similarity between the ''value'' parameter and each verb in the query verb list in turn. For word vectors x_i and y_i, the cosine similarity is calculated as:

$$\cos < x_i, y_i >= \frac{\sum_{i=1}^{n}(x_i \times y_i)}{\sqrt{\sum_{i=1}^{n} x_i^2 \times \sum_{i=1}^{n} y_i^2}} \quad (8)$$

The cosine similarity can calculate the similarity between words, and its value range is [0,1], the closer the result is to 1, the more similar the semantics of the two words are. Here we set the threshold to 0.68, when the cosine similarity between the ''value'' parameter in the structured logical form and any verb in the verb list is greater than 0.68, we consider the user's intent of the structured logical form to be a query, and vice versa for control.

IV. EXPERIMENT

A. HCIC DATASET

To train and test IoT-NLI, we built HCIC (Human-computer interaction, Chinese), a Chinese dataset for human-computer interaction on IoT platforms, containing a total of 12,812 natural language commands and their corresponding structured logical forms, via crowdsourcing. The dataset can be found at https://github.com/CodaChan/HCIC. In the crowdsourcing process, we only give structured logical forms and employ crowdsourcing workers to expand the structured logical forms into spoken natural language commands. The crowdsourcing workflow is shown in Table 3. Since the same structured logical form can be expressed using many different ways of natural language commands, each structured logical form will be given to several different crowdsourcing workers to increase the structural diversity of natural language commands.

The HCIC dataset contains 12,812 data from three application areas: agriculture, industry, and smart home, each of which includes basic natural language commands as well as complex natural language commands, and the distribution of the data is shown in the Table 4. Basic natural language commands are used to evaluate the metrics of the semantic parsing algorithm based on dependency parsing. Complex natural language commands will be used for semantic parsing algorithm based on two-level taggers training and evaluation.

1) EVALUATION METRICS FOR SEMANTIC PARSING ALGORITHMS

IoT-NLI uses two semantic parsing algorithms and the CSRG intermediate language framework to parse natural language commands and finally generate a unique structured logical form. We use Accuracy, Precision, Recall and F-score, to evaluate the semantic parsing algorithm. Let N represent the number of all samples tested and N_ex represent the number of samples with exactly correct parsing results. Accuracy is defined as

$$\text{Accuracy} = \frac{N_{ex}}{N} \quad (9)$$

Precision represents the ratio of the number of correct structured logical forms N_ts obtained by the corresponding algorithm to the number of all generated structured logical forms

TABLE 3. Crowdsourcing workflow.

Structured Logical Form	Natural Language Command
{ ‘location’: “大棚” (“shed”), ‘device’: “恒温器” (“ thermostat”), ‘property’: “电源” (“power”), ‘value’: “启动” (“start”) }	请帮我启动大棚里恒温器的电源。 Please help me to start the power to the thermostat in the shed. 太阳快要落山了，将大棚里恒温器的电源启动。 The sun is about to set, powering up the thermostat in the shed. 启动大棚里恒温器的电源，看看是否正常工作。 Start the power to the thermostat in the shed and see if it works properly.

TABLE 4. HCIC dataset distribution and sample.

Field	Total	Type and number	Sample of command
Agriculture	4262	Basic command (821)	查看大棚里土壤的湿度。 (Check the moisture content of the soil in the greenhouse)
		Complex command (3441)	土壤湿度太低，请帮我将大棚里水阀的流量调整到最大。 (The soil moisture is too low, please help me to adjust the flow rate of the water valve in the greenhouse to the maximum.)
Industry	4282	Basic command (740)	打开锅炉的泄压阀的开关。 (Open the pressure relief valve of the boiler.)
		Complex command (3542)	检查已经完成，将车间里指示灯的颜色变成蓝色，提示员工可以休息了。 (The safety check has been completed, turning the color of the indicator lights in the workshop to blue, signaling employees are ready to take a break.)
Smart home	4268	Basic command (793)	查看客厅里空调的温度。 (Check the temperature of the air conditioner in the living room)
		Complex command (3475)	将客厅里的空调的温度设置在25度，同时关闭房间里加湿器的电源。 (Set the temperature of the air conditioner in the living room at 25 degrees and turn off the power of the humidifier in the room.)

N_s, which is calculated as follows.

$$\text{Precision} = \frac{N_{ts}}{N_s} \tag{10}$$

Recall represents the proportion of the number of correct structured logical forms N_ts obtained by the corresponding algorithm to the number of all structured logical forms N_sd contained in the command, which is calculated as follows.

$$\text{Recall} = \frac{N_{ts}}{N_{sd}} \tag{11}$$

The F-score is the summed average of precision and recall, which can reflect the performance of the semantic parsing algorithm in a comprehensive way, and is calculated as follows.

$$F - \text{score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{12}$$

B. PERFORMANCE EVALUATION OF SEMANTIC PARSING ALGORITHMS BASED ON DEPENDENCY PARSING

This subsection evaluates the performance of the semantic parsing algorithm based on dependency parsing using samples of basic natural language commands from the HCIC dataset. Considering the effect of the number of samples and the fields to which they belong on the experimental results, the experiments will be conducted separately for each of the three fields and 50/100/150/200 samples will be randomly selected from the data sets of the respective fields for testing. In addition, we recorded the maximum time consumption for parsing natural language commands to evaluate whether the response time of the semantic parsing algorithm meets the requirements. The experimental results for natural language commands for the three fields are shown in Tables 5, 6, and 7, respectively. Analysis of the above experimental results leads to the following conclusions.

TABLE 5. Test results for parsing basic natural language commands in agriculture.

No.	Number	Accuracy	Precision	Recall	F-score	Max-Time
1	50	93.33	92.14	91.43	92.29	1.25s
2	100	91.67	90.89	90.44	90.66	1.14s
3	150	89.87	90.24	89.23	89.73	1.21s
4	200	89.67	89.81	88.93	89.37	1.47s
-	Avg	91.16	90.77	90.01	90.51	1.27s

TABLE 6. Test results for parsing basic natural language commands in industry.

No.	Number	Accuracy	Precision	Recall	F-score	Max-Time
1	50	90.33	89.15	87.64	88.39	1.43s
2	100	88.10	90.10	87.23	88.69	1.24s
3	150	87.67	87.56	86.14	86.84	1.32s
4	200	87.25	86.45	85.43	85.94	1.61s
-	Avg	88.40	88.36	86.61	87.47	1.40s

TABLE 7. Test results for parsing basic natural language commands in smart home.

No.	Number	Accuracy	Precision	Recall	F-score	Max-Time
1	50	92.73	90.32	89.54	89.93	1.24s
2	100	91.24	90.21	88.67	89.43	1.31s
3	150	90.39	89.78	88.53	89.15	1.31s
4	200	89.27	88.58	87.91	88.24	1.25s
-	Avg	90.91	89.72	88.67	89.19	1.28s

- With the increase in the number of commands, the metrics for generating the structured logical form are decreasing in the range of about 2%-4%, which is within the acceptable range.
- In the field of agriculture and smart home, the semantic parsing algorithm based on dependency parsing analysis has an average value of about 90% for all evaluation metrics, and the parsing time is within 1.3s, which can be considered to have good results in the field of agriculture and smart home, and the response time meets the requirements.
- For basic natural language commands in the industry, the parsing algorithms have relatively low relevant metrics and long parsing times. The reason for this is the long length of commands in industry and the low ability of dependency parsing tools to parse long sequences. However, the accuracy and response time are still within the acceptable range.

C. PERFORMANCE EVALUATION OF SEMANTIC PARSING ALGORITHM BASED ON TWO-LEVEL TAGGERS

1) MODEL TRAINING

This subsection uses 10458 samples of complex language commands from HCIC to train and test the model of semantic parsing algorithm based on two-level taggers, where the ratio of training set, validation set and test set is 8:1:1. It is worth noting that the number of samples from different fields is similar in each set. IoT-NLI uses the open-source BERT-wwm from HIT as the pre-training model. The maximum length of one-time encoding is 60, the batch size is 16, and the learning rate is set to 1e-5. The loss function is set to the sum of the binary cross-entropy of all predicted tag lists and the true tag lists. The change curves of accuracy, precision, recall and F-score during training are shown in Figure 9, and the change curves of Loss during training are shown in Figure 10. After 8 epochs of training, the accuracy of the model in the test set is 73.52%. The parsing accuracy is within a reasonable range considering the complexity of this task and the weakening of parsing ability in the face of commands containing multiple structured logical forms.

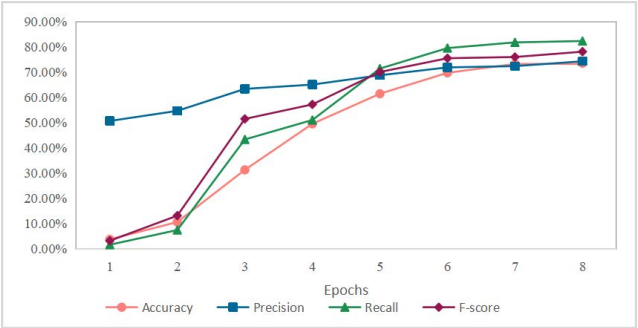

FIGURE 9. Accuracy, precision, recall and F-score variation curve.

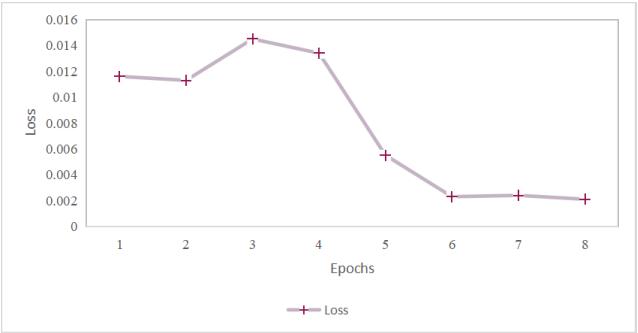

FIGURE 10. Loss variation curve.

TABLE 8. Test results for parsing complex natural language commands in agriculture.

No.	Number	Accuracy	Precision	Recall	F-score	Max-Time
1	50	75.14	74.78	82.67	78.53	3.72s
2	100	73.93	73.54	79.42	76.37	3.83s
3	150	72.16	73.81	80.45	76.99	3.76s
-	Avg	73.74	74.04	80.85	77.30	3.77s

2) SUITABILITY TESTING

To explore the actual performance of the model in each field, we took 150 samples from the test set, in each field, for the suitability test of the model, and the experimental results are shown in Tables 8, 9 and 10.

TABLE 9. Test results for parsing complex natural language commands in industry.

No.	Number	Accuracy	Precision	Recall	F-score	Max-Time
1	50	73.67	73.42	80.54	76.82	3.31s
2	100	72.36	73.01	78.40	75.61	3.64s
3	150	72.24	72.54	78.36	75.34	3.73s
-	Avg	73.09	72.99.	79.10	75.92	3.56s

TABLE 10. Test results for parsing complex natural language commands in smart home.

No.	Number	Accuracy	Precision	Recall	F-score	Max-Time
1	50	73.34	74.98	81.43	78.07	3.54s
2	100	73.12	75.17	80.24	77.62	3.71s
3	150	74.02	75.02	80.14	77.50	3.63s
-	Avg	73.49	75.06	80.60	77.73	3.63s

TABLE 11. Test results of CSRG2API module.

Field	Type	Test situation	Accuracy	Max-Time
Agriculture	Query	77/77	100.00	0.17s
-	Control	71/73	97.26	0.21s
Industrial	Query	71/72	98.61	0.16s
-	Control	76/78	97.44	0.19s
Smart Home	Query	73/76	97.37	0.17s
-	Control	72/74	97.30	0.18s

The analysis of the above data leads to the following conclusions.

- The results of the independent tests in the three fields are less different from the results of the mixed tests. It can be concluded that the model has good results in all three fields of agriculture, industry, and smart home.
- The maximum time taken for the parsing process of the model in the three fields is within 4s, which is longer than the semantic analysis algorithm based on dependency parsing. However, since the algorithm is used to parse more complex natural language commands, the time taken is still within an acceptable range and meets the general requirements for real-time interaction.

D. PERFORMANCE EVALUATION OF CSRG2API MODULE

The CSRG2API module is a CSRG-oriented API generation method proposed in this paper, which aims to use CSRG and parameter dependencies to select the appropriate APIs and fill them automatically. This subsection will use a sample of 150 structured logical forms in each field, the results of which are shown in Table 11.

E. COMPOSITE SERVICE PERFORMANCE EVALUATION OF IOT-NLI

We evaluate the comprehensive service performance of IoT-NLI in this subsection. To ensure the fairness of the experimental process, the evaluation process of all methods is performed using 600 natural language commands randomly selected from the HCIC dataset. The ratio of basic natural language commands to complex natural language commands is 3:2, and complex natural language commands are extracted from the test set. the uniqueness of IoT-NLI is that it tries to extract all the information needed in the API from natural language commands. We tried our best to migrate several common approaches to building natural language interfaces for IoT platforms to our task for comparative experiments. It should be noted that the training and validation sets during the training of the deep learning-based approach are consistent with the IoT-NLI, only in a different form. The experimental results are shown in Table 12.

TABLE 12. Test results of IoT-NLI composite service performance.

No	Method	Test situation	Call success rate	Avg-Time
1	Seq2Seq[11]	123/600	20.50	3.13s
2	BERT-Seq2seq	159/600	26.50	3.31s
3	BiLSTM-CRF[13]	214/600	35.67	2.30s
4	BERT-BiLSTM-CRF	254/600	42.33	2.51s
5	DSA[10]	327/600	54.50	1.59s
6	IoT-NLI-Basic	335/600	55.83	1.52s
7	IoT-NLI-Complex	503/600	83.83	3.84s
8	IoT-NLI	502/600	83.67	2.76s

Gui *et al.* [11] proposed the use of the SeqSeq model to construct an industrial IoT natural language interface to convert natural language commands into their corresponding logical form. This is in line with the goal of IoT-NLI in the structured parsing process. In addition, a variant model using BERT for encoding was added to the scope of the comparison experiments. Batch size was set to 64 and epoch was set to 15 during the training of the Seq2Seq model. Experimental results showed that the Seq2Seq model failed to achieve good results under the current task, which may be mainly due to the long sequence length of the commands and the Seq2Seq model's feature extraction is not obvious enough. Park *et al.* [13] developed a natural language interface for smart home scenarios using BiLSTM-CRF as a main model for named entity recognition of keywords in commands. Their variant using BERT for encoding was also added to the comparison scope. The problem with the sequence annotation model is that although it is able to extract entities from natural language commands more accurately, the relationships between keywords will be ignored in the extraction results. This leads to the inability of such models to effectively parse natural language commands with complex logic. The DSA algorithm proposed by Wu *et al.* [10] uses a dependency syntax tree for instruction analysis to extract the " ACTION"-"LOCATION"-"DEVICE" triples from the commands to simple manipulation of equipment. We add the rule 3 of semantic parsing algorithm based on dependency parsing to make it applicable to our task. DSA solves the problem of parsing natural language commands containing multiple operations with its unique rules, but the drawback is obvious: the dependency analysis approach relies heavily on the

(a) Authentication (b) Natural Language Interface (c) Air conditioning is on (d) The temperature of the air conditioner has been adjusted

FIGURE 11. IoT-NLI application.

defined rules, which leads to weak generalization capabilities and makes it difficult to parse commands with complex logic.

Compared with the above approaches, IoT-NLI solves the problem of parsing natural language commands containing multiple operations and complex logic with the help of its hierarchical semantic parsing framework and unique graph structure. In the ablation experiments, IoT-NLI-Basic (parsing using only semantic parsing algorithm based on dependency parsing) is not strong enough for parsing complex natural language commands, but significantly reduces the parsing time. IoT-NLI-Complex (parsing using only semantic parsing algorithm based on two-level taggers) has stronger parsing power and consumes longer time. The hierarchical framework effectively reduces the average parsing time while ensuring accuracy.

V. NATURAL LANGUAGE INTERFACE

In this subsection, we developed a mobile application for the Android platform to integrate IoT-NLI with an IoT platform with an interactive interface similar to popular instant messaging software such as WhatsApp, WeChat, and LINE. In this way, we hope to minimize the learning cost and distance of users to the new interaction method, so that they can get started quickly. When users first use the IoT-NLI application, they need to fill in their AK (Access Key ID) and SK (Secret Access Key) to authenticate with the IoT platform. The authentication interface is shown in Figure 11(a). (During the demonstration of this application, IoT-NLI is chosen to be integrated into AliCloud IoT platform according to the sample in this paper). After completing the authentication, the application provides a natural language interface as well as a device status display interface, and the device status is obtained by the software making continuous requests to the IoT platform.

The operation process is shown in Figure 11(b). The IoT-NLI semantic parsing framework receives the command "请先帮我打开客厅里的空调的开关，再将温度设置到24度" ("Please turn on the switch of the air conditioner in the living room for me first, and then set the temperature to 24 °C")and parses it, then fills in the key information into the appropriate API with the help of parameter dependencies. While returning the parsing result to the user, the software will send an HTTPS/HTTP GET or POST request to the OpenAPI server of the IoT platform to invoke the API just generated, and the platform will perform parameter setting or data query operation on the specific IoT device according to the processing of the request. In Figure 11(c) as well as in Figure 11(d), the software provides feedback in the graphical interface on the commands just entered by the user (i.e., in Figure 11(c), the air conditioner has been turned on, while in Figure 11(b), the air conditioner temperature has been lowered from 26 °C to 24 °C) The user can use the device status display interface to determine whether the operation is successful, observe the device status before and after device manipulation, and perform next operation.

VI. CONCLUSION

In this paper, we design and propose IoT-NLI, a natural language query and control interface for IoT platforms. IoT-NLI accomplishes the operation of converting natural language commands into APIs with the help of command semantic relationship graph, CSRG2API module and parameter dependencies. Finally, the software queries and controls the devices by sending requests to the OpenAPI server. To test the performance of the interface, we built a Chinese natural language

command dataset by crowdsourcing. The experiments on the datasets of agriculture, industry and smart home show that IoT-NLI has a relatively good parsing performance and the time consumption meets the basic requirements of human-computer interaction. With its unique graph structure, IoT-NLI solves to a certain extent the problem of parsing natural language commands that contain multiple operations and more complex logical structures, enriching user scenarios and making user interaction more free and coherent.

This study has two main limitations that could be investigated in more depth in future work. First, the study focuses on the construction of Chinese natural language interface for IoT platforms, which are not very migratory for other languages. This is due to the fact that the semantic parsing algorithm used in the framework based on dependency parsing relies on the syntactic structural properties of Chinese commands. Nevertheless, in terms of ideas, the algorithm can be adapted to different languages by researchers who redefine the relevant rules according to the characteristics of different languages. However, in future work, we hope to further explore generic frameworks that can be used in different languages. Second, during the interaction, the user commands must mention all the information needed by the API. This makes the user use process slightly cumbersome. In future work, we hope to integrate research related to natural language inference into natural language interfaces or provide multimodal input for users to accommodate more information. Overall, our approach provides a more convenient way for users to interact with the IoT platform. In future work we will further explore the potential of natural human-computer interaction approaches.

ACKNOWLEDGMENT

The authors sincerely thank the anonymous reviewers for their helpful comments and suggestions. They also sincerely thank Ziwei Niu from the School of Computer Science and Technology, Zhejiang University, for his valuable suggestions.

REFERENCES

[1] J. Wang, M. K. Lim, C. Wang, and M.-L. Tseng, "The evolution of the Internet of Things (IoT) over the past 20 years," *Comput. Ind. Eng.*, vol. 155, May 2021, Art. no. 107174, doi: 10.1016/j.cie.2021.107174.

[2] J. Grudin, "From tool to partner: The evolution of human-computer interaction," *Synth. Lectures Hum.-Centered Informat.*, vol. 10, no. 1, pp. 1–183, Jan. 2017, doi: 10.2200/S00745ED1V01Y201612HCI035.

[3] I. Androutsopoulos and M. Aretoulaki, *Natural Language Interaction*, vol. 1. Oxford, U.K.: Oxford Univ. Press, 2012, doi: 10.1093/oxfordhb/9780199276349.013.0035.

[4] M. B. Hoy, "Alexa, Siri, Cortana, and more: An introduction to voice assistants," *Med. Reference Services Quart.*, vol. 37, no. 1, pp. 81–88, Jan. 2018, doi: 10.1080/02763869.2018.1404391.

[5] M. Noura, S. Heil, and M. Gaedke, "GROWTH: Goal-oriented end user development for web of things devices," in *Proc. Int. Conf. Web Eng.*, 2018, pp. 358–365.

[6] J. Austerjost, M. Porr, N. Riedel, D. Geier, T. Becker, T. Scheper, D. Marquard, P. Lindner, and S. Beutel, "Introducing a virtual assistant to the lab: A voice user interface for the intuitive control of laboratory instruments," *SLAS Technol.*, vol. 23, no. 5, pp. 476–482, Oct. 2018, doi: 10.1177/2472630318788040.

[7] C. J. Baby, F. A. Khan, and J. N. Swathi, "Home automation using IoT and a chatbot using natural language processing," in *Proc. Innov. Power Adv. Comput. Technol. (i-PACT)*, Apr. 2017, pp. 1–6, doi: 10.1109/IPACT.2017.8245185.

[8] M. Mehrabani, S. Bangalore, and B. Stern, "Personalized speech recognition for Internet of Things," in *Proc. IEEE 2nd World Forum Internet Things (WF-IoT)*, Dec. 2015, pp. 369–374, doi: 10.1109/WF-IoT.2015.7389082.

[9] M. Noura, S. Heil, and M. Gaedke, "Natural language goal understanding for smart home environments," in *Proc. 10th Int. Conf. Internet Things*, Malmö, Sweden, Oct. 2020, pp. 1–8, doi: 10.1145/3410992.3410996.

[10] H. Wu, C. Shen, Z. He, Y. Wang, and X. Xu, "SCADA-NLI: A natural language query and control interface for distributed systems," *IEEE Access*, vol. 9, pp. 78108–78127, 2021, doi: 10.1109/ACCESS.2021.3083540.

[11] Z. Gui and A. Harth, "Towards a data driven natural language interface for industrial IoT use cases," in *Proc. IEEE 2nd Int. Conf. Hum.-Mach. Syst. (ICHMS)*, Magdeburg, Germany, Sep. 2021, pp. 1–3, doi: 10.1109/ICHMS53169.2021.9582450.

[12] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 27, 2014, pp. 1–9.

[13] G. Park and H. Kim, "Low-cost implementation of a named entity recognition system for voice-activated human-appliance interfaces in a smart home," *Sustainability*, vol. 10, no. 2, p. 488, Feb. 2018, doi: 10.3390/su10020488.

[14] J. Ye, L. Zhang, P. Lan, H. He, D. Yang, and Z. Wu, "Improved intelligent semantics based Chinese sentence similarity computing for natural language processing in IoT," in *IoT as a Service*, vol. 346, B. Li, C. Li, M. Yang, Z. Yan, and J. Zheng, Eds. Cham, Switzerland: Springer, 2021, pp. 234–246, doi: 10.1007/978-3-030-67514-1_19.

[15] L. Qin, T. Xie, W. Che, and T. Liu, "A survey on spoken language understanding: Recent advances and new frontiers," 2021, *arXiv:2103.03095*.

[16] H. Kim, B.-H. So, W.-S. Han, and H. Lee, "Natural language to SQL: Where are we today?" *Proc. VLDB Endowment*, vol. 13, no. 10, pp. 1737–1750, Jun. 2020, doi: 10.14778/3401960.3401970.

[17] X. Victoria Lin, C. Wang, L. Zettlemoyer, and M. D. Ernst, "NL2Bash: A corpus and semantic parser for natural language interface to the Linux operating system," 2018, *arXiv:1802.08979*.

[18] A. Ait-Mlouk and L. Jiang, "KBot: A knowledge graph based ChatBot for natural language understanding over linked data," *IEEE Access*, vol. 8, pp. 149220–149230, 2020, doi: 10.1109/ACCESS.2020.3016142.

[19] S. Mazumder and O. Riva, "FLIN: A flexible natural language interface for web navigation," 2020, *arXiv:2010.12844*.

[20] X. Xu, C. Liu, and D. Song, "SQLNet: Generating structured queries from natural language without reinforcement learning," 2017, *arXiv:1711.04436*.

[21] Y. Su, A. Hassan Awadallah, M. Wang, and R. W. White, "Natural language interfaces with fine-grained user interaction: A case study on web APIs," in *Proc. 41st Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jun. 2018, pp. 855–864, doi: 10.1145/3209978.3210013.

[22] S. Mazumder, B. Liu, S. Wang, and N. Ma, "Lifelong and interactive learning of factual knowledge in dialogues," in *Proc. 20th Annu. SIGdial Meeting Discourse Dialogue*, 2019, pp. 21–31, doi: 10.18653/v1/W19-5903.

[23] H. Wang, M. Tan, M. Yu, S. Chang, D. Wang, K. Xu, X. Guo, and S. Potdar, "Extracting multiple-relations in one-pass with pre-trained transformers," 2019, *arXiv:1902.01030*.

[24] Z. Wei, J. Su, Y. Wang, Y. Tian, and Y. Chang, "A novel cascade binary tagging framework for relational triple extraction," 2019, *arXiv:1909.03227*.

[25] Z. Jie and W. Lu, "Dependency-guided LSTM-CRF for named entity recognition," in *Proc. Conf. Empirical Methods Natural Lang. Process., 9th Int. Joint Conf. Natural Lang. Process. (EMNLP-IJCNLP)*, Hong Kong, 2019, pp. 3860–3870, doi: 10.18653/v1/D19-1399.

[26] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural architectures for named entity recognition," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, San Diego, CA, USA, 2016, pp. 260–270, doi: 10.18653/v1/N16-1030.

[27] S. Zhang, L. Wang, K. Sun, and X. Xiao, "A practical Chinese dependency parser based on a large-scale dataset," 2020, *arXiv:2009.00901*.

[28] T. Dozat and C. D. Manning, "Simpler but more accurate semantic dependency parsing," in *Proc. 56th Annu. Meeting Assoc. Comput. Linguistics*, Melbourne, VIC, Australia, vol. 2, 2018, pp. 484–490, doi: 10.18653/v1/P18-2077.

[29] A. Hogan, E. Blomqvist, M. Cochez, C. D'amato, G. D. Melo, C. Gutierrez, S. Kirrane, J. E. L. Gayo, R. Navigli, S. Neumaier, A.-C.-N. Ngomo, A. Polleres, S. M. Rashid, A. Rula, L. Schmelzeisen, J. Sequeda, S. Staab, and A. Zimmermann, "Knowledge graphs," *ACM Comput. Surveys*, vol. 54, no. 4, pp. 1–37, May 2022, doi: 10.1145/3447772.
[30] R. Angles and C. Gutírrez, "Survey of graph database models," *ACM Comput. Surv.*, vol. 40, no. 1, pp. 1–39, Feb. 2008, doi: 10.1145/1322432.1322433.
[31] R. Angles, M. Arenas, P. Barceló, A. Hogan, J. Reutter, and D. Vrgoč, "Foundations of modern query languages for graph databases," *ACM Comput. Surveys*, vol. 50, no. 5, pp. 1–40, Sep. 2018, doi: 10.1145/3104031.
[32] C. F. Baker, C. J. Fillmore, and J. B. Lowe, "The Berkeley FrameNet project," presented at the 17th Int. Conf. Comput. Linguistics, vol. 1, 1998.
[33] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North*, Minneapolis, MN, USA, 2019, pp. 4171–4186, doi: 10.18653/v1/N19-1423.

ZHIPENG XU was born in Hefei, Anhui, China. He is currently pursuing the bachelor's degree with the School of Information and Computer Science, Anhui Agricultural University. His research interests include natural language processing, knowledge graphs, and human–computer interaction.

HAO WU was born in Anhui, China. He is currently pursuing the bachelor's degree in computer science and technology with the School of Information and Computer Science, Anhui Agricultural University. He has been guaranteed a place at the Department of Computer Science, University of Science and Technology of China. His research interests include deep learning, natural language processing, and human–computer dialogue interaction.

XU CHEN is currently pursuing the bachelor's degree with the School of Information and Computer Science, Anhui Agricultural University. His research interests include information extraction and natural language generation and inference.

YONGMEI WANG was born in Hefei, Anhui, China, in 1974. She received the master's degree in computer science and technology from the School of Information and Computer Science, Hefei University of Technology, in 2010. She has been engaged in teaching and research in the fields of big data, agricultural informatization, and agricultural products traceability. She has presided over and participated in more than ten national and provincial educational and scientific research projects.

ZHENYU YUE received the Ph.D. degree from Anhui University. He is currently a Teacher at the School of Information and Computer, Anhui Agricultural University. His research interests include development of bioinformatics tools and databases and deep learning.