Development of a problem solver for automatic answer verification in the intelligent tutoring systems

Wenzu Li, Longwei Qian
Belarussian State University Informatics and Radioelectronics
Minsk, Belarus
lwzzggml@gmail.com, qianlw1226@gmail.com

***Abstract*—This article proposes an approach to developing a problem solver for automatic answer verification in intelligent tutoring systems constructed using OSTIS Technology. The problem solver is developed based on multi-agent technology. The developed problem solver automatically verifies the correctness and completeness of the user answer at the semantic level by calling the corresponding sc-agent sets according to the type of the questions (multiple-choice questions, fill in the blank questions, questions of definition interpretation, etc.). Sometimes there may be multiple standard answers for the question of definition interpretation, but the problem solver can automatically filter a semantic fragment of the standard answer in advance that best matches the user answer according to the semantic fragment of the user answer, and then continue to verify the correctness and completeness between them.**

***Keywords*—problem solver, answer verification, OSTIS Technology, intelligent tutoring systems, ontology, knowledge base**

I. Introduction

With the development of modern information technology, artificial intelligence as an important part of modern computer applications is rapidly integrating into the field of education. Applying artificial intelligence technology to the field of education can not only provide new teaching methods and tools, but also enable learners to spend less time acquiring more useful knowledge, effectively improving the accuracy of information retrieval and the efficiency of knowledge selection. At the same time, the combination of artificial intelligence technology and the educational process is also one of the important means to ensure the fairness of education, so that every learner has an equal opportunity to obtain knowledge. Especially since the outbreak of COVID-19, intelligent tutoring systems (ITS) have played an increasingly important role in distance education [3].

As more and more ITS used in different fields are developed, most developers believe that ITS need to meet at least the following basic functions:

- automatic verification of answers, if an error is found, the cause of the error needs to be analyzed and corrective measures taken;
- according to the learner's level and learning situation, automatically adjust the learning content and progress;
- automatic generation of various questions and exercises;
- have the ability to generate and understand natural language, and realize relatively free Human-Machine conversation;
- have the ability to explain teaching content;
- automatic problem solving based on understanding the teaching content.

As the most basic and critical function of ITS, automatic answer verification can quickly check the user's grasp of new knowledge, and can greatly improve the user's learning efficiency, allowing users to obtain the most knowledge in a limited time. Usually answer verification needs to solve the following basic tasks:

1) subjective question answer verification and objective question answer verification;
2) analysis of correctness and completeness of answers;
3) verification a type of question with multiple standard answers, and these standard answers do not satisfy logical equivalence (for example, the definition of a square);
4) whether the logic formula of standard answer and user answer meets the equivalence;
5) analysis of decision sequence and logic rationality when users solve problems;

According to the types of questions, automatic answer verification can be divided into: 1. objective question answer verification; 2. subjective question answer verification. Objective questions refer to a type of question that has a unique standard answer. In this article, objective questions include: multiple-choice questions, fill in the blank questions and judgment questions. Objective questions differ from subjective questions, which have more than one potential correct answer and sometimes have room for a justified opinion. Subjective questions in this article include: definition explanation questions, proof questions and theorem interpretation questions. Because the answers to objective questions are fixed and simple, ITS on the market basically have the function of objective question answer verification. However, since subjective question answer verification needs to involve natural language processing (NLP), linguistics and other aspects of knowledge, currently only some ITS have the function of subjective question answer verification [4], [7].

With the development of semantic web, deep learning and NLP technology, subjective question answer verification has become a very important research direction. Therefore, we have introduced in detail the existing subjective question answer verification approaches and their advantages and disadvantages in the literature [4], and on the basis of these approaches, we propose an semantic-based answer verification approaches (subjective question answer verification and objective question answer verification). The basic principle of this approach is to first decompose the standard answers and user answers in the form of semantic graphs into substructures according to the knowledge representation rules in the ostis-systems (system built using OSTIS Technology (Open Semantic Technology for

Intelligent Systems)), and then calculate the similarity between the semantic graphs by comparing the matching relationships of the decomposed substructures, finally, the correctness of the user answer is judged according to the similarity [1], [2], [6]. A semantic graph is a network that represents semantic relationships between concepts. In the ostis-systems, the semantic graph is constructed using SC-code (as a basis for knowledge representation within the OSTIS Technology, a unified coding language for information of any kind based on semantic networks is used, named SC-code). The user answer in natural languages is converted to SC-code using the natural language interface [5].

In literature [4], we only briefly introduced the process of using semantics to verify user answers at the theoretical level, and conducted a feasibility analysis of the proposed approach, but the article does not involve the specific process of using the program to implement each step. Therefore, in this article, a problem solver for answer verification in the ostis-systems is developed based on the answer verification approach proposed in [4]. One of the key components of each intelligent system is the problem solver, which provides the ability to solve various problems. The developed problem solver mainly solves the tasks (1), (2), (3) listed above, and for the solutions to the remaining more complex tasks, we will introduce them in detail in the subsequent articles. The discrete mathematics tutoring system developed using OSTIS Technology will serve as a demonstration system for the problem solver.

II. Existing approaches and problems

According to the type of knowledge in the knowledge base of the ITS, answer verification can be divided into:

- factual knowledge answer verification;
- logical knowledge answer verification.

Factual knowledge refers to knowledge that does not contain variable types, and this type of knowledge expresses facts. In the knowledge base of ostis-systems, objective questions and their answers are usually described using factual knowledge. Among them, the user answers to objective questions in the form of natural language have been aligned (entity alignment) with the existing knowledge in the knowledge base when they are transformed into SC-code through the natural language interface. Therefore, there is no need to consider the similarity between concepts or relations at the language level when calculating the similarity between the answers to objective questions. That is, SC-nodes that represent the same concept or relation in the knowledge base have a unique main identifier.

Logical knowledge usually contains variables, and there are logical relations between knowledge. In the ostis-systems SCL-code (a special sub-language of the SC language intended for formalizing logical formulas) is used to represent logical knowledge. The answers to subjective questions in the knowledge base are described in the form of logical formula using logical knowledge. Because, the semantic segment used to represent the answer to subjective question in the knowledge base contains variables (equivalent to the bound variable in the predicate logic formula), and there is a strict logical sequence between each sc-node in the semantic segment. Therefore, when using the problem solver to calculate the similarity between the semantic graph of standard answer and the semantic graph of user answer, it is necessary to establish the mapping relationship between the potential equivalent variables between the two semantic graphs according to the semantic structure and the position of the variables in the logical formula. Among them, establishing the mapping relationship of potential equivalent variables between semantic graphs is the most basic and critical step in subjective question answer verification. In this article, we can regard semantic graphs describing standard answers and user answers as partial fragment of the ontology (ontology is a type of knowledge, each of which is a specification of the corresponding subject domain, focused on describing the properties and relations of concepts that are part of the specified subject domain) [2], [4], [6]. Therefore, the approach to establishing a mapping relationship between semantic graphs is similar to the approach to establishing a mapping relationship between ontology. At present, there are many mature tools and approaches to establishing the mapping relationship between ontology, and we will consider them in detail next.

Ontology mapping

With the rapid development of the new generation of semantic web, the ontology as the foundation of the semantic web has become a research hot-spot, and many ontology libraries with rich semantic information and practical value have emerged. These ontology libraries have huge differences due to different developers, application purposes and application fields, and they cannot communicate and inter-operate effectively with each other. The ontology mapping is a key step to solve the heterogeneity of ontology and realize knowledge sharing and ontology inter-operation. The core idea of ontology mapping is to calculate the similarity between elements (concepts, attributes, and instances) in different ontology, and then establish the mapping relationship between the elements according to the similarity and mapping strategy.

Due to the rapid development of ontology mapping related technologies, many concepts with similar semantics and different names have emerged, such as **Ontology Mapping**, **Ontology Alignment**, **Ontology Matching**, **Ontology Integration**, **Ontology Fusion**, and **Ontology Merging**. It is generally believed that ontology mapping, ontology alignment and ontology matching are concepts with the same semantics, that is, the mapping relationship between elements in different ontology is established according to the mapping strategy. Ontology integration, ontology fusion, and ontology merging generally produce new ontology, and ontology mapping is their basic task [10], [11]. Ontology mapping involves multiple steps such as ontology preprocessing. Since this article focuses on the establishment of element mapping relationships between ontology, other steps will not be introduced in detail.

There are already many mature ontology mapping algorithms and mapping systems:

- a comprehensive similarity calculation algorithm that establishes semantic mapping relationships between elements (concepts, attributes, and instances) between RDFS ontology is introduced in literature [9]. Taking concepts between ontology as an example, the algorithm first uses Edit Distance (Levenshtein Distance) to calculate the similarity of names between concepts, and then calculates the similarity of the instances of the concept according to the ratio of the number of instances matched between the concepts and the number of all instances, and finally, the structural similarity of the concepts is calculated according to the relationship between the number of the same father and child concepts and the number of all adjacent concepts between the concepts. Combine the above three similarities and set different weights to get the final similarity of the concept. Finally, according to the comprehensive similarity between concepts and the mapping strategy, the mapping relationship of equivalent concepts between different ontology is established;
- with the rapid development of machine learning in recent years, many ontology mapping approaches based on machine learning frameworks have emerged. An approach

to alignment of entities between knowledge graphs based on machine learning is introduced in the literature [12]. The knowledge graph is regarded as a formal description of things and their interrelationships in the objective world. The approach consists of two parts: 1. knowledge representation learning; 2. learning of mapping relationships between entities; Knowledge representation learning refers to the use of machine learning algorithms to learn the semantic representation of entities and relationships in the knowledge graph. The learning of the mapping relationship between entities refers to learning the mapping relationship of entity pairs between knowledge graphs according to the manually labeled data sets [13];
- with the development of ontology mapping technology, many mature ontology mapping systems have emerged, among which the most representative ones are RiMOM and ASMOW. RiMOM is an ontology mapping system developed based on Bayesian decision theory. RiMOM uses similarity propagation theory and multiple heuristic rules to establish the mapping relationship between concepts. ASMOV is an automated ontology mapping tool developed by Jean-Mary et al. Its goal is to promote the integration of heterogeneous ontology. ASMOV uses an iterative calculation method to analyze multiple characteristics of elements to calculate the similarity of element pairs between ontology, and to establish mapping relationships between concepts, attributes, and instances in turn [15], [16].

Although the ontology mapping approaches introduced above have many advantages, they also have many problems:

- the traditional algorithm for calculating the similarity of element pairs between ontology requires iterative calculation of the similarity between the current element in the source ontology and each element in the target ontology. Therefore, when the amount of knowledge contained in the ontology is very large, it may take several minutes or more to establish the mapping relationship between the ontology, and real-time mapping cannot be performed;
- using machine learning algorithms for ontology mapping has improved the accuracy of ontology mapping to a certain extent, but it requires a huge amount of human resources to label matching element pairs between ontology;
- ontology mapping is a very complicated process, and no approach is perfect. Especially in recent years, the generation of big data has led to the generation of big ontology, but the existing ontology mapping system and approach cannot perform ontology mapping in a big data environment.

Establishing the mapping relationship of potential equivalent variable pairs between the semantic fragments of the answers to subjective question is a key step of logical knowledge answer verification. The part of the process of establishing the mapping relationship between potential equivalent variables pairs is similar to the establishment of the mapping relationship between the equivalent element pairs of the ontology. However, in the ostis-systems, the knowledge base is constructed using SC-code, and the knowledge in the knowledge base has a specific knowledge structure and knowledge representation approach, so the existing ontology mapping algorithms cannot be used directly. Therefore, based on the existing ontology mapping approach and OSTIS Technology, this article proposes an approach to establish the mapping relationship of the potential equivalent variables pairs between the semantic fragments of the answers to subjective questions based on the semantic structure.

III. Proposed approach

Based on the OSTIS Technology used to develop semantic intelligence systems and the corresponding platforms, tools and approaches, an approach to developing a problem solver for automatic answer verification is proposed in this article.

Each ostis-system for different application fields includes a platform for interpretation semantic models of ostis-systems, as well as a semantic model of ostis-systems using SC-code (sc-model of ostis-systems). At the same time, the sc-model of the ostis-systems includes the sc-model of the knowledge base, the sc-model of the problem solver and the sc-model of the interface (in particular, the user-oriented intelligent interface). The rules and methods for detailed design of the knowledge base and problem solver in the ostis-systems are described in [1].

Using the models, application tools and approaches provided by OSTIS Technology in the framework of this work will provide the following possibilities:

- the developed problem solver can be easily transplanted to the ostis-systems for different application fields;
- verifying the correctness and completeness of user answers at the semantic level;
- saving the user's test record;
- analyzing the user's test results from the semantic layer and logic layer, and give reference opinions;
- answer verification does not depend on the natural language (English, Russian, Chinese, etc.).

Next, we will introduce in detail the development process of the problem solver for automatic answer verification. In order to facilitate the explanation of the working principle of the problem solver, the illustrations and knowledge base fragments we choose in this article are all in English, but it needs to be emphasized that the problem solver developed does not depend on natural language.

IV. Development of problem solver

In the ostis-systems, the problem solvers are constructed based on a multi-agent approach. According to this approach, the problem solver is implemented as a set of agents called sc-agents. All sc-agents interact through common memory, passing data to each other as semantic network structures (sc-texts) [1], [2].

According to the requirements of the task, the developed problem solver for answer verification in this article needs to solve the following problems:

- the problem solver can not only verify the answer to objective question with the only correct option, but also verify the answer to objective question with multiple correct options (multiple-choice questions with multiple options and partially fill in the blank questions). If the user's answer is incorrect or incomplete, the correct standard answer needs to be displayed at the end;
- for subjective questions, it is necessary to verify the completeness and correctness of the user answer (for example, the answer is correct but incomplete, or the answer is partially correct, etc.). If the subjective question has multiple logically unequal standard answers (for example, the definition of triangle), the problem solver can automatically select the appropriate standard answer according to the user answer, and then verify the answer. Finally, if the user answer is incorrect (complete error or partial error), the problem solver also needs to find the incorrect part of the user's answer and display the correct standard answer.

The problem solver of any ostis-system is a hierarchical system of knowledge processing agents in semantic memory. In order to achieve the tasks listed above, some sc-agents are developed in this article. The hierarchy of the knowledge processing agents in the problem solver for automatic answer verification is shown in SCn-code (one of SC-code external visualization languages) [6] as follows:

Problem Solver For Automatic Answer Verification
⇐ *abstract sc-agent decomposition**:
{
- *Sc-agent for computing semantic similarity of factual knowledge*
- *Sc-agent for processing semantic similarity calculation results of factual knowledge*
- *Sc-agent for computing semantic similarity of logical knowledge*

}

The basic principle of automatic answer verification in the ostis-systems is to calculate the semantic similarity between the standard answer and the user answer. In the knowledge base, the answers to objective questions are described using factual knowledge, and the answers to subjective questions are described using logical knowledge [4]. Therefore, when using the problem solver to verify the answer, it realizes the automatic verification of the answer by calling different sc-agents according to the type of question. It should be emphasized that answer verification is only one of the main uses of the problem solver. The problem solver can also calculate the similarity between arbitrary semantic fragments constructed using sc-code by calling different sc-agents.

When verifying the answer to the objective question, the calling flow of sc-agents is as follows:

1) if the problem solver judges that the current question is an objective question, the sc-agent for computing semantic similarity of factual knowledge is called to start calculating the similarity between the semantic graph of standard answer and the semantic graph of user answer;
2) when the first step is completed, the problem solver automatically calls the sc-agent for processing semantic similarity calculation results of factual knowledge. The final verification result is given by this sc-agent according to the question type, characteristics (for example, multiple-choice questions with multiple correct options) and similarity between answers.

When verifying the answer to the subjective question, the calling flow of sc-agents is as follows:

- if the problem solver judges that the current question is a subjective question, the sc-agent for computing semantic similarity of logical knowledge is called.

It should be emphasized that the answers to subjective questions are described based on logical formula, so under certain conditions, the logical equivalence between the answers (equivalence judgment between the logic formulas) needs to be considered. Due to the limitation of the number of pages in this article, we will introduce the design approach and calling flow of sc-agent for verifying logical equivalence in the following article. Next, the specific functions and implementation process of each sc-agent will be introduced in detail.

A. Sc-agent for computing semantic similarity of factual knowledge

The basic function of the sc-agent for computing semantic similarity of factual knowledge is to calculate the similarity between semantic graphs described using factual knowledge. Because the answers to objective questions in the knowledge base are described using factual knowledge, the similarity between the answers can be calculated using this sc-agent. The similarity between answers is the basis for the objective question answer verification. When the user answer in natural language is converted to SC-code, it has been aligned with the existing knowledge in the knowledge base (such as coreference resolution, etc.), that is, elements with the same semantics have the same main identifier in the knowledge base. Therefore, in this article, the similarity between semantic graphs is calculated based on semantic and knowledge representation structures [4], [8].

The sc-agent for computing semantic similarity of factual knowledge needs to complete the following tasks:

1) according to the representation rules of factual knowledge, the standard answers and user answers in the form of semantic graphs are decomposed into substructures;
2) using formulas (1), (2), and (3) to calculate the precision P_{sc}, recall R_{sc} and similarity F_{sc} between semantic graphs.

$$P_{sc}(u,s) = \frac{|T_{sc}(u) \otimes T_{sc}(s)|}{|T_{sc}(u)|} \tag{1}$$

$$R_{sc}(u,s) = \frac{|T_{sc}(u) \otimes T_{sc}(s)|}{|T_{sc}(s)|} \tag{2}$$

$$F_{sc}(u,s) = \frac{2 \cdot P_{sc}(u,s) \cdot R_{sc}(u,s)}{P_{sc}(u,s) + R_{sc}(u,s)} \tag{3}$$

The main calculation parameters in the formulas include:

- $T_{sc}(u)$ — all substructures after the decomposition of the user answers u;
- $T_{sc}(s)$ — all substructures after the decomposition of the standard answers s;
- $\otimes$ — binary matching operator, which represents the number of matching substructures in the set of two substructures.

Next, we will introduce the working algorithm of this sc-agent in detail:

Algorithm 1 — The working algorithm of sc-agent for computing semantic similarity of factual knowledge

Input: The specific objective question and the corresponding semantic graph of standard answer and the semantic graph of user answer. The condition of the sc-agent response is that two semantic graphs that use factual knowledge to represent the answer appear in the sc-memory, and the similarity between them has not been calculated.

Output: The precision, recall and similarity between answers, and the sc-node used to record the matching status of substructures.

1) checking whether the standard answer and user answer exist at the same time, if so, go to step 2), otherwise, go to step 10);
2) according to the rules of factual knowledge representation (various types of sc-constructions), the semantic graphs of standard answers and user answers are decomposed into substructures [4];
3) iteratively traverse each substructure of the standard answer and user answer, classify them according to the type of substructure (three element sc-construction, five element sc-construction, etc.), and count the number of all substructures;
4) one type of substructure is randomly selected from the set of recorded standard answer substructure types;

5) according to the standard answer substructure type selected in step 4), a corresponding type of substructure is selected from the set of recorded user answer substructure types;
6) iteratively compare each substructure with the same substructure type between the standard answer and the user answer, and record the number of matched substructures and the matched substructures. The criterion for judging the matching of the same type of substructures is that the sc-nodes at the corresponding positions between the two substructures have the same main identifier. If the substructure contains sc-links, the contents of the sc-links at the corresponding positions must be also the same;
7) repeat step 4 — step 6 until all types of substructures have been traversed;
8) using formulas (1), (2), (3) calculate precision, recall and similarity, and generate semantic fragments for recording sc-agent running results;
9) removing all temporary sc-elements created while the sc-agent is running;
10) exit the program.

B. Sc-agent for processing semantic similarity calculation results of factual knowledge

The sc-agent for computing semantic similarity of factual knowledge only calculates the precision, recall and similarity between the standard answer and the user answer to the objective questions. However, because some objective questions have multiple correct options, it is necessary to comprehensively consider the precision, recall and similarity to fully judge the correctness of a question. Therefore, the sc-agent for processing semantic similarity calculation results of factual knowledge is developed in this article, its main function is to further judge the correctness and completeness of the current objective question based on the three information measurement parameters obtained in the previous step and the specific types and characteristics of the objective question [4].

The sc-agent for processing semantic similarity calculation results of factual knowledge needs to complete the following tasks:

1) judging the correctness and completeness of current question according to the precision, recall and similarity, as well as the evaluation strategies for the correctness and completeness of objective questions;
2) according to the correctness and completeness of the current question, generate some semantic fragments for prompting users;

The evaluation strategies for the correctness and completeness of objective questions mainly include the following:

- if the current question has the only correct option (multiple-choice questions with a correct option, judgment questions, and partially fill in the blank questions), then only the standard answer and the user answer exactly match, that is, the similarity is equal to 1 ($F_{sc} = 1$), the question is considered correct, otherwise the question is incorrect. The answer options in the semantic graph are described using the three element sc-construction;
- if the current question has multiple correct options (multiple-choice question with multiple correct options and partially fill in the blank questions), it can be subdivided into the following situations for judgment:
  - as long as the user answer contains the wrong option, the question is considered wrong. According to the definition of formulas (1), (2), (3), the similarity and precision are both less than 1 at this time ($F_{sc} < 1$ and $P_{sc} < 1$);
  - the all options included in the user answer are correct, but the number of correct options is less than the number of correct options in the standard answer, then the question is considered partially correct and incomplete. In this case, the precision is equal to 1, the similarity is less than 1, and the ratio of the number of all options in the user answer to the number of all options in the standard answer is the recall ($F_{sc} < 1$ and $P_{sc} = 1$);
  - if the options in the standard answer exactly match the options in the user answer, then the question is completely correct and complete. At this time, the similarity is equal to 1 ($F_{sc} = 1$);

Next, we will introduce the working algorithm of this sc-agent in detail:

Algorithm 2 — The working algorithm of sc-agent for processing semantic similarity calculation results of factual knowledge

Input: The semantic fragments of specific objective question, as well as the precision, recall and similarity between answers. The condition of the sc-agent response is that the similarity between the semantic graphs of the answers described using factual knowledge has been calculated, but the correctness and completeness of the answers have not been judged.

Output: The final answer verification result of a specific objective question, and the necessary semantic fragments used to display the answer verification result.

1) checking whether all input parameters used for sc-agent work meet the conditions, if so, go to step 2), otherwise, go to step 5);
2) according to the evaluation strategies for the correctness and completeness of objective questions, combined with the precision, recall, and similarity between answers, verify the correctness and completeness of specific objective question;
3) generating semantic fragments used to record the execution results of sc-agent;
4) removing all temporary sc-elements created while the sc-agent is running;
5) exit the program.

Combining sc-agent for computing semantic similarity of factual knowledge and sc-agent for processing semantic similarity calculation results of factual knowledge can verify the correctness and completeness of any type of objective question. Fig. 1 shows an example of using the problem solver to automatically verify the correctness and completeness of the answers to the multiple-choice questions in SCg-code (SCg-code is a graphical version for the external visual representation of SC-code) [1], [6].

C. Sc-agent for computing semantic similarity of logical knowledge

The basic function of the sc-agent for computing semantic similarity of logical knowledge is to calculate the similarity between semantic graphs described by logical knowledge. Because the answers to subjective questions in the form of semantic graphs in the knowledge base are described in the form of logical formula using logical knowledge (SCL-code), the similarity between the answers can be calculated using this sc-agent [1], [4]. Usually, the answers to subjective questions are not unique, so the similarity between answers becomes a key indicator for evaluating the correctness and completeness of user answers to subjective questions. Among them, user answers in natural language can be converted into SCL-code either manually or through natural language interfaces. Before

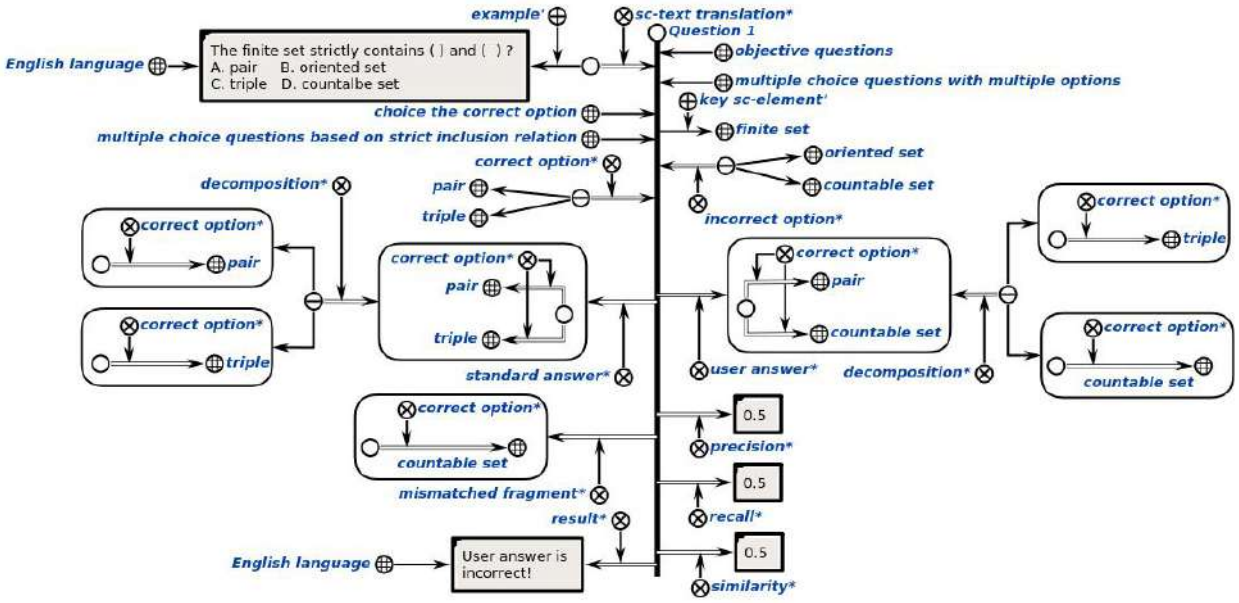

Figure 1. An example of automatic verification of answers to the multiple-choice questions

the subjective question answer verification, we assume that the factual knowledge contained in the user answer (for example, the constant sc-nodes used to represent concepts or relations) has been aligned with the existing knowledge in the knowledge base (through natural language interface) [5]. Therefore, in this article, the similarity between logical knowledge is calculated based on semantic and logical knowledge representation structures;

The sc-agent for computing semantic similarity of logical knowledge needs to complete the following tasks:

1) automatic selection of potential equivalent standard answer;
2) according to the representation rules of logical knowledge, the standard answer and user answer in the form of semantic graphs are decomposed into substructures;
3) establishing the mapping relationship of potential equivalent variable sc-node pairs between the semantic graph of the standard answer and the semantic graph of the user answer;
4) using formulas (1), (2), and (3) to calculate the precision P_{sc}, recall R_{sc} and similarity F_{sc} between semantic graphs.

Automatic selection of potential equivalent standard answer

Because some subjective questions usually have multiple standard answers (pre-stored in the knowledge base), and between the logic formulas used to formalize these answers do not satisfy logical equivalence. For example, the definition of equivalence relation: 1. in mathematics, an equivalence relation is a binary relation that is reflexive, symmetric and transitive; 2. for any binary relationship, if it is a tolerant relationship and is transitive, then it is an equivalence relation. Therefore, when calculating the similarity between answers, it is necessary to filter a standard answer that best matches the user answer from multiple possible standard answers in advance, and then calculate the similarity between them.

Because the answers to the subjective question in the knowledge base of the ostis-systems are described by logical knowledge in the form of logical formula. Therefore, if there are multiple standard answers to a question, and the logic formulas between them do not satisfy the equivalence, we find that the biggest difference between these answers is that the predicates used to describe them are different (that is, the constant sc-nodes used to represent concepts, relations, and elements in different answers are different) [10]. Therefore, this article proposes an approach to filtering the standard answer that best matches the user answer according to the similarity between all the predicates in the standard answer and all the predicates in the user answer.

The approach to filtering the standard answer that best matches the user answer according to the similarity between the predicates mainly includes the following steps:

1) if sc-agent judges that there are multiple standard answers to the current question, it will find all non-repeated predicates in each answer (the constant sc-nodes used to represent concepts, relations, and elements in the answer);
2) using formulas (1), (2), and (3) to iteratively calculate the similarity between all the predicates in the user answer and all the predicates in each standard answer. Here, the parameters in the formulas need to be given new meanings.
   - $T_{sc}(u)$ — all non-repeated predicates in the user answer u;
   - $T_{sc}(s)$ — all non-repeated predicates in the standard answer s;
   - $\otimes$ — binary matching operator, which represents the number of matching between all the predicates in the user answer and all the predicates in the standard answer.
3) choosing the standard answer with the greatest similarity to the user answer as the final standard answer.

Fig. 2 shows an example of filtering a standard answer in advance that best matches the user answer according to the predicate similarity between answers in SCg-code.

The establishment of mapping relationship of the potential equivalent variable sc-node pairs between answers

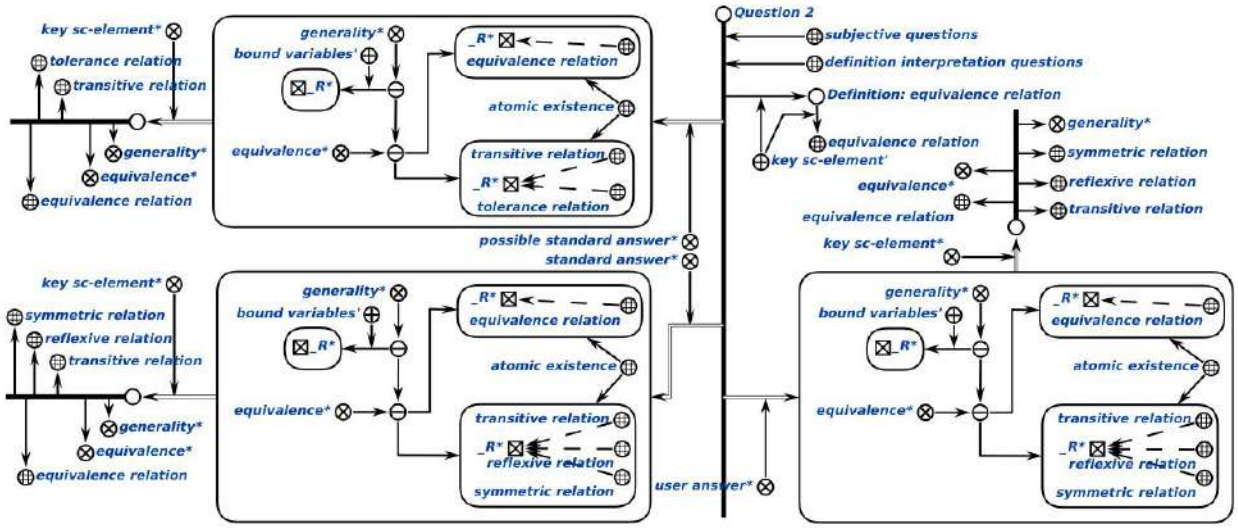

Figure 2. An example of filtering a standard answer that best matches the user answer according to the predicate similarity between answers

As we have already introduced, since the semantic graphs in the knowledge base used to describe the answers to subjective questions contain variables sc-nodes (equivalent to the bound variables in the predicate logic formula), when calculating the similarity between answers, the most critical step is to establish the mapping relationship (injection) of potential equivalent variable sc-node pairs between answers [11], [14]. Therefore, based on the existing ontology mapping methods, this article proposes an approach to establish the mapping relationship of potential equivalent variable sc-node pairs based on the semantic structures (various sc-constructions). In order to establish the mapping relationship of potential equivalent variable sc-node pairs, the following problems need to be solved first:

1) first, the position of the variable sc-nodes in the semantic graph needs to be determined;
2) it is necessary to determine the semantic connotation of the variable sc-nodes in the semantic graph.

Usually any predicate logic formula can be seen as consisting of two parts: 1. connective (such as negation ($\neg$) and implication ($\rightarrow$), etc.) used to describe logical relations, and quantifiers (universal quantifier ($\forall$) and existential quantifier ($\exists$)) used to carve the arbitrariness and existence of bound variables; 2. atomic predicate formula that uses predicates to describe variable attributes or relationships between variables. The predicate formulas used to formalize the answer in this article do not include free variables, so the variables in this article specifically refer to bound variables [4], [10]. Because the semantic graph used to describe the answer in the knowledge base is constructed strictly according to the logic formula. Therefore, the semantic graph used to represent the answer can also be regarded as composed of these two parts.

In the ostis-systems, the sc-construction composed of sc-tuple, relation sc-node, role relation sc-node and sc-connector is used to describe logical connectives and quantifiers, atomic predicate formula or multiple atomic predicate formulas that satisfy conjunctive relation are contained in the sc-structure and connected with the corresponding sc-tuple, and these sc-elements together constitute the semantic graph used to represent the answer. The atomic predicate formula is described using various sc-constructions. In the semantic graph, all sc-tuples and sc-connectors form a tree, which completely describes the logical sequence between connectives and quantifiers in the predicate formula. Because the sc-structure containing the atomic predicate formula is connected to the corresponding sc-tuple, as long as the position of each sc-tuple and sc-structure in the semantic graph is determined, the position of each variable sc-node in the semantic graph can be determined. In order to determine the position of each variable sc-node in the semantic graph, this article proposes an approach to numbering each sc-tuple and sc-structure in the semantic graph according to a depth-first search strategy (DFS). The approach mainly includes the following steps:

1) starting from the root of the tree structure composed of sc-tuples, each sc-tuple node in the tree is numbered in turn according to the DFS strategy (the numbering sequence starts from 0). If some nodes in the tree have multiple child nodes, the sub-trees with these child nodes as the root are sequentially numbered according to the node priority specified below;
   - if the child nodes and the parent node constitute the semantic structure that expresses the implication relation, then the priority of the conditional node (this node is connected with the parent node using a role relation " if' ") is greater than the priority of the conclusion node. That is, the node representing the condition and the corresponding sub-tree priority are numbered according to the DFS strategy;
   - if a node has multiple child nodes, and there is a node representing negative connective in the child nodes, then the priority of this node is higher than other nodes. If there are multiple nodes representing negative connective in the child nodes, the priority between them is related to the height of their corresponding sub-tree, and the higher the height of the sub-tree, the greater the priority;
   - in the current version, for other situations, a node is randomly selected for numbering.
2) according to the numbering sequence of sc-tuple, each sc-tuple in the tree is traversed from small to large, and the sc-structure connected to the current sc-tuple is numbered while traversing (the numbering sequence starts from 1). If there are multiple sc-structures connected

to the same sc-tuple, the sc-structure will be numbered according to the priority specified below.

- if there are multiple sc-structures connected to sc-tuple that represents universal quantifier or existential quantifier, the sc-structure that only contains variables is numbered preferentially;
- if there are multiple sc-structures connected to the sc-tuple that represents the implication relation, the sc-structure representing the condition is numbered preferentially;
- in the current version, for other situations, the numbering is based on the number of elements contained in the sc-structure. The fewer the number of elements contained in the sc-structure, the numbering will be given priority.

Because the atomic predicate formula or the conjunctive formula of the atomic predicate formula is included in the sc-structure, once the position of the sc-structure in the semantic graph is determined, the position of each atomic predicate formula in the semantic graph can be determined indirectly. In answer verification, if the standard answer and the user answer are exactly equal, it means that the atomic predicate formulas with the same semantics between the answers have the same position in the semantic graph (That is, the numbering sequence of sc-structure is the same). In the ostis-systems, the atomic predicate formula is expressed using various sc-constructions, so this article will establish the mapping relationship of potential equivalent variable sc-node pairs between the answers according to the matching relationship of the sc-constructions in the same position between the answers [4], [8]. The establishment of mapping relationship of the potential equivalent variable sc-node pairs between answers mainly includes the following steps:

1) according to the numbering sequence of the sc-structure in the semantic graph, each time a sc-structure pair with the same number is found from the standard answer and the user answer;
2) according to the priority order (from high to low) of the various types of sc-constructions used to describe the atomic predicate formula, it is determined in turn whether the current sc-structure pair contains this type of sc-construction at the same time. If the current sc-structure pair contains this type of sc-construction at the same time, then, according to the matching relationship of each sc-element between the current sc-construction in the standard answer and the current sc-construction in the user answer, the mapping relationship of the potential equivalent variable sc-node pairs between the current sc-construction pair is established. The priority order of various types of sc-constructions, and the criteria for judging whether the same type of sc-constructions match are as follows:
   - because there may be multiple sc-constructions in the same sc-structure, in order to ensure the uniqueness and accuracy of the mapping relationship of the potential equivalent variables sc-node pairs between sc-constructions, this article proposes to establish the mapping relationship between variables sc-nodes according to the priority order of sc-constructions. There are 14 types of sc-constructions in the current version, and the order of priority between them is determined by the number of sc-nodes contained in the sc-construction. The greater the number of sc-nodes, the higher the priority. If the number of sc-nodes is the same, it is determined according to the number of variable sc-nodes, the greater the number of variable sc-nodes, the higher the priority;
   - the criteria for judging the matching of the same type of sc-constructions are: 1. the constant sc-node at the corresponding position between them exactly matches; 2. there is a mapping relationship between the variable sc-nodes of the corresponding position, or there is no mapping relationship between the corresponding position variable sc-nodes, and there is no mapping relationship between these variables sc-nodes and other variables sc-nodes. If any pair of sc-constructions of the same type in the same position between the answers satisfy the above two conditions at the same time, the mapping relationship between the corresponding position variables sc-nodes between the sc-constructions is established (if there is already a mapping relationship between the two variable sc-nodes, it will not be created repeatedly).
3) repeat step 1 — step 2 until all potential equivalent variable sc-node pairs between semantic graphs have established a mapping relationship.

Fig. 3 shows an example of establishing the mapping relationship of potential equivalent variable sc-node pairs between semantic graphs according to the numbering order of sc-structures in SCg-code.

In Fig. 3, the definition of the inclusion relation is described in the form of a semantic graph $(\forall A \forall B(A \subseteq B) \Longleftrightarrow (\forall a(a \in A \rightarrow a \in B)))$.

When the mapping relationship between the potential equivalent variable sc-node pairs between the semantic graphs is established according to the positions of sc-tuples and sc-structures in the semantic graphs, the similarity between the answers can be calculated using formulas (1), (2), and (3). The criteria for judging the matching of substructures are: 1. the constant sc-nodes in the corresponding position between substructures have the same main identifier in the knowledge base or the same number in the semantic graphs (sc-tuple and sc-structure); 2. there is a mapping relationship between the variable sc-nodes at the corresponding position between the substructures.

Next, we will introduce the working algorithm of this sc-agent in detail:

Algorithm 3 — The working algorithm of sc-agent for computing semantic similarity of logical knowledge

Input: The specific subjective question and the corresponding semantic graph of standard answer and the semantic graph of user answer. The condition of the sc-agent response is that two semantic graphs that use logical knowledge to represent the answer appear in the sc-memory, and the similarity between them has not been calculated.

Output: The precision, recall and similarity between answers, and the sc-node used to record the matching status of substructures.

1) checking whether all input parameters used for sc-agent work meet the conditions, if so, go to step 2), otherwise,go to step 12);
2) checking whether the current question has multiple standard answers, if so, automatically select a standard answer that best matches the user answer according to the approach introduced earlier;
3) according to the rules of logical knowledge representation, the semantic graphs of standard answers and user answers are decomposed into substructures;
4) the sc-tuples and sc-structures in the semantic graph of the standard answer and the semantic graph of the

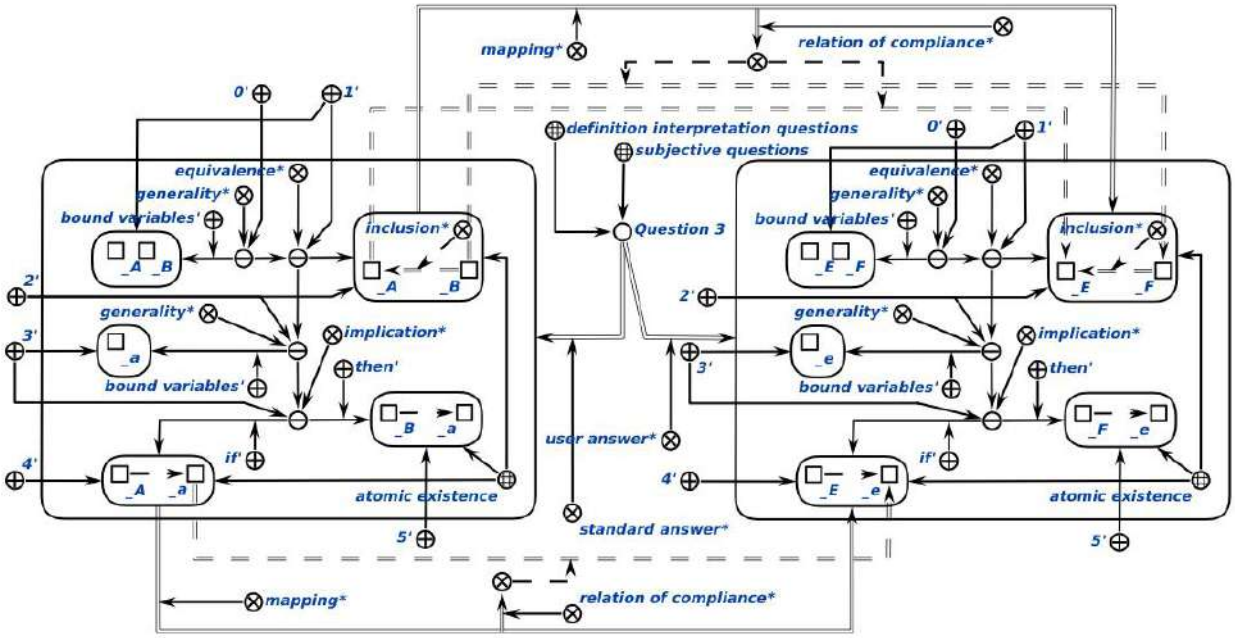

Figure 3. An example of establishing the mapping relationship of potential equivalent variable sc-node pairs between semantic graphs

user answer are numbered respectively, and the mapping relationship of potential equivalent variable sc-node pairs between answers is established;

5) iteratively traverse each substructure of the standard answer and user answer, classify them according to the type of substructure, and count the number of all substructures;
6) one type of substructure is randomly selected from the set of recorded standard answer substructure types;
7) according to the standard answer substructure type selected in step 6), a corresponding type of substructure is selected from the set of recorded user answer substructure types;
8) iteratively compare each substructure with the same substructure type between the standard answer and the user answer, and record the number of matched substructures and the matched substructures.
9) repeat step 6 — step 8 until all types of substructures have been traversed;
10) using formulas (1), (2), (3) calculate precision, recall and similarity, and generate semantic fragments for recording sc-agent running results;
11) removing all temporary sc-elements created while the sc-agent is running;
12) exit the program.

When the precision, recall and similarity between the answers to the subjective questions are obtained, the completeness and correctness of the user answers can be evaluated. According to the similarity, user answers are divided into the following situations:

- if the similarity is equal to 1, the user answer is completely correct ($F_{sc} = 1$);
- if the similarity is greater than 0 and less than 1 ($0 < F_{sc} < 1$), there may be two situations:
  - the user answer is partially correct and incomplete (the default mode of the current version);
  - the logical formulas used to formalize standard answers and user answers may satisfy logical equivalence (in the following article, we will introduce in detail the approach to judging the equivalence between answers based on predicate logic).
- if the similarity is equal to 0 ($F_{sc} = 0$), the user answer is completely wrong.

V. Conclusion and further work

Automatic answer verification is one of the most basic functions of ITS, which can quickly check the user mastery of new knowledge and improve the user learning efficiency. Therefore, this article introduces in detail an approach to developing a problem solver for automatic answer verification in the ITS developed using OSTIS Technology. The developed problem solver can not only verify the correctness and completeness of the answer to the subjective question, but also the correctness and completeness of the answer to the objective question. Because the problem solver is developed based on multi-agent technology, sc-agent for computing semantic similarity of factual knowledge, sc-agent for processing semantic similarity calculation results of factual knowledge, and sc-agent for computing semantic similarity of logical knowledge are developed in this article respectively. The developed problem solver completes the answer verification by combining different sc-agents according to the type of the question.

The developed problem solver for automatic answer verification in this article has the following advantages:

- verify the correctness and completeness of user answers based on semantics;
- by calling different sc-agents, the similarity between any two semantic fragments in the knowledge base can be calculated;
- because the problem solver is developed based on multi-agent technology, it is easy to add new functions;
- because the ostis-systems developed for different application fields have the same knowledge representation

approach and knowledge processing model, the problem solver developed in this article can be easily transplanted to other ostis-systems;

In future work, we hope to automatically generate some comments and suggestions by analyzing the verification results of user answers.

ACKNOWLEDGMENT

The work in this article was done with the support of research teams of the Department of Intelligent Information Technologies of Belarusian State University of Informatics and Radioelectronics. Authors would like to thank every researcher in the Department of Intelligent Information Technologies.

REFERENCES

[1] V. V. Golenkov and N. A. Guljakina, "Proekt otkrytoj semanticheskoj tehnologii komponentnogo proektirovanija intellektual'nyh sistem. chast' 1: Principy sozdanija project of open semantic technology for component design of intelligent systems. part 1: Creation principles]," Ontologija proektirovanija [Ontology of design], no. 1, pp. 42–64, 2014.

[2] V. Golenkov, N. Guliakina, I. Davydenko, and A. Eremeev, "Methods and tools for ensuring compatibility of computer systems," in Otkrytye semanticheskie tehnologii proektirovanija intellektual'nyh sistem [Open semantic technologies for intelligent systems], V. Golenkov, Ed., BSUIR. Minsk , BSUIR, 2019, pp. 25–52.

[3] Xu G. P., Zeng W. H., Huang C. L. Research on intelligent tutoring system. Application research of computers, 2009, Vol. 26(11), pp. 4020-4030.

[4] Li W., Grakova N., Qian L. Ontological Approach for Question Generation and Knowledge Control. Communications in Computer and Information Science, 2020, Vol. 1282, pp. 161-175.

[5] Qian L., Sadouski M., Li W. Ontological Approach for Chinese Language Interface Design. Communications in Computer and Information Science, 2020, Vol. 1282, pp. 146-160.

[6] (2021, JAN) Ims.ostis metasystem. [Online]. Available: https://ims.ostis.net

[7] Li X. J. Realization of automatic scoring algorithm for subjective questions based on artificial intelligence. Journal of Jiangnan University (Natural Science Edition), 2009, Vol. 08(03), pp. 292-295.

[8] Peter A., Basura F., Mark J. SPICE: Semantic Propositional Image Caption Evaluation. Computer Vision and Pattern Recognition (cs.CV), 2016.

[9] Zhang Z. P., Zhao H. L., Tian S. X. Ontology integration method based on RDFS. Computer Engineering and Applications, 2008, Vol. 44(15), pp. 131-141.

[10] Pan M. Q., Ding Z. J. A Simple Method for Solving Pyrenex Disjunction(Conjunction) Normal Forms. Computer Engineering and Science, 2013, Vol. 30(10), pp. 80-84.

[11] Wan H. R., Yang Y. H., Deng F. Review on Research Progress of Text Similarity Calculation. Journal of Beijing Information Science (Technology University), 2019, Vol. 34(01), pp. 68-74.

[12] Zhu J. Z., Qiao J. Z., Lin S. K. Entity Alignment Algorithm for Knowledge Graph of Representation Learning. Journal of Northeastern University (Natural Science), 2018, Vol. 11(39), pp. 1535-1539.

[13] Socher R., Chen D., Manning C. D., et al. Reasoning with neural tensor networks for knowledge base completion. Advances in Neural Information Processing Systems, 2013, pp. 926-934.

[14] Su J. L., Wang Y. Z., Jin X. L., et al. Knowledge Graph Entity Alignment with Semantic and Structural Information. Journal of Shanxi University(Nat. Sci. Ed.), 2018, Vol. 42(1), pp. 23-30.

[15] Zhuang Y., Li G. L., Feng J. H. A Survey Entity Alignment of Knowledge Base. Journal of Computer Research and Development, 2016, Vol. 53(1), pp. 165-192.

[16] Wang X. Y., Hu Z. W., Bai R. J., et al. Review on Concepts, Processes, Tools and Methods Ontology Integration. Library and Information Service, 2011, Vol. 55(16), pp. 119-125.

Разработка решателя задач для автоматической проверки ответов в интеллектуальных обучающих системах

Ли Вэньцзу, Цянь Лунвэй

В данной работе предложен подход к разработке решателя задач для автоматической проверки ответов в интеллектуальных обучающих системах, построенных с использованием Технологии OSTIS. Решатель задач разработан на основе многоагентного подхода к обработке информации. Разработанный решатель задач автоматически проверяет правильность и полноту ответа пользователя на семантическом уровне, используя соответствующие наборы sc-агентов в соответствии с типом вопросов (вопросы на выбор, вопросы на заполнение пробелов, вопросы на толкование определений и т. д.). В ситуации, когда может существовать несколько стандартных ответов (например, для вопросов на толкование определений), решатель задач может автоматически заранее отфильтровать фрагмент стандартного ответа, который наилучшим образом соответствует ответу пользователя, а затем продолжить проверку правильности и полноты между ними.

Keywords—решатель задач, проверка ответов, технология OSTIS, интеллектуальные обучающие системы, онтология, база знаний

С развитием современных информационных технологий искусственный интеллект как важная часть современных компьютерных приложений быстро применяется в сфере образования. Применение технологий искусственного интеллекта в сфере образования может не только предоставить новые методы и инструменты обучения, но и позволить учащимся тратить меньше времени на приобретение более полезных знаний, эффективно повышая точность поиска информации и эффективность отбора знаний. В то же время сочетание технологий искусственного интеллекта и образовательного процесса также является одним из важных средств обеспечения справедливости образования, чтобы каждый учащийся имел равные возможности для получения знаний. Особенно после вспышки COVID-19 интеллектуальные обучающие системы (ИОС) играют все более важную роль в дистанционном образовании. По мере того как разрабатывается все больше и больше ИОС, используемых в различных областях, большинство разработчиков считают, что ИОС необходимо удовлетворять, по крайней мере, следующим основным функциям:

- автоматическая проверка ответов, если обнаружена ошибка, необходимо проанализировать причину ошибки и принять корректирующие меры;
- в соответствии с уровнем обучающегося и ситуацией обучения автоматически корректируется содержание и прогресс обучения;
- автоматическая генерация различных вопросов и упражнений;
- обладая способностью генерировать и понимать естественный язык, а также осуществлять относительно свободный человеко-машинный разговор;
- наличие способности объяснять содержание обучения;
- автоматическое решение вопросов на основе понимания содержания обучения.

Являясь наиболее основной и важной функцией ИОС, автоматическая проверка ответов может быстро проверить усвоение пользователем новых знаний и значительно повысить эффективность обучения пользователя, позволяя пользователям получить больше знаний за ограниченное время.

Received 01.06.21