Received 1 May 2022, accepted 17 June 2022, date of publication 22 June 2022, date of current version 29 June 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3185400

CQACD: A Concept Question-Answering System for Intelligent Tutoring Using a Domain Ontology With Rich Semantics

YU WEN[1], XINHUA ZHU[2], AND LANFANG ZHANG[3]
[1]Student Work Department, Guilin Tourism University, Guilin 541006, China
[2]School of Computer Science and Engineering, Guangxi Normal University, Guilin 541004, China
[3]Faculty of Education, Guangxi Normal University, Guilin 541004, China

Corresponding author: Xinhua Zhu (2715163927@qq.com)

This work was supported in part by the National Natural Science Foundation of China under Contract 62062012 and Contract 61967003, in part by the Natural Science Foundation of Guangxi of China under Contract 2020GXNSFAA159082, and in part by the National Social Science Fund of China under Contract 17BTY063.

ABSTRACT In this study, a Concept Question Answering system applied to the Computer Domain (CQACD) for intelligent tutoring is proposed. This system is a dialogue-based Intelligent Tutoring System (ITS) that allows the tutor and student with mixed-initiative and natural language to ask each other questions concerning the basic computer knowledge in the *Computer Basics* course. CQACD is based on constructivist principles and encourages the learner to construct knowledge rather than merely receiving knowledge, which has the following characteristics: (a) this system employs a domain ontology with rich semantic relationships to model the basic computer knowledge and build up a concept-centric knowledge model, (b) uses a limited number of 80 input templates with description logics to acquire the intention of questions posed by students, (c) a textual entailment algorithm with semantic technologies is proposed to match the input template and assess the student's contribution to improve the flexibility of the system, and (d) an ontology-driven dialogue management mechanism is proposed, which can quickly form the conversational content and conversational sequence. The experimental results show that CQACD can replace the teachers' tutoring in large classes and can promote the learning of poor students in large classes better than teachers can. The paper reveals that the domain ontology with rich semantic relationships plays an important role in the Concept Question Answer System (CQAS). It can model CQAS's discipline knowledge, provide structured domain knowledge for student model, template design and matching, and provide basic architectural architecture for dialogue management.

INDEX TERMS Dialogue-based ITSs, domain ontology, question-answering system, template logics, ontology learning, NLP, answer reasoning, dialogue management.

I. INTRODUCTION

As an important research field in natural language processing, Question-Answering (QA) systems are advanced Information Retrieval Systems (IRSs) that can answer questions formulated by users in natural language form. Based on the domains to which they are oriented, QA systems can be classified into two types [1]: open domain-oriented systems and restricted domain-oriented QA systems. For open domain-oriented QA systems, no domain restriction is placed on topics, and these systems can accept questions of topics in any domain. While restricted domain-oriented QA systems can accept questions only about topics in a certain domain. Moreover, based on the answer type, the QA systems in IRSs can further divided into two other [2]: Named Entity (NE)-based systems and Common Noun (CN)-based QA systems, of which NE-based QA systems are the most common. NE-based QA systems extract potential answers from the corpora using named entity recognizers, whereas CN-based QA systems are relatively special in that the answers that they provide are common nouns in the corpora instead of named entities. In fact, in domain-oriented intelligent retrieval systems and ontology-driven Intelligent Tutoring Systems (ITSs), there exists also another type of Ontology Element (OE)-based QA systems [3]–[7], in which the expected answers originate from concepts, properties, relations or axioms in

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

the domain ontology. Ontology is a sharable knowledge base template based on a structured Frame and Semantic Network with Description Logics (DL) [8]. Therefore, ontology-driven QA systems have strong knowledge organization and logical reasoning capabilities, and OE-based QA systems are particularly suitable for the development of dialogue-based ITSs that might require integrating various types of resources for personalized learning [9]–[12] and providing reasoning-based answers for intelligent tutoring [6], [13].

With the rise of online education, various MOOCs have emerged, moving more and more courses from physical classrooms to online, and the COVID-19 pandemic has exacerbated the process. In online teaching, a prominent contradiction is: there is a large number of registered students, but the tutor is insufficient. Therefore, developing more question-answer ITSs becomes a very urgent task. The *Computer Basics* course, which teaches about the basic theory and principle of computer science including the basic knowledge in the composition structure and working principle of computers, computer applications, operating systems, information security and computer networks, is a compulsory general course for non-computer major students in most of colleges and universities in China. A common problem that is associated with this course is the large number of registered students and a lack of relevant teachers. To address this problem, a number of colleges and universities in China have allowed students to study this course by employing educational websites. However, to provide sufficient help to students to facilitate their learning remains a problem that needs to be urgently solved. To solve this difficulty, we have designed CQACD, a concept QA system, for this course. CQACD, which is a restricted do-main-oriented QA system, can accept questions that concern the basic and theoretical knowledge of computer science posed by students in the natural language and can guide them through tutor-initiative dialogues to learn and construct the basic computer knowledge in the *Computer Basics* course. We have designed CQACD in two language versions (English and Chinese). For ease of reading and understanding, only the English version of CQACD is discussed in this paper. The main contributions of this paper can be summarized as follows:

(1) We abstract the text corpora in the conventional dialogue tutoring system to the relationship and attribute restrictions and axioms of concepts in a domain ontology and build up a concept-centric knowledge model to provide powerful semantic support for natural language processing and answer reasoning in the system, which improve the subject knowledge model in in the conventional dialogue-based ITSs from the curriculum script [21], [28], [29] to structured domain ontology;

(2) We design a template description logic system with domain ontology elements as operands to define the input templates and tag ontology's attribute restrictions so as to accurately represent their semantics and significantly reduce the number of question templates in QA systems from 27, 126, 355 [34] to less than 100;

(3) We employ semantic technologies including mandatory word annotation, ontology element type analysis, word similarity and tone similarity to improve the flexibility and accuracy of Lexical-based Textual Entailment (TE) algorithms [41] for template matching and contribution assessing by more than 20%;

(4) We propose an ontology-driven dialogue management mechanism based on the scaffold instruction theory of the zone of proximal development [14], which improves the design of the conversational content and conversational sequence from the hand-crafted mode [24] to the ontology-based automatic generation.

II. RELATED STUDIES

A QA system generally includes three modules: question pre-processing and classification, information retrieval and answer extraction. In a QA system, question pre-processing and classification is the first execution phase and comprises a crucial step with the objective to enable a computer to understand and recognize a user's question and determine the user's intention in asking the question so as to provide clues for the subsequent information retrieval and answer extraction steps. In the pre-processing phase, users' questions are processed by a procedure that generally involves lexical analysis, syntactic parsing and semantic analysis. Question classification is the process of determining to which type the user's question belongs in the predefined question patterns. Two methods exist to classify questions. In open domain-oriented QA systems, users' questions are automatically classified by machine learning methods, including the SVM (Support Vector Machine) classification [15] and neural network classification methods [16], in which the semantic characterizations of questions are usually represented by named entity recognition [17] and semantic role labeling [18]. In restricted domain-oriented QA systems, users' questions can be processed and classified by ontology-based triples [4], [6] or ontology-based language templates (formulae) for questions [19], [20], in which question semantics are interpreted by the ontology. In the information retrieval step, an open domain-oriented QA system primarily acquires candidate answers from documents in websites or a large corpus based on the question's semantics and its answer type and then selects the best answer using the answer extraction module [47], whereas a restricted domain-oriented QA system usually completes answer retrieval, reasoning and extraction by the domain ontology.

A. DIALOGUE-BASED-ITSS

Dialogue-Based ITSs (DBITSs) is a subset of ITSs that emulate the natural language dialogues in human tutoring [21]. Based on the way of dialogue, Domeshek [22] classified DBITSs into two types, namely, tutor-initiative DBITSs and student-initiative DBITSs. Tutor-initiative DBITSs that adhere to constructivist principles attempt to get learners to do most of the talking by providing questions with hints or prompts, forced choices and other pedagogical scaffolds [21], such as DIALOG project in ΩMEGA [3], Geometry Explanation Tutor [23], Why2-ATLAS [24] and BEETLE II [6].These computer tutors vary in the extent to which they simulate human dialog mechanisms, but all of them attempt to comprehend natural language, formulate adaptive responses and implement pedagogical strategies to facilitate learning. In contrast from tutor-initiative DBITSs, the objective of student-initiative DBITSs is to provide a convenient man–machine interface for students to ask questions to the system during their learning process instead of facilitating learning by pedagogical strategies. To enhance students' initiative in their knowledge construction, many DBITSs provide the student-initiative dialogue function, such as the SCHOLAR system [25], CIRCSIM-Tutor [26], ATLAS [27], AutoTutor [28] and Conversational-Tutoring-Agent [29]. In today's mobile Internet era, where knowledge explosion occurs, student-initiative DBITSs are significant in helping students to acquire the knowledge they are interested in. The concept QA system CQACD in this study is a tutor and student mixed-initiative DBITS.

At present, the most prominent deficiency of DBITSs is that systems lack a rich semantic and highly structured domain ontology to model subject knowledge, and some DBITSs [21], [28], [29] even use course scripts to express subject knowledge, which cannot provide effective support for template design, knowledge management, knowledge inference, building dialogue order and student model.

B. QUESTION TEMPLATES IN QA SYSTEMS

Question templates [19] in QA systems, which are also referred to as question formulae [20] or question patterns [2], [30], [31], are a question processing mechanism that is widely employed in question clustering and classification. The main goal of question templates is to map the problem pattern into the answer pattern. Based on the number of their semantic components, we classify question templates into two types, namely, Simple Question (SQ) templates and Language Question (LQ) templates. SQ templates refer to question formulae that contain only the core semantics of users' questions. For example, in question patterns adopted by Ray *et al.* [30], the simple template "when questions" is the abstraction and representation of all questions that are similar to "when is the operating system loaded?". SQ templates directly contain only a relatively small amount of semantics. To achieve fine classification of a question, additional semantic characteristics of questions need to be extracted by the syntactic parsing and semantic analysis [15], [18], [32] and a machine learning method, such as an SVM method [15], [32], need to be employed. SQ template-based QA systems are highly automated, eliminating the need to manually build large-scale question templates, but they have a relatively complex question classification process that is significantly dependent on the syntactic parsing and semantic analysis accuracies and only moderately classification precision because of the widespread existence of semantic ambiguities [33]. Therefore, SQ templates are generally employed in open domain-oriented QA systems that can't pre-establish a complete question template library.

LQ templates refer to question formulae that contain the entire sentence structure and all semantics of users' questions [5], [19], [20]. For example, in the QACID system designed by Ferrández *et al.* [20], the LQ template "where can I see [MOVIE]?" is the abstraction and representation of all questions that are similar to "where can I see Saw 3?". Because a LQ template already contains the entire sentence structure and all semantics of a user's question, the question does not have to be parsed and can be directly classified using a textual similarity or TE-based template-matching algorithm [5], [19]. To reduce the size of the template library, LQ templates often represent and interpret formulate the core semantics of questions with ontology element symbols. For example, in the QACID system [20], the LQ template "where can I see [MOVIE]?" represents all movie entities with the ontology concept *MOVIE*. An ontology usually contains a large number of concepts and relationships. Consequently, such ontology element-based LQ templates still have these shortcomings as insufficient representational capacity and overly large template libraries [2], [6], [19], [20], [34]. For example, in the template library adopted by Cuix *et al.* [34], 27, 126, 355 LQ templates with concept names are used to process CN-based question classification. To improve the universality of question templates and further reduce the size of the template library, a first-order template description logic system with domain ontology elements as operands that can ensure decidable reasoning is introduced in this study. This logic system can use a first-order predicate to abstract the same ontology element into a variable in the question template, e.g., our template "What is the <p: stringProperty> of <c: Concept>?" is the abstraction and representation of all questions that ask any a string property of any a concept in the ontology.

In this study, LQ templates carry two functions of acquiring the intention of questions posed by students and formulating questions of the tutor to students and were expanded into input templates that include Meta COMmunication (Meta-COM) templates and question templates. After introducing description logics in the template, only 80 input templates are required to satisfy all the requirements of the students' interactions and asking questions.

III. SYSTEM OVERVIEW

The concept QA system CQACD designed and implemented in this study is a mixed (both tutor and student) initiative DBITS that concerns the basic knowledge of computer science. CQACD can play two roles in the *Computer Basics* course: an intelligent answering system to answer questions about basic computer knowledge posed by students in natural language form and a dialogue-based ITS for guiding students to learn and construct basic computer knowledge. CQACD is a typical knowledge-based system because it contains a domain ontology, a template library, a semantic dictionary and a student model library. Under the support of these knowledge bases, the mixed-initiative dialogue in CQACD

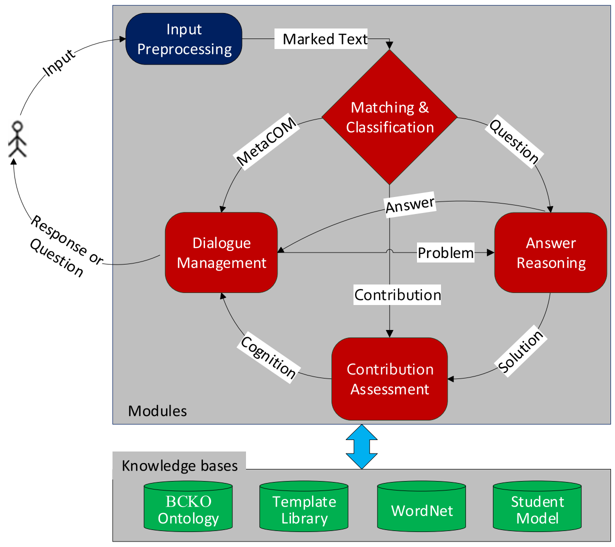


FIGURE 1. **CQACD system architecture.**

is implemented by five functional modules, as shown in Figure 1. This chapter gives a brief introduction to each part of the CQACD system. They are described in more detail in subsequent chapters. Table 1 gives an example of the whole QA process.

A. BASIC COMPUTER KNOWLEDGE ONTOLOGY

Our CQACD system uses a domain ontology of Basic Computer Knowledge Ontology (BCKO) with rich semantic relations to formalize and manage all subject knowledge in the basic areas of computer science and build up a concept-centric knowledge model. This domain ontology is the core knowledge base in CQACD system, which plays the following roles: (a) provides all the subject knowledge for the QA and dialogue-based tutoring in CQACD, (b) pro-vides a glossary for the annotation of the ontology elements in the input preprocessing and the definition of question templates, (c) provides the dialogue order for the dialogue management through its concept hierarchy, (d) provides complete cognitive units for the student model, and (e) provides the basis for answer reasoning through its semantic relations and concept axioms.

B. TEMPLATE LIBRARY

Our template library is a collection of 80 input templates:

$$Templatelibrary = \{MetaCOMtemplates\} \cup \{Questiontemplates\} \qquad (1)$$

where MetaCOM templates are used to classify students' metacognitive inputs (e.g. "I need help", "I don't know") and navigation inputs (e.g. "I want to learn computer hardware"). Question templates are ontology and logic-based question formulas to classify students' question inputs; they reflect students' interests and needs to the basic computer knowledge. In addition, the question template in the template library can also serve as a language template for the tutor's asking questions. Each input template is composed of five bound parts: template category, template structure, synonymous structure, reasoning rule and reasoning function.

TABLE 1. **Example of the whole QA process.**

1 Informal student question *Besides RAM, what does Memory include?*
2 Lemmatization and Morphological Analysis besides/IN RAM/NNP , what/WP do/VBP memory/NN include /VB ?
3 Tagging the question with ontology elements besides/IN <**RAM**: Concept >/NNP , what/WP do/VBP <**memory**: Concept>/NN include /VB ?
4 Template returned by the matching and classification module <Besides> <*s*1: ConceptSet>, <what> [else] <do> <*c*1: Concept > <include><?>
5 Incorporating the original input entities into the template <Besides> <*s*1: {**RAM**}>, <what> [else] <do> <*c*1: **memory**> <include><?>
6 Answer returned by the corresponding reasoning function **ROM**, **Cache**

C. WORDNET

WordNet is a large synonyms semantic dictionary based on cognitive linguistics and is designed and realized by psychologists, linguisticians and computer engineers in Princeton University [35]. It can be widely used for text classification [45] and semantic similarity calculation [46]. WordNet, in our CQACD system, is used to provide more semantic interpretation for student inputs based on BCKO ontology. The introduction of WordNet in CQACD significantly improves the accuracy of template matching and contribution assessments.

D. STUDENT MODEL

Student model is used to track students' learning status (how much/how well has been learned) and is an overlay of the domain ontology in CQACD system. Our CQACD system is a concept-based ITS that guides students through dialogues to gradually grasp the conceptual knowledge in basic computer areas. Therefore, it does not require the complex Bayesian knowledge tracing [36], [37] for student model, which is often used to build student models for skill-based domains like mathematics. We directly use concept maps [38], [39], which are widely used in the ontology-driven ITSs [11], [12], to build the student model in CQACD as follows:

$$\text{Student\_model} = \{\text{score}(c) | c \in \text{BCKO}\} \qquad (2)$$

where score(c) represents the confidence value that the student knows the concept *c*, which is measured by an improved TE algorithm with semantic technologies.

E. INPUT PREPROCESSING

Input preprocessing is a process of the pronoun elimination, lemmatization and morphological analysis for the student's input, in which the student's input is eventually marked with part-of-speech (POS) and ontology elements, as shown in Table 2. We use Stanford CoreNLP[1] to implement the lemmatization and POS tagging for student's inputs. Ontology element tagging is the process of successively marking the concepts, properties, relationships and concept sets of the domain ontology that appear in the student's input. In the ontology element identification and tagging process, fuzzy recognition is employed by our extended tagging software, which allows errors (individual misspelled letters) in the student's spelling of ontology elements.

F. MATCHING AND CLASSIFICATION

Matching and classification is used to classify the marked student input by matching it with input templates in the template library as follows:

$$\mathrm{MC(MI)} = \begin{cases} \textit{a MetaCOM} & \text{if matching} \\ \textit{a question} & \text{if matching} \\ \textit{a contribution} & \text{else if responsive} \\ \textit{a exception} & \text{else if non\_responsive} \end{cases} \tag{3}$$

where M&C(MI) represents matching and classifying the marked student input MI.

In our CQACD system, student inputs are divided into four types: MetaCOMs, questions, potential contributions to questions posed by the tutor and exceptions. MetaCOMs and questions posed by the student are identified and classified by the corresponding input templates. In this study, a improve TE algorithm combined with ontology element tagging and semantic similarities is employed to determine whether the student's input matches a template. If a relatively high degree of entailment exists between a student's input and a certain template in the template library that exceeds the threshold value, the student's input is considered to match the input template. Contributions and exceptions are classified by our proposed environment sensitive algorithm, in which in the responsive status after tutor's posing a question, the unmatched input is processed as potential contribution to the answer while in non-responsive status the unmatched input is processed as an exception.

G. ANSWER REASONING

Based on the semantic clues in a question and the corresponding template, answer reasoning module provides the answer or solution, which is extracted from the domain ontology BCKO, for the question or problem posed by the student or the tutor. Answer reasoning based on the domain ontology is done by calling the corresponding reasoning function bound with the question template.

H. CONTRIBUTION ASSESSMENT

Based on the solutions extracted from the domain ontology BCKO, contribution assessment module employs a proposed TE algorithm with semantic technologies to assess the student's contribution (response) to the question posed by the tutor and to determine whether the student's contribution is correct, error or a misconception. Moreover, this module is also responsible for updating the student model with the evaluation results.

[1]http://nlp.stanford.edu/software/corenlp.shtml

TABLE 2. Algorithm 1. Input preprocessing.

```
Algorithm 1.  Input preprocessing
input: student input  $I$
  1: LI= lemmatization ($I$)
  2: POSI=POS_tagging (LI)
  3: MI= Ontology_element_tagging (POSI)
     /*  First, the concepts are marked. The noun-centric phrases in the
     student's input are one-by-one matched with the concept synonyms
     in the domain ontology. If a concept exists that is the same as a
     noun-centric phrase, this phrase is tagged as an ontology concept in
     the following format: <Concept name: Concept>, e.g. <memory:
     Concept>. Second, according to the above method, the properties are
     tagged based on the remaining nouns in the student's input. Third,
     the relationships are tagged based on verbs and the remaining nouns
     in the student's input. Finally, the concept sets are tagged. Multiple
     concept names linked by conjunctions or punctuations are combined
     to form a concept set. */
  4: MI=  pronoun_elimination (MKI)
     /* The pronouns in inputs are replaced by the current learning
     concept or a phrase associated with the current concept through our
     proposed disambiguation algorithm based on the current dialogue
     topic. */
Output:  marked student input MI
```

I. DIALOGUE MANAGEMENT

Dialogue management is the most important and complex module of the system. It is responsible for the following personalized dialog functions: (a) to respond to the student's input, including positive encouragement for the correct contribution, the corresponding prompt for the wrong contribution or a misconception and the answer to the student's question; (b) to organize dialogue branches and the dialogue order according to the student's response and ontology's concept hierarchy; (c) to provide the student with help or learning contents in accordance with the current dialogue topic and the student's cognitive level.

IV. DOMAIN ONTOLOGY DESIGN

In this study, our CQACD system replaces the curriculum script in the conventional dialogue-based ITSs [21], [28], [29] using a domain ontology BCKO with rich semantic relations, which can bring significant advantage and convenience in semantic annotation and semantic analysis, knowledge management, knowledge inference, building dialogue order and student model, etc. BCKO ontology consists of six parts: concepts, attributes, relations, synonyms, a concept hierarchy and axioms.

A concept refers to a collection or abstraction of entities with the same characteristics in the basic areas of computer science. An attribute is a binary relationship between a concept and a data object that is mainly used to reflect a certain characteristic of the concept, such as definition, feature, etc. A relation is a binary relationship between concepts, such as the relations store and display in the

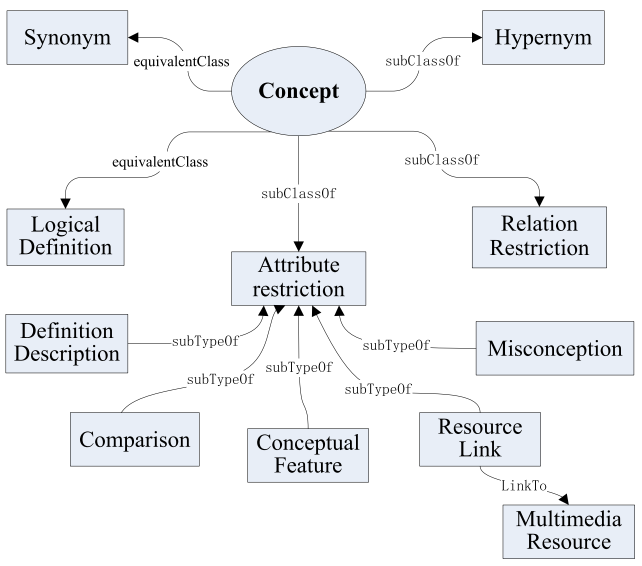


FIGURE 2. A concept-centric knowledge structure in BCKO.

computer field. Synonyms refer to the set of the concept, attribute and relation names that have the same meaning. All synonymous collections of concepts, attributes and relations constitutes the term set in the basic areas of computer science, which can provide the vocabulary for the recognition and annotation of the ontology element in stutents' inputs and question templates.

Concept hierarchy is a taxonomic structure composed of "is-a" relationships (*hypernymies*) between concepts, which reflects the inheritance relations between concepts in the computer field. It can provide important basis for the automatic reasoning of QA such as "Is *concept1* a type of *concept2*?".

An axiom is an assertion of permanent truth about domain knowledge, which is a type of theoretical knowledge that can be understood by the machine, as shown in Table 3. Axioms are used to declare conceptual knowledge and organize a concept-centric knowledge structure (as shown in Figure 2) for our CQACD system, in which axioms with inclusion operator ($\subseteq$) are used to declare the hypernym, attribute restrictions and relation restrictions of a concept in BCKO ontology, while axioms with equivalent operator ($\equiv$) are used to declare the synonyms and logical definitions of a given concept. The attribute and relation restrictions of concepts in BCKO ontology are important basis for the retrieval and reasoning of answers, which are an abstraction of attribute values or relation targets for all instances of the concept, that is, the same attribute value or relation target of all instances in a concept are extracted as an attribute or relation restriction of the concept. For example, in our BCKO ontology, the value of the *definition* attribute of the *memory* concept can be constrained as "the memory used to store the data" and the target of the *storage* relation of *memory* concept can be constrained as the *data* concept. There are two restraint ways respectively formed by the universal quantifier ($\forall$) and the existential quantifier ($\exists$).

TABLE 3. Examples of axioms in BCKO ontology.

Concept	Axiom
Storage	$\mathrm{Storage} \subseteq \exists \mathrm{store}.(\mathrm{Data} \cup \mathrm{Progyams})$
CPU	$\mathrm{CPU} \subseteq \forall \mathrm{access}.\mathrm{Memory}$
Memory	$Memory \subseteq \exists definition.strimg1$
Memory	$Memory \subseteq \exists mis\_definition.strimg2$
Memory	$Memory \equiv Storage \cap \exists be\_access\_by.CPU$
Memory	$Memory \subseteq \exists has\_feature.string3$
Memory	$Memory \subseteq \exists difference\_from\_hard\_disk.strimg4$
Memory	$Memory \subseteq \exists has\_resouce\_link.URL1$

In this study, the universal quantifier ($\forall$) is interpreted as "can only" while the existential quantifier ($\exists$) is interpreted as "can". Logical definition axioms are usually used to get the machine to understand how concepts are defined, e.g. the logical definition axiom $Memory \equiv Storage \cap \exists accessed\_by.CPU$ tries to get the machine to understand "*Memory* is a type of *storage* that can be accessed by the *CPU*".

To extend the role of BCKO ontology in dialogue-based ITSs, we use some special attributes and attribute nomenclature in BCKO ontology. For example, we use the existential restriction of the attribute *has_resource_link* to point to a URL of the learning resource directory that contains multiple multimedia learning resources corresponding to different cognitive levels for a given concept. Moreover, we use the nomenclature of *mis_attribute-name* to represent the misconception attribute for a given attribute, e.g. the attribute *mis_definition* represents the misconception of the attribute *definition*; and use the nomenclature of *difference_from_concept-name* to represent a differential comparison attribute between concepts, e.g. the attribute *difference_from_hard_disk* represents the difference of a given concept with the concept *hard disk*. More importantly, we also use some special string tags and structures to bind some related information in attribute restrictions. For example, we use the string structure of <*misconception description* | *hint* | *remedial concepts*> to bind the information about an existential misconception restriction, and in Table 3, we define *string*2=<*Memory refers to the storage in the mainframe box* | *There may be a hard disk in the mainframe box besides memory* | *host, main box*>, which is the value of an existential restriction for the attribute *mis_definition* of the concept *memory*.

In this study, we use OWL[2] ontology model to build the BCKO domain ontology that includes the basic knowledge in the composition structure and working principle of computers, computer applications, operating system, information security and computer networks and contains 947 concepts, 66 relations, 55 attributes and over 30,000 axioms.

V. INPUT TEMPLATE DESIGN

A. INPUT TEMPLATE STRUCTURE

The input templates in this study are a type of domain ontology-based input formulae that bind the template category, template structure, synonymous structure, reasoning

[2]https://www.w3.org/TR/owl-guide/

TABLE 4. Template tags.

Tag	Description
< >	Define a mandatory element in the template
[ ]	Define an omissible element in the template
{ }	Define a collection of elements in the template

rule and rule's reasoning program, which form the mapping from students' inputs to their intentions or answers. Their Backus–Naur form is defined as follows:

<input template>::= (<template category>, <template structure>, {<synonymous structure>}, <reasoning rule>, < reasoning function >).

1) TEMPLATE CATEGORY

To facilitate ontology-based knowledge retrieval and reasoning, we decompose 18 templates that are identified by Graesser and Person [40] in educational settings, and build up 80 input templates based on students' interaction and question requirements, which are classified into seven categories based on the nature of the input and the asked ontological components: metacognitive input, navigation input, ask the data attribute restrictions of concepts, ask the relation (object attribute) restrictions of concepts, ask the definition domain of attribute restrictions, ask the hierarchical structure of concepts and ask the definitional axioms for concepts.

2) TEMPLATE STRUCTURE

Template structure refers to the language structure of an input template and its shallow semantics that are represented with variables and tags.

3) SYNONYMOUS STRUCTURE

Synonymous structure refers to a language structure with the same question semantics as the main template structure. One input template may contain multiple synonymous structures.

4) REASONING RULE

Reasoning rule represents the deep semantics of an input template and the reasoning process of the expected answer or intention of the student's input. The rule semantics is accurately represented by a domain ontology-based predicate formula.

5) REASONING FUNCTION

Reasoning function represents a template-bound program that performs the reasoning function specified by the reasoning rule. Unlike SPARQL inference programs that are suitable for querying the related information of named entities (instances) in an OWL ontology and used in NE-based QA systems [4], [7], [34], we employ Java reasoning programs based on Jena OWL Ontology API[3] to query axioms in BCKO ontology.

[3]http://jena.apache.org/download/index.cgi

B. TEMPLATE DESCRIPTION LOGICS

To accurately represent their semantics, a template description logic system (TDLS) is designed to define the input templates. This logic system is first-order description logics with domain ontology elements as operands and used for the semantic tagging and interpretation of input templates, which is defined as the following triplet:

TDLS::= (<predicate set>, <operators>, <tags>)

1) PREDICATES

Predicates are used to reveal, recognize and determine the ontology elements in an input template. In TDLS there exist two types of predicates: one-place predicates and two-place predicates. The one-place predicate is used to declare the category of the ontology element to which a template variable belongs. The two-place predicate is used to declare the semantic relationship between two template variables. One-place predicates can also be employed as variable type keywords in the template structure. For example, $< c_1$: Concept $>$ indicates that variable c_1 is an ontology concept, in which the predicate *Concept* is employed as an ontological element type.

2) OPERATORS

In this study, we employ the following two extended operations to improve the representational capacity of the template logic: a type constructor ":" that define an ontology element variable, e.g. "x : Concept" represents that the variable x is defined as an ontological concept; and a conditional operator "? : " used to organize the semantics of the branch structure, e.g. "a ? b : c" represents that if the condition *a* is satisfied then the expression *b* is evaluated otherwise c is calculated.

3) TAGS

Three template element tags are designed to separate and define various types of elements in templates, as shown in Table 4.

C. EXAMPLES OF TEMPLATE DEFINITION

In this section, we give three typical structural definitions of question templates in CQACD system to illustrate the role of the template description logic in the template definition, in which all words in the template are used in their original forms. Moreover, we also give a reasoning function bound with Question Template2 using Jena OWL Ontology API in Appendix A.

Question Template1: Query the character attributes of a concept

<Template category>::= Ask the data attribute restrictions of concepts

<Template structure>::=<what><be> [the] $<$ p : stringProperty$>$<of>$<$ c : Concept$>$<?>

<Synonymous structure>::=<what><do> [the] $<$ p : stringProperty$>$<of>$<$ c : Concept$>$<mean><?>

$$<\text{ReasonRule}>::= \forall c, p(Concept(c) \wedge stringProperty(p)$$
$$\wedge \exists s(String(s) \wedge subClass(c, \exists p.s) \rightarrow Answer(s)))$$

<Reasoning function>::= Reasoning_function1(c, p)

<Example>: *What is the definition of memory?*

Question Template2: Query a subclass collection of a given concept

<Template category>::= Ask the hierarchical structure of concepts

<Template structure>::=<Besides>< s_1: ConceptSet>, <what> [else] <do> < c_1 Concept ><include><?>

<Synonymous structure>::=<what> [else] <do> < c_1 Concept ><include><besides> < s_1: ConceptSet><?>

<ReasonRule>::= $\forall s_1, c_1(ConceptSet(s_1) \wedge Concept(c_1) \wedge \exists s_2(ConceptSet(s_2)$

$$\wedge \forall c_2 \in s_2(subClassOf(c_2, c_1) \wedge \forall c_3 \in s_1(c_2 \neq c_3)) \rightarrow Answer(s_2)))$$

<Reasoning function>::= Reasoning_function2(c_1, s_1)

<Example>*Besides RAM, what else does memory include?*

Question Template3: Query a relation restriction of a given concept

<Template category>::= Ask the relation restriction of concepts

<Template structure>::=<Can>< c_1: Concept> <only>< r: relation>< c_2: Concept ><?>

<ReasonRule>::= $\forall c_1, r, c_2(Concept(c_1) \wedge Concept(c_2) \wedge Relation(r)$

$$\wedge \exists a(a = subClass(c_1, \exists r.c_2)?\text{“yes”} : \text{“not”} \rightarrow Answer(a)))$$

<Reasoning function>::= Reasoning_function3(c_1, c_2)

<Example>*Can the CPU only access memory?*

VI. TEMPLATE MATCHING INFERENCES

[4] Template matching, which is an important step in the input classification, is the process of selecting an input template that matches a given student's input from the template library. To enhance the robustness of template matching, the calculation of the matching between a student's input and a template is considered to be a textual entailment solution-finding process with a focus on determining whether a student's input contains the key semantics of the template. We propose the following three semantic methods to improve the conventional textual entailment algorithms based on n-gram [20], [41], [42].

A. MANDATORY WORD-BASED MATCHING INFERENCE

We argue the key semantics of an input template is mainly characterized by the mandatory words that are marked by the delimiter “<>” in the template. Therefore, as long as a student's input contains the key semantics of an input template completely, we think that the student's input matches the template. In this study, mandatory word-based template matching inference conclusions are drawn by comprehensively considering the following two TE algorithms:

1) SIMPLE MATCHING

This simple matching algorithm does not consider word continuity. As long as a student's input contains all the mandatory words in an input template, this algorithm considers that the student's input matches the template in question, and the formula is:

$$spMatch(Inp, Temp) = \frac{\sum_{i \in T} Lmatch(i)}{|T|} \tag{4}$$

where *Inp* represents any one student input, *Temp* represents any an input template, T is a set of mandatory words included in the input template Temp and Lmatch(i) is calculated as follows:

$$Lmatch(i) = \begin{cases} 1 & \text{if} \exists j \in U j = i, \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

where U represents the set of words in the student input Inp.

2) CONTINUOUS SUBSEQUENCE MATCHING

This algorithm focuses on continuous subsequences of mandatory words and determines whether a student's input contains any possible continuous subsequences of mandatory words in a template, and the formula is:

$$LCSmatch(Inp, Temp) = \frac{\sum_{k=2}^{|T|} f(ST_k)}{|T| - 1} \tag{6}$$

where ST_k represents the set that contains all the contiguous subsequences whose length is k in the template *Temp*, and the formula computing $f(ST_k)$ is as follows:

$$f(ST_k) = \frac{\sum_{i \in ST_k} Smatch(i)}{|T| - k + 1} \tag{7}$$

$$Smatch(i) = \begin{cases} 1 & \text{if} \exists\, j \in SI_k j = i, \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

where SI_Krepresents the set of the contiguous subsequences with length k in the student input *Inp*.

B. ONTOLOGY-BASED MATCHING INFERENCE

This inference further considers the ontology element type based on mandatory word-based matching inference, that is, the ontology elements and their types in students' inputs are automatically marked out before matching, and for any one ontology element variable in the template, it is matched with an ontology element in the student's input as long as they have the same type in the matching computations of the formulae (5) and (8).

C. SEMANTIC SIMILARITY-BASED MATCHING INFERENCE

To improve matching flexibility, the semantic similarity is integrated into mandatory word-based matching inference process. Firstly, when determining whether two words match in Eq. (5) or Eq. (8), these two words are not directly

[4]The code is on https://github.com/wuhan-1222/CQACD

determined whether they are the same term but are instead analyzed whether they are near-synonyms whose similarity degree is greater than the threshold value STH (STH is set to 0.75 in this study) in WordNet. The improved formula of Eq. (5) or Eq. (8) is as follows:

$$Sem\_match(i) = \begin{cases} 1 & \text{if} \exists j \in P \text{sim}(j, \text{i}) > \text{STH} \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

where P represents the set or subset of words in the student input *Inp,* in which P is equal to U for Eq. (5) and equal to $\mathrm{SI_K}$ for Eq. (8). Sim (j, i) represents the similarity between words j and i in WordNet, it is computed as follows according to the part of speech of j:

$$Sim(j, i) = \begin{cases} 1 \text{if synonyms(j, i) else} \\ sim_{noun}(j, i) \text{ifj} \in \text{Nouns else} \\ 0.8 \text{ifj} \in \text{Verb}s\text{and pathLen(j, i)} \leq 2 \text{ else} \\ 0.9 \text{ifj} \in \text{Adjs} \cup \text{Advs and similar(j, i) else} \\ 0 \text{ otherwise.} \end{cases} \tag{10}$$

where synonyms (j, i) declares that the words j and i are synonyms in WordNet. $\mathrm{Sim_{noun}}$ (j, i) represents the similarity between nouns j and i in WordNet, we employ an efficient path computing model proposed by Li et al [46] for measuring the semantic similarity between nouns in WordNet. PathLen (j, i) represents the shortest path length between verbs j and i in WordNet. Similar (j, i) declares there exists a similar-to relation between the adjectives or adverbs j and i in WordNet.

In addition, to distinguish between positive and negative tones, we use Stanford NLP Lex-Parser[5] to analyze the dependency between words in the input and template sentences and define the following tone similarity between the input and template sentences:

$$ToneSim(Inp, Temp) = \begin{cases} 1 & \text{ifneg(Inp)=neg(Temp),} \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

where the function neg(s) determines whether there exists a neg dependency in the sentence s, and if existing, it returns 1, otherwise it returns 0. In Stanford NLP, neg dependency indicates there exists a negation modifier in the sentence.

Based on the analysis of student input corpus, we discovered that there was negative tone only in verification questions. In order to reduce the size of the template library, we did not design any negative templates in the template library. Therefore, when a student enters a negative sentence, only the template with the corresponding positive tone can be matched in the template library. At this time, ToneSim (Inp, Temp) in Eq. (11) is equal to 0, and the system performs the following processing: For a verification question with the answer "yes/no", the system returns the same answer with the corresponding positive input; for other questions, the system returns the answer "I don't know" to avoid erroneous answers and enhance the robustness of the system.

[5]https://nlp.stanford.edu/nlp/javadoc/javanlp/edu/stanford/nlp/parser/lexparser/package-summary.html

VII. CONTRIBUTION ASSESSMENT

The traditional ITSs based on the curriculum script usually employ latent semantic analysis (LSA) to evaluate free response questions [21], [43]. LSA for educational purposes use a vector space (also called semantic space) model, formed by the curriculum script, to calculate the similarity between the student response and the standard answer, so as to give the evaluation conclusions of the student's response. LSA solves the problem of response evaluation in the textual domain model without ontology, but its computing process is relatively complex and has only moderately evaluation accurate because it lacks the support of the semantic dictionary and the domain ontology. In this study, we use a domain ontology with rich semantics to model the subject knowledge in the system. Therefore, we employ proposed TE algorithms with semantic technologies to assess the student's contribution (response).

Firstly, we divide students' contributions (responses) to the questions posed by the tutor into the following two categories: (a) a short contribution corresponding to the question whose answer is a single word (e.g. "yes" or "no", "right" or "wrong"), a single value, and one or more unordered ontology elements; and (b) a long contribution corresponding to the question whose answer is a text related to the concept's definition or the attribute restriction. For short contributions, we use the formula (4) combined with WordNet-based word similarity and synonyms in the domain ontology to calculate the entailment degree of the student's contribution to the correct answer, so as to give a score for the student's contribution. For assessing long contributions, firstly we, in the process of building the domain ontology, use the same way of the input template to mark the mandatory words and ontology elements in the attribute restrictions and misconceptions of all the concepts; and then we comprehensively consider the formulae (4), (6) and (11) with WordNet-based word similarity and synonyms in the domain ontology to determine whether the student's contribution is correct, error or a misconception.

The student's contributions to a question may be given step by step in the dialogue process; the system can automatically accumulate them. When the accumulated contribution score exceeds a threshold of 0.9, the student's answer to the question is determined to be correct. If a student is able to answer the question correctly without any help, the student is judged to have mastered the knowledge point corresponding to the question and a corresponding update is done in the student model.

VIII. DIALOGUE MANAGEMENT DESIGN

Dialogue management is the most flexible and challenging module in dialogue-based ITSs. It usually implements and balances three competing goals: effective pedagogy, smooth conversation and personalized guidance [21]. Effective pedagogy requires that dialogue-based ITSs can organize conversational teaching in accordance with the pedagogical scaffolds of constructivist principles, so as to enhance students' domain level learning outcomes. Smooth conversation is needed in order to accommodate virtually any input of the learner without having the conversation break down. Personalized guidance requires that ITSs can provide the corresponding feedbacks for different student's responses and organize the corresponding sub-dialogs according to students' misconceptions. Advanced dialogue management usually requires teachers to use a scripting language to create a finite state diagram and the corresponding technical texts for each dialogue [24], including defining specific recipes (conversation planning) and their steps, defining the initial answer class labels, assigning optional semantic labels to be used in implementing optional step and difficulty level transitions, and indicating the difficulty levels for each arc and which steps are optional, which is a very complex and time-consuming process.

We argue that the conceptual knowledge model and taxonomic hierarchy in our BCKO ontology is a type of natural pedagogical scaffold of conceptual learning, which is in line with the theory of the zone of proximal development [14]. Based on this perception, we propose an ontology-driven dialogue management mechanism for conceptual learning dialogue, which can quickly form the conversational content and conversational sequence and doesn't require teachers to design extra conversation planning. Our dialogue management mechanism consists of the following ontology-based measures: (a) conceptual knowledge is decomposed into the concept's definitional axioms, definitional attributes restrictions, other attribute restrictions and relation restrictions, and all concepts are organized into an "is-a" taxonomic hierarchy; (b) the learning dialogue process for a given concept is carried out according to the order of the above conceptual knowledge structure; the learning dialogue sequence between concepts is based on the taxonomic hierarchy in the domain ontology, and the next learned concept is selected from the hierarchy according to the depth-first rule; (c) a misconception language description is associated with the corresponding feedback information and remedial concepts through a special string attribute restriction in the domain ontology; (d) the question templates in the template library are used as the question types and question language templates for the tutor's asking questions to students; (e) the targeted prompt information is automatically generated through a comprehensive consideration of the wrong key elements in the student's answer and conversational turns; and (f) polite phrases (e.g. "please tell me") and cognitive questions (e.g. "Do you understand?") are automatically generated and inserted through a proposed context-based algorithm to make the conversation smoother.

Based on the scaffold instruction theory of the zone of proximal development [14], our ontology-driven dialogue frame involves collaborative discussion, heuristic teaching, and encouragement for the learner to construct knowledge rather than merely receiving knowledge, which is implemented in Table 5.

TABLE 5. Algorithm 2. dialogue management.

```
Algorithm 2. Ontology-driven dialogue management
input: student s
  s←q=getFirstQuestion(s) /*The tutor presents the first question q for
  the student s based on the definition of the concept that may be the
  first concept in the domain ontology or the student's specified
  learning concept or the continuation of the last conversation*/
  inp←getInput(s)
  while inp is not a termination dialogue instruction do
    c←getConcept(q)          /*Current Learning Concept*/
    if inp is a question or help request then
      s←Response(q, inp)
      inp←getInput(s)
    else if inp is a misconception then
      s←misPrompt(q, inp)
      subDialogueForMisconception (s, inp)
      inp←getInput(s)
    else if inp is a wrong then
      if the conversation of question q is less than 5 turns then
        s←Error-basedHint(q, inp) /*a targeted hint is given by an
          error-condition-based feedback function*/
        inp←getInput(s)
      else
        s←getAnswer(q)
        s←getMultimediaLink(q)
      end if
    else if inp is correct then
      s←PositiveFeedback(q)
      s←q=getNextQuestion(s)/* Firstly, it is determined whether
  the current concept's learning is completed, if not, the next dialogue
  question is chosen in the order of the conceptual knowledge structure;
  if yes, the next learned concept that is not covered by the student's
  learner model is chosen in the taxonomic hierarchy of the domain
  ontology according to the depth-first rule and the learner model*/
      inp←getInput(s)
    end if
  end while
output: Current Learning Concept c
```

Table 6 shows a dialogue fragment based on the above dialogue management algorithm in CQACD system.

IX. EVALUATIONS

As stated in Chapter 3, our CQACD system can play the following two roles in the *computer basics* course: an intelligent answering system to answer the questions from students about basic computer knowledge and a dialogue-based ITS for guiding students to learn and construct basic computer knowledge. Therefore, CQACD system is evaluated from three perspectives: template matching performance, QA system performance and tutoring effectiveness. A template matching evaluation primarily examines the classification performance of the matching & classification module in CQACD system, which investigates the classification performance of each matching algorithm based on the following indices: the input's classification precision, recall and F_1-measure, of which the F_1-measure value is employed as the final comprehensive evaluation index. A QA system evaluation examines the total performance of CQACD as a QA system, which investigates the overall performance of related modules and the completeness and accuracy of the

TABLE 6. A dialogue fragment in CQACD.

Conversational content
1 Tutor: What type of storage is memory?
2 Student: A storage that can be accessed by the CPU
3 Tutor: Yes, and please tell me, what are its components?
4 Student: Integrated circuit
5 Tutor: Very good, please keep this state. Please tell me further, what is the difference between it and the hard disk?
6 Student: I do not know, please tell me!
7 Tutor: The speed of memory is faster than that of hard disk, and the capacity of memory is usually smaller than that of hard disk. After power off, the data in the memory will disappear, and the data in the hard disk will not.
8 Student: Why is this?
9 Tutor: Because the components of the memory are integrated circuits and hard disk is composed of the magnetic disk. Do you understand? More detailed learning resources: *http://172.20.78.130/ CQACD/Memory/Learn? content= difference_from_hard_disk*
10 Student: Yes, I got it.
11 Tutor: Ok, please tell me further, what does the memory include?
12 Student: RAM,ROM
13 Tutor: Yes, but what else?
14 Student: Cache
15 Tutor: Yes, very good. Let's learn CPU. Please tell me, what is CPU?
……

subject knowledge in the domain ontology with the rate of correct answers provided by the QA system to the questions as the final evaluation index. A tutoring effectiveness evaluation examines the effectiveness of CQACD system as a DBITS for guiding or helping students to learn and construct the basic computer knowledge in the *Computer Basics* course, which is done by comparing it with the learning effectiveness of the traditional class instruction based on teacher lectures and self-regulated learning based on the teaching website.

A. TEMPLATE MATCHING EVALUATION

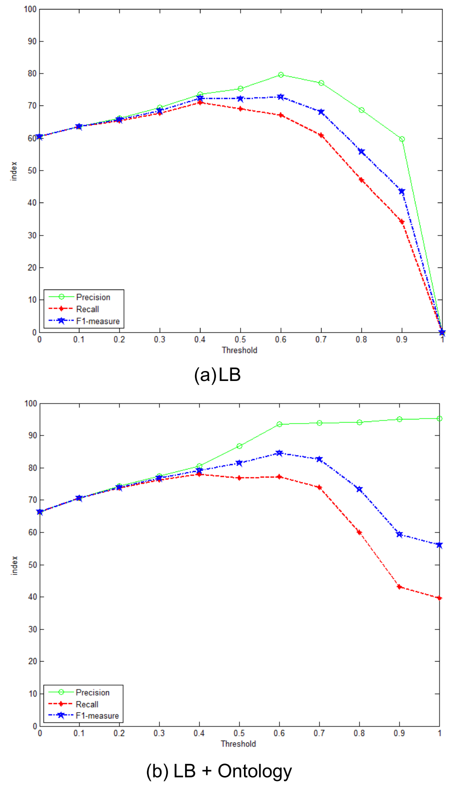
In order to evaluate the matching & classification module, we asked 10 students to pose a related question or instruction based on each one in the template library, respectively, resulting in a total of 530 different input sentences, and we apply these 530 questions or instructions to template matching evaluation experiments. The questions given by these 10 students are all based on the relevant concepts and relationships in the basic areas of computer science. We use the following three inference methods to evaluate template matching:

(1) **Lexical baseline inference (LB)**[41]: directly consider the mandatory word-based lexical inference in formulae (4) and (6) to evaluate template matching, which is the benchmark for subsequent two improved algorithms. The final lexical-based entailment degree of the input to the template is calculated as follows;

$$Ent(Inp, Temp) = \frac{2 \times spMatch(Inp, Temp) \times LCSmatch(Inp, Temp)}{spMatch(Inp, Temp) + LCSmatch(Inp, Temp)} \quad (12)$$

TABLE 7. The best input classification indices of three methods.

Method	Matching Threshold	Precision	Recall	Best F_1-measure
LB [41]	0.6	79.66	67.14	72.87
LB + Ontology	0.6	93.43	77.14	84.51
LB + Ontology + Semantics	0.5	**95.54**	**91.71**	**93.59**



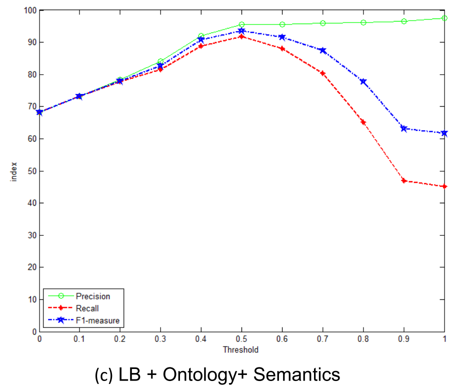


FIGURE 3. Precision, recall and F-measure for each matching inference.

(2) **Ontology-Based lexical baseline inference (LB + Ontology):** the ontology element type is considered in the lexical inference formulae (12), as stated in Section 6.2.

(3) **LB + Ontology + Semantics-based inference (LB + Ontology + Semantics):** Based on the above two methods, further consider the semantic similarity-based matching inference described in Section 6.3. The final semantics-based entailment degree of the input to the template is calculated as

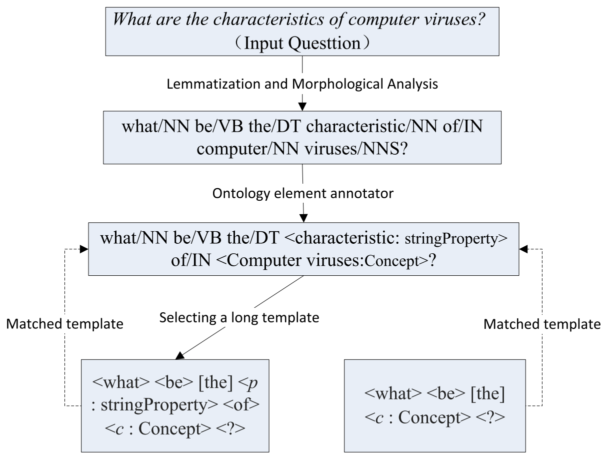

What are the characteristics of computer viruses?
(Input Questtion)
Lemmatization and Morphological Analysis
what/NN be/VB the/DT characteristic/NN of/IN computer/NN viruses/NNS?
Ontology element annotator
what/NN be/VB the/DT <characteristic: stringProperty> of/IN <Computer viruses:Concept>?
Matched template
Selecting a long template
Matched template
<what> <be> [the] <p : stringProperty> <of> <c : Concept> <?>
<what> <be> [the] <c : Concept> <?>

FIGURE 4. Selecting a long template form two matched templates.

follows;

$$Ent^{I}(Inp, Temp) = \frac{2 \times spMatch^{I}(Inp, Temp) \times LCSmatch^{I}(Inp, Temp)}{spMatch^{I}(Inp, Temp) + LCSmatch^{I}(Inp, Temp)} \tag{13}$$

where $spMatch^{I}(Inp, Temp)$ is a simple matching algorithm with word similarity based on Eq. (9) and Eq. (10), $LCSmatch^{I}(Inp, Temp)$ is a continuous subsequence matching algorithm with word similarity based on Eq. (9) and Eq. (10). In addition, in LB + ontology + Semantics-based inference, the system will perform the tone analysis and processing described in Section 6.3.

If in the template library there are multiple templates whose matching degrees with a given student's input are greater than the specified matching threshold, the template with the highest matching degree is selected as the final matched template for the student's input. Further, if there are multiple templates with the same maximum matching degree, the template with more mandatory elements is given a higher priority. The classification results of three matching inference methods are shown in Figure 3.

Through the trend analysis in Figure 3, we can get the best F_1-measure value of each method and its corresponding matching threshold, precision and recall, as shown in Table 7.

To facilitate the understanding of the role of TE algorithms in template matching, we give the following two typical cases in the evaluation:

Case 1: The system selects a long template form two matched templates with the same maximum matching degree, as shown in Figure 4.

Case 2: The system selects a correct template through the semantic similarity based on WordNet in LB + Ontology + Semantics-based inference, as shown in Figure 5.

B. QA SYSTEM EVALUATION

In order to evaluate the overall performance of CQACD as a QA system, we asked 10 students who didn't know anything about our system and research to ask questions to the system.

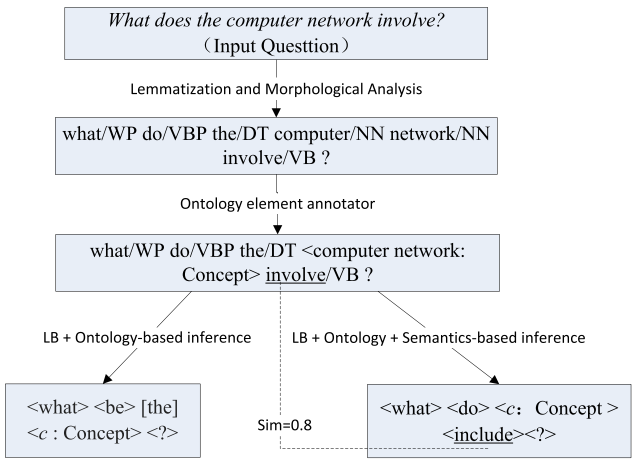

What does the computer network involve?
(Input Questtion)
Lemmatization and Morphological Analysis
what/WP do/VBP the/DT computer/NN network/NN involve/VB ?
Ontology element annotator
what/WP do/VBP the/DT <computer network: Concept> involve/VB ?
LB + Ontology-based inference
LB + Ontology + Semantics-based inference
<what> <be> [the] <c : Concept> <?>
Sim=0.8
<what> <do> <c: Concept > <include><?>

FIGURE 5. Selecting a correct template through semantic similarity.

TABLE 8. QA system performance evaluation results.

Method	Matching Threshold	Accuracy	Correct Rate	Wrong Answer	Uncertain
LB [41]	0.6	65.51	57	30	13
LB+ Glove[6]	0.5	71.59	63	25	12
InferSent [48]	0.5	78.65	70	19	11
LB + Ontology	0.6	83.33	75	15	10
LB + Ontology + Semantics	0.5	**92.63**	**88**	7	5

Everyone asks 15 questions about basic computer knowledge, resulting in a total of 100 different questions, and then let the system make reply. Based on the best matching threshold selected for each method in Table 7, we employed the following formulae to evaluate the QA system performance, yielding the experimental results shown in Table 8.

$$\text{Accuracy} = \frac{\text{Number of questions answered correctly}}{\text{Number of questions answered}} \times 100\% \tag{14}$$

$$\text{Correct rate} = \frac{\text{Number of questions answered correctly}}{\text{Total number of questions}} \times 100\% \tag{15}$$

In Table 8, we use five template-matching methods to evaluate our QA system, in which there are three benchmark methods:

(1) **Lexical baseline inference (LB)** is derived from [41] and follows the same computational procedure as described in the previous subsection.

(2) **LB+Glove** is a semantic similarity-based matching inference method. It uses Eqs. (4) to (9) and (13) for matching inference, in which *Sim* (*j*, *i*) in Eq. (9) is achieved by the cosine similarity between the Glove word embedding vectors of words *j* and *i*.

(3) **Inference sentence (InferSent)** [48] is also a semantic similarity-based matching inference method. It uses InferSent that is a pre-trained sentence embedding model [48] based on

the Bi-LSTM neural network to encode the matched student input and input template and uses the sentence embedding cosine similarity to calculate the semantic similarity between the student input and the input template.

The matching thresholds in Table 8 are derived from the best matching threshold in Table 7. For fairness reasons, we use the same matching threshold 0.5 of our proposed semantic matching method LB+Ontology +Semantics for two semantic benchmark methods LB+Glove and InferSent.

C. TUTORING EFFECTIVENESS EVALUATION

We set up three learning models to investigate and compare the tutoring effectiveness of CQACD system: the traditional class instruction (TCI) model based on teacher lectures, the self-regulated learning (SRL) model based on the educational website, and the personalized tutoring model based on our CQACD system. In order to reduce the impact of students' own factors on learning outcomes, all the students participating in the experiment were selected from the freshmen of physics, mathematics and electronics in our school, their entrance examination subjects were the same, and the average enrollment scores for these three majors were very close, which ensured their cognitive levels were close. In addition, in order to investigate the impact of class size on learning outcomes, we set up four class types with different student numbers for each of three learning modes. Finally, we evaluated the learning effectiveness of three learning models with the same test paper that consists of 100 single-choice questions, and obtained the experimental results shown in Table 9.

X. DISCUSSIONS

From the above experiment results, we can draw several conclusions. First, the results presented in Fig. 3 and Table 7 show that when the matching threshold is less than 0.4, the classification precision is the same as or similar to the recall rate because most of student inputs can match a template from the template library. When the matching threshold is greater than 0.4, the difference between the classification precision and recall rate increases because there are more student inputs that cannot match a template from the template library. The classification precision and recall rate in the mandatory word-based lexical inference are lower because in this inference method the ontology element variable in the template cannot be matched by the ontology element in the student input. Fig. 3 shows that the classification precision in LB inference can be significantly improved by the ontology element type-based inference while the classification recall rate in LB inference can be significantly improved by the semantic similarity-based inference.

Secondly, the evaluation results in Table 8 show that CQACD system can give 88% correct answers through our proposed template matching method based on the ontology and semantics, which demonstrates that the overall QA performance of CQACD system has reached a higher level and CQACD is fully capable of acting as an intelligent

TABLE 9. Learning effectiveness comparisons.

Learning Model	30 students / class	50 students / class
	M ± SD	M ± SD
TCI	(84.2 ± 12.2)†	(83.6 ± 12.4)†
SRL	(75.9 ± 14.6)‡	(76.3 ± 14.8)‡
CQACD	(81.2 ± 12.2)‡	(81.4 ± 12.4)‡
Learning Model	75 students / class	100 students/class
	M ± SD	M ± SD
TCI	(82.5 ± 12.6)‡	(81.8 ± 12.7)‡
SRL	76.2 ± 14.7	76.2 ± 14.7
CQACD	81.3 ± 12.3	81.3 ± 12.3

Note. Confidence levels=0.999 except for †=0.95, ‡=0.99;
M refers to mean score, SD refers to Standard Deviation.

answering system of basic computer knowledge in the *computer basics* course. Through further analysis, we found that after the introduction of semantic analyses, the system's error answering and failure to identify the question are mainly due to that students' questions are beyond the coverage of domain ontology knowledge rather than template matching errors. This out of coverage is mainly reflected in the following three aspects:

(1) Being completely beyond, that is, the system cannot identify any ontology elements in the student's question. At this point, the student's question sentence is judged as being uncertain. For example, the question "What is a television?", in which the student mistakes the television as a type of computer.

(2) The partial excess that causes the question to be mismatched. That is, the system identifies only some of the ontology elements in the student's question, causing the question to be mismatched and wrongly answered. For example, for the question "What is the price of memory?", because the system did not define the attribute *price*, the question is mismatched with the template "What is < c: concept>?" and wrongly answered with the definition of *memory*.

(3) The partial excess that causes the system cannot correctly infer the answer. For example, for the question "Can *Data* be transferred to the *CPU*?", it is correctly classified as matching the template "<can> [the] < c_1:Concept> < r:Relation>[the]< c_1:Concept>[?]", but the system defines only the existence restriction$Data \subseteq \exists be_transfer_to.host$ rather than $Data \subseteq \exists be_transfer_to.CPU$in the Domain ontology, resulting in that the system gives the wrong answer *No*. Such errors can be gradually reduced with the gradual improvement of the domain ontology.

Table 8 also shows that although two semantic benchmark methods LB+Glove and InferSent can improve the lexical baseline inference (LB), they are still far inferior to our proposed matching methods LB+Ontology and LB+Ontology + Semantics, which reveals that ontology elements and their types play a key role in question template matching since the template contains ontology element variable.

Finally, Table 9 demonstrates that the teacher-led class instruction model gained a greater advantage in the small class instruction, such as obtaining a mean score of 84.2 in the small class with 30 students. However, with the number of students in the class being increased, the learning effect of this model showed a significant downward trend, for example, when the number of students per class is increased to 100, the mean score dropped to 81.8, which is mainly due to the limited capacity of a single teacher, so when the increase of students in the class, the attention and personalized tutoring of the teacher to each student in the classroom will decline. On the contrary, website-based SRL and CQACD-based learning are both technology-enabled services that have a higher confidence level than TCI-based learning, so the increase of students in the class has little effect on both. Especially CQACD-based tutoring mode still maintained a mean score of 81.3 in the large class (100 students / class), this mean score is much higher than self-regulated learning effectiveness (76.1) and has approached the teacher-led tutoring effectiveness (81.8) in the large class instruction, thereby demonstrating the CQACD's ability to replace the teachers' tutoring in large classes. In addition, Table 9 also demonstrates that CQACD-based tutoring mode has a lower standard deviation for the test scores of large classes than both TCI and SRL models, which proves that the constructivist-based conversational pedagogical strategy in CQACD can promote poor students' learning.

XI. CONCLUSION

In this study, we employ a domain ontology with rich semantic relationships and a limited number of input templates with description logics to design and implement a tutor and student mixed initiative concept QA system of CQACD that can give 88% of the correct answer and replace the teachers' tutoring in large classes. The design ideas of this paper can be used and referenced by other concept-centered computing courses such as *operating systems* and *computer networks*, and even by courses in history and philosophy in the humanities and social sciences.

The development of proposed CQACD system has gone through 8 years and released 3 versions. The axioms it contains have grown from less than 5,000 to more than 30,000. In the future, we intend to improve and perfect CQACD system in the following aspects: (a) we intend to add voice input/output functions to CQACD system. At present, a new CQACD version with voice input/output is being designed and tested by Automatic Speech Recognition (ASR) technology in Ali Cloud; (b) we will refine the domain ontology and input template library in CQACD to further improve CQACD's performance; (c) we intend to standardize CQACD system with SCORM Specifications [44] (ADL, 2009) and the interoperability model proposed by Zhu *et al.* [13] so that our CQACD system can apply to various discipline ITSs.

APPENDIX A

Aninstanceof reasoning function based on Jena OWL Ontology API.

```
public Set<String> Reasoning_function2(String c1, Set<String> S1)
throws Exception {
  if (model == null)
    throw new Exception("The inference engine is not initialized!");
  // model is a global variable that represents the loaded domain ontology
  Set<String> set = new HashSet<String>();
  Iterator<OntClass> iter = model.listClasses();
    //Get the root of the domain ontology
  while(iter.hasNext()){
    OntClass c = iter.next();
      if(c.getLocalNametoStringequals(c1)){
          //Find out the queried concept
        Iterator<OntClass> inneriter = c.listSubClasses();
        while(inneriter.hasNext()){
        // Get all the subclasses of the concept c1
        OntClass sp = (OntClass) inneriter.next();
        String strSP = sp.getURI();
        try{
          String st = strSP.substring(strSP.indexOf('#')+1); set.add(st);
        }catch( Exception e ){ }
        }
        break;
      }
  }
  set.removeAll(S1); // Delete subclasses that are excluded
  return set; // Return the answer set
}
```

REFERENCES

[1] D. Moldovan, S. Harabagiu, M. Pasca, R. Mihalcea, R. Girju, R. Goodrum, and V. Rus, "The structure and performance of an open-domain question answering system," in *Proc. 38th Annu. Meeting Assoc. Comput. Linguistics*, 2000, pp. 563–570.

[2] P. Moreda, H. Llorens, E. Saquete, and M. Palomar, "Combining semantic information in question answering systems," *Inf. Process. Manage.*, vol. 47, no. 6, pp. 870–885, Nov. 2011.

[3] H. Horacek and M. Wolska, "Interpreting semi-formal utterances in dialogs about mathematical proofs," *Data Knowl. Eng.*, vol. 58, no. 1, pp. 90–106, Jul. 2006.

[4] V. Lopez, M. Fernández, E. Motta, and N. Stieler, "PowerAqua: Supporting users in querying and exploring the semantic web," *Semantic Web*, vol. 3, no. 3, pp. 249–265, Jul. 2012.

[5] X. H. Zhu, Q. H. Cao, and F. F. Su, "A Chinese intelligent question answering system based on domain ontology and sentence templates," *Int. J. Digit. Content Technol. Appl.*, vol. 5, no. 11, pp. 158–165, Nov. 2011.

[6] M. Dzikovska, N. Steinhauser, E. Farrow, J. Moore, and G. Campbell, "BEETLE II: Deep natural language understanding and automatic feedback generation for intelligent tutoring in basic electricity and electronics," *Int. J. Artif. Intell. Educ.*, vol. 24, no. 3, pp. 284–332, Sep. 2014.

[7] A. H. Asiaee, T. Minning, P. Doshi, and R. L. Tarleton, "A framework for ontology-based question answering with application to parasite immunology," *J. Biomed. Semantics*, vol. 6, no. 1, pp. 31–56, Jul. 2015.

[8] F. Baader, D. Calvanese, D. L. Mcguinness, D. Nardi, and P. F. Patel-Schneider, *The Description Logic Handbook: Theory, Implementation, and Applications*. New York, NY, USA: Cambridge Univ. Press, 2007.

[9] S. Nesic, D. Gasevic, M. Jazayeri, and M. Landoni, "A learning content authoring approach based on semantic technologies and social networking: An empirical study," *J. Educ. Technol. Soc.*, vol. 14, no. 4, pp. 35–48, Oct. 2011.

[10] J. Marciniak, "Building intelligent tutoring systems immersed in repositories of E-learning content," *Proc. Comput. Sci.*, vol. 35, no. 1, pp. 541–550, 2014.

[11] B. Vesin, M. Ivanović, A. Klašnja-Milićević, and Z. Budimac, "Protus 2.0: Ontology-based semantic recommendation in programming tutoring system," *Expert Syst. Appl.*, vol. 39, no. 15, pp. 12229–12246, 2012.

[12] M. H. Mahmoud and S. H. A. El-Hamayed, "An intelligent tutoring system for teaching the grammar of the Arabic language," *J. Electr. Syst. Inf. Technol.*, vol. 3, no. 2, pp. 282–294, Sep. 2016.

[13] X.-H. Zhu, T.-J. Wu, and H.-C. Chen, "An interoperable model for the intelligent content object based on a knowledge ontology and the SCORM specification," *J. Educ. Comput. Res.*, vol. 56, no. 5, pp. 723–749, Sep. 2018.

[14] L. S. Vygotsky, *Mind in Society: The Development of Higher Psychological Processes*. Cambridge, MA, USA: Harvard Univ. Press, 1978.

[15] S.-J. Yen, Y.-C. Wu, J.-C. Yang, Y.-S. Lee, C.-J. Lee, and J.-J. Liu, "A support vector machine-based context-ranking model for question answering," *Inf. Sci.*, vol. 224, no. 2, pp. 77–87, Mar. 2013.

[16] G. Zhou, Y. Zhou, T. He, and W. Wu, "Learning semantic representation with neural networks for community question answering retrieval," *Knowl.-Based Syst.*, vol. 93, no. 1, pp. 75–83, Feb. 2016.

[17] P. Molino, P. Lops, G. Semeraro, M. D. Gemmis, and P. Basile, "Playing with knowledge: A virtual player for 'who wants to be a millionaire?' that leverages question answering techniques," *Artif. Intell.*, vol. 222, no. 1, pp. 157–181, May 2015.

[18] A. Figueroa and G. Neumann, "Context-aware semantic classification of search queries for browsing community question–answering archives," *Knowl.-Based Syst.*, vol. 96, pp. 1–13, Mar. 2016.

[19] D. Wang, "Answering contextual questions based on ontologies and question templates," *Frontiers Comput. Sci. China*, vol. 5, no. 4, pp. 405–418, Dec. 2011.

[20] Ó. Ferrández, R. Izquierdo, S. Ferrández, and J. L. Vicedo, "Addressing ontology-based question answering with collections of user queries," *Inf. Process. Manage.*, vol. 45, no. 2, pp. 175–188, Mar. 2009.

[21] S. D'Mello and A. Graesser, "Design of dialog-based intelligent tutoring systems to simulate human-to-human tutoring," in *Where Humans Meet Machines*, A. Neustein and J. A. Markowitz, Eds. New York, NY, USA: Springer, 2013, pp. 233–269.

[22] E. A. Domeshek, "Scenario-based conversational intelligent tutoring systems for decision-making skills," in *Proc. Interservice/Ind. Training, Simulation, Educ. Conf. (I/ITSEC)*, 2009, pp. 1–11.

[23] V. Aleven, O. Popescu, A. Ogan, and K. R. Koedinger, "A formative classroom evaluation of a tutorial dialogue system that supports self-explanation," in *Proc. 11th Int. Conf. Artif. Intell. Educ.*, 2003, pp. 303–312.

[24] P. W. Jordan, M. Makatchev, U. Pappuswamy, K. Vanlehn, and P. L. Albacete, "A natural language tutorial dialogue system for physics," in *Proc. 19th Int. FLAIRS Conf.*, 2006, pp. 521–526.

[25] J. R. Carbonell, "AI in CAI: An artificial-intelligence approach to computer-assisted instruction," *IEEE Trans. Man-Machine Syst.*, vol. MMS-11, no. 4, pp. 190–202, Dec. 1970, doi: 10.1109/TMMS.1970.299942.

[26] M. W. Evens, R.-C. Chang, Y. H. Lee, L. S. Shim, C. W. Woo, Y. Zhang, J. A. Michael, and A. A. Rovick, "CIRCSIM-tutor: An intelligent tutoring system using natural language dialogue," in *Proc. 5th Conf. Appl. natural Lang. Process. Descriptions Syst. Demonstrations Videos*, 1997, pp. 13–14.

[27] R. Freedman, "Atlas: A plan manager for mixed-initiative, multimodal dialogue," in *Proc. Workshop Mixed-Initiative Intell.*, 1999, pp. 1–8.

[28] A. C. Graesser, S. Lu, G. T. Jackson, H. H. Mitchell, M. Ventura, and A. Olney, "Autotutor: A tutor with dialogue in natural language," *Behav. Res. Methods Instrum. Comput., J. Psychonomic Soc.*, vol. 36, no. 2, pp. 180–193, Jun. 2004.

[29] K.-S. Song, X. Hu, A. Olney, and A. C. Graesser, "A framework of synthesizing tutoring conversation capability with web-based distance education courseware," *Comput. Educ.*, vol. 42, no. 4, pp. 375–388, May 2004.

[30] S. K. Ray, S. Singh, and B. P. Joshi, "A semantic approach for question classification using WordNet and Wikipedia," *Pattern Recognit. Lett.*, vol. 31, no. 13, pp. 1935–1943, Oct. 2010.

[31] A. Mishra and S. K. Jain, "A survey on question answering systems with classification," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 28, no. 3, pp. 345–361, Jul. 2016.

[32] D. Zhang and W. S. Lee, "Question classification using support vector machines," in *Proc. 26th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2003, pp. 26–32.

[33] S. Hazrina, N. M. Sharef, H. Ibrahim, M. A. A. Murad, and S. A. M. Noah, "Review on the advancements of disambiguation in semantic question answering system," *Inf. Process. Manage.*, vol. 53, no. 1, pp. 52–69, Jan. 2017.

[34] W. Cui, H. Wang, H. Wang, Y. Song, S. W. Hwang, and W. Wang, "KBQA: Learning question answering over QA corpora and knowledge bases," in *Proc. Vldb Endowment*, 2017, pp. 565–576.

[35] G. A. Miller and C. Fellbaum, "Semantic networks of English," *Cognition*, vol. 41, nos. 1–3, pp. 197–229, Dec. 1991.

[36] A. T. Corbett and J. R. Anderson, "Knowledge tracing: Modeling the acquisition of procedural knowledge," *User Model. User-Adapted Interact.*, vol. 4, no. 4, pp. 253–278, Dec. 1995.

[37] C. Conati, A. Gertner, and K. Vanlehn, "Using Bayesian networks to manage uncertainty in student modeling," *User Model. User-Adapted Interact.*, vol. 12, no. 4, pp. 371–417, Nov. 2002.

[38] V. Dimitrova, "STyLE-OLM: Interactive open learner modelling," *Int. J. Artif. Intell. Educ.*, vol. 13, no. 1, pp. 35–78, Jan. 2003.

[39] D. Pérez-Marín, E. Alfonseca, P. Rodríguez, and I. Pascual-Nieto, "A study on the possibility of automatically estimating the confidence value of students' knowledge in generated conceptual models," *J. Comput.*, vol. 2, no. 5, pp. 17–26, Jul. 2007.

[40] A. C. Graesser and N. K. Person, "Question asking during tutoring," *Amer. Educ. Res. J.*, vol. 31, no. 1, pp. 104–137, Mar. 1994.

[41] O. Ferrández, D. Micol, R. Munoz, and M. Palomar, "DLSITE-1: Lexical analysis for solving textual entailment recognition," in *Proc. 12th Int. Conf. Appl. Natural Lang. Inf. Syst.*, 2007, pp. 284–294.

[42] E. M. Flores and S. J. Rigo, "A model for textual entailment based on linguistic rules," in *Proc. Int. Conf. Comput. Process. Portuguese Lang.*, 2016, pp. 251–255.

[43] G. Jorge-Botana, J. M. Luzón, I. Gómez-Veiga, and J. I. Martín-Cordero, "Automated LSA assessment of summaries in distance education: Some variables to be considered," *J. Educ. Comput. Res.*, vol. 52, no. 3, pp. 341–364, Mar. 2015.

[44] ADL. (2009). *Sharable Content Object Reference Model (SCORM) (4th ed., Version1.1)*. [Online]. Available: http://www.adlnet.gov

[45] X. Zhu, Q. Xu, Y. Chen, H. Chen, and T. Wu, "A novel class-center vector model for text classification using dependencies and a semantic dictionary," *IEEE Access*, vol. 8, pp. 24990–25000, 2020.

[46] F. Li, L. Liao, L. Zhang, X. Zhu, B. Zhang, and Z. Wang, "An efficient approach for measuring semantic similarity combining WordNet and Wikipedia," *IEEE Access*, vol. 8, pp. 184318–184338, 2020.

[47] Y. C. Fung, C. W. Kwok, L. K. Lee, K. T. Chui, and H. U. Leong, "Automatic question generation system for English reading comprehension," in *Proc. Int. Conf. Technol. Educ. (CCIS)*, vol. 1302, 2020, pp. 136–146.

[48] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes, "Supervised learning of universal sentence representations from natural language inference data," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2017, pp. 670–680.

YU WEN received the B.S. and M.S. degrees in law from Guangxi Normal University, China. She is currently with the Student Work Department, Guilin Tourism University. Her research interests include intelligent tutoring systems and question-answering systems.

XINHUA ZHU received the B.S. degree in computer science from the Department of Radio Electronics, Beijing Normal University, China. He is a Professor with the School of Computer Science and Engineering, Guangxi Normal University, China. His research interests include intelligent tutoring systems, natural language processing, and knowledge graphs.

LANFANG ZHANG received the B.S. and M.S. degrees in computer science from Guangxi Normal University, China. She is a Professor with the Faculty of Education, Guangxi Normal University. Her research interests include artificial intelligence in education, natural language processing, and knowledge graphs.