International Journal of Mechanical, Industrial and Aerospace Sciences
ISSN: 2517-9950
Vol:14, No:12, 2020

AI Tutor: A Computer Science Domain Knowledge Graph-Based QA System on JADE platform

Yingqi Cui, Changran Huang, Raymond Lee

***Abstract*—**In this paper, we proposed an AI Tutor using ontology and natural language process techniques to generate a computer science domain knowledge graph and answer users' questions based on the knowledge graph. We define eight types of relation to extract relationships between entities according to the computer science domain text. The AI tutor is separated into two agents: learning agent and Question-Answer (QA) agent and developed on JADE (a multi-agent system) platform. The learning agent is responsible for reading text to extract information and generate a corresponding knowledge graph by defined patterns. The QA agent can understand the users' questions and answer humans' questions based on the knowledge graph generated by the learning agent.

***Keywords*—**Artificial intelligence, natural language process, knowledge graph, agent, QA system.

I. Introduction

SINCE Dartmouth Meeting on Artificial Intelligence held in 1956, the technology had become a significant topic in the world to develop a machine that can behave and think intelligently. [1] Many scientists started to design and implement artificial machines hoping that could succeed in the Turing test by 2000. Such tests can reflect the NLP (Natural Language Processing) ability of a program which is one of the research domains on AI. [1] In such circumstances, chatbot, a special program is designed to communicate with humans by natural language such as English and Chinese automatically, has shown the ability to succeed in Turing test with high precision. [2]

Chatbot, also referred to as a conversational program, is an intelligent program that can communicate with humans via texture techniques. Users can enter input in natural language and the chatbot will start to talk with humans in the same language. NLP is one of the most effective techniques in developing an intelligent chatbot and is used to process sentences in natural language to ensure that computers can recognize human languages. There are two crucial effects when humans use natural language to chat with computers. One is the computers need to know the meaning of natural language text which belongs to natural language understanding. The other is the computers need to deliver the intention and idea in natural language text which belongs to natural language generation. Both of the effects are two necessary parts of NLP. Also, machine learning can help train the chatbot with much data to expand intellectually. Thus, AI chatbot has become a crucial application recently and can be used in different areas such as:

- Customer service to answer enquiry from customers efficiently.
- Social media as entertainment with other users.
- A tutor at educational institutions for students to learn knowledge.

According to [3], an agent is a kind of software component with autonomy that can perceive the environment according to own knowledge and achieve their goals through cooperation with other agents in the same environment. A multi-agent system comprises of many different agents to communicate with each other via message-passing protocol to reach their goals. They can enhance intelligent chatbots' performances because of their features [2]. Nevertheless, all multi-agent frameworks like JADE (Java Agent Development Framework) only provide with a platform that supports agents' interact by message exchange and agents cannot communicate with users by natural language.

Many chatbots nowadays have been developed to communicate with humans. In such, a conversational agent can be a very good tutor to help students with their studies at universities, which is an efficient way to enhance the quality of students' learning. Our AI Tutor contains a learning agent and a QA agent. The learning agent uses computer science domain English textbooks to extract knowledge and construct a knowledge graph. The QA agent processes input natural language questions and generates query statements to search answers from the knowledge graph.

The main contributions of this paper are as follows: (1) Design and build a computer science domain knowledge graph. (2) Combine natural language processes and multi-agent system.

II. Literature Review

Hettige and Karunana [4] designed and implemented an intelligent chatbot named Octopus through a multi-agent architecture. There are eight sub-multi-agent systems in Octopus: core system, GUI system, natural language process system, communication system, learning system, action system, searching system, and data access system. Fig. 1 shows the architecture of Octopus. The core system is a manager of the whole chatbot system responsible for managing and deploying other sub-systems. The GUI system provides a graphical user interface for users to enter inputs and monitor Octopus outputs. It contains two agents: text and voice agents to handle inputs and outputs by text or voice language. The

Yingqi Cui and Changran Huang are with the Division of Computer Science and Technology, BNU-HKBU United International College, China (e-mail: kryiecui1999@gmail.com, chris_0319@outlook.com).

Raymond Lee is with the Division of Computer Science and Technology, BNU-HKBU United International College, China (corresponding author, e-mail: raymondshtlee@uic.edu.cn).

NLP system includes several NLP agents that analyze natural language in handling users' inputs and generate natural language as output.

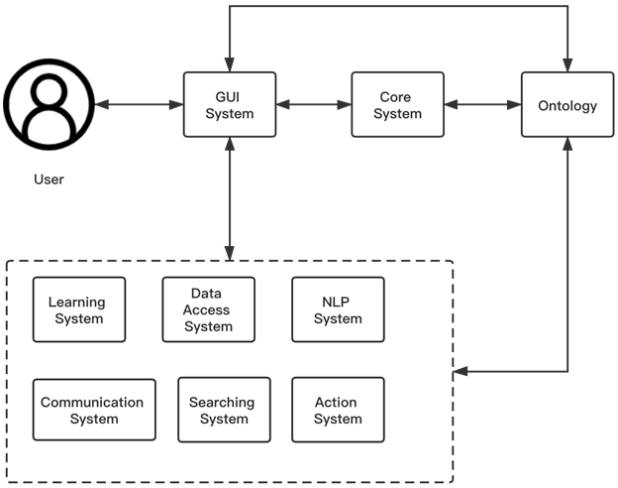

Fig. 1 Octopus' Architecture

The communication system is in charge of communication between different agents and systems. The learning system contains learning agents which can learn and update knowledge ontology to enhance the quality of chatbot's intelligence. The action system has many active agents that are used to perform tasks and reduce workload i.e. open programs, read and search files. Search agents form the searching system. They can search for information from the Internet or their networks and collect results to update knowledge base. The data access system contains data access agents that can send and reserve data through the network. When chatbot executes, GUI system sends input information to the message space, then the natural language process system tries to recognize the text and generates outputs. Finally, GUI system will display the output by voice or text.

Memon et al. [2] developed a multi-agent system that allows two chatbot agents to communicate with each other and users by natural language. When a user asks questions, the chatbot will communicate with other chatbots to generate appropriate answers. A single chatbot at times cannot handle some complex problems but using multi-chatbot can improve the performance and quality of chatbot's intelligence.

The system design methodology is shown in Fig. 2. The user inputs questions in natural language, then the text will preprocess and search in a knowledge base. The Client-Server Socket System provides a mechanism to connect, remote devices, send, and receive data. The Natural Language Process system will divide the sentences into separate words. The result will be used for knowledge recognition. In knowledge recognition system, input is the data that have been processed in the Natural Language Process system, and the knowledge recognition system uses them to generate reasonable answers for corresponding inputs. Knowledge Base is a database that stores knowledge used to interpret input. Chatbot operates knowledge base to generate the response by rule matching. Finally, chatbot selects the correct answers and shares them among different chatbots to obtain the most reasonable answers.

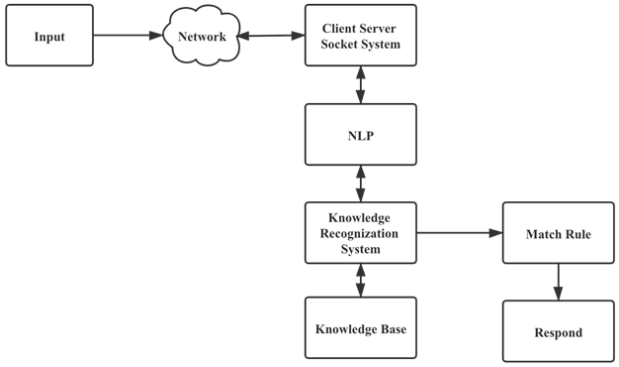

Fig. 2 Chatbots' Methodology

Kumar and Rosé [5] developed an architecture to implement conversational agents that support collaborative learning. Conversational agents have been proved effective in one on one tutorial. However, it is difficult to chat with two or more students at the same time. When the engineer designed conversational agents, they supposed that there are only two participants in the interaction, one is human, and the other is agent. Besides, the speakers take turn alternatively, and the receiver always is the one who is not the speaker. Based on this hypothesis, the design of the agent has been simplified. However, when there is more than one human in the interaction, more mistakes occur in the agent as time grows. To put it differently, suppose that there are two humans in a chatting room and talking. The agent does not know whether the users chat with it. For example, two students are in a chat room. When the first student answers the agent's question, the agent will record the address of student 1 and generate text according to student 1's answers. If student 2 gives a wrong answer followed by student 1 before agent's reply, the agent only replies to student 1 whether the answer is correct, and student 2 will not receive feedback from the agent. Another problem in collaborative learning is that communication between students is the most crucial factor in maintaining conservation quality. The conversation agent needs to monitor and adjust the interaction to ensure that all students participate in the discussion. However, the agent cannot do this because of the above hypothesis. To solve these problems, scientists developed a model representing the interaction between the participant and the environment [5]. The participant can observe stimulation from the environment, and the stimulation will be transferred to the perception component. The agent will then handle stimulation and determine whether to trigger any related behaviors to deal with the stimulus. The behavior can generate events and respond to the environment. Due to the response can be a return to the environment, the internal state of components can be updated. Besides, they developed an architecture called Basilica to control connections and

communications between different components. The advantage of this architecture is that it builds a component network to solve complex interaction in a multi-party interactive environment. It is an object-oriented programming which makes the development more efficient.

According to [3], JADE is a Java-based multi-agent development framework. It follows the FIPA (Foundation for Intelligence Physical Agent) specification and provides various middleware that can be effectively integrated with other Java development platform technologies. The agent platform can be distributed across machines and configuration can be controlled remotely. AMS (Agent Management System) of the JADE platform can maintain and dispatch many agents. It can control an agent's life cycle i.e. initiate, activate, suspend, block, delete, transmit, and copy an agent. Further, each agent can be implemented to execute various types of behaviors. With MTS (Message Transport Service), agents can communicate with other agents in the same container or host. The messages exchanged by agents have a format defined by the ACL (Agent Communication Language). This format includes sender, receiver, communicative intention, content, language, and ontology. Further, JADE uses a yellow page service provided by DF (Director Facilitator) Agent. Yellow page service allows agents to publish one or more functions so that other agents can discover and use them

The idea of ontology was originated from the philosophical field. [6] Its philosophical definition is: ontology is a systematic description of the world's objective things, which is the most metaphysical knowledge. Metaphysics does not mean isolation or stillness but refers to the abstract meaning beyond stillness. It is the most common knowledge, the most general, and the least specific laws in the material world. An ontology defines the fundamental terminology of glossary and their relationships that make up the subject area and the regulations that integrate these terminologies and relationships to define the glossary extension.

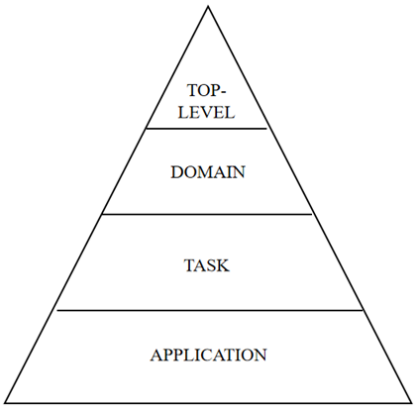

Fig. 3 Pyramid of ontology

As Fig. 3 shows, based on the degree of domain dependence, ontology can be divided into four types:

1. Top-level ontology. It describes the universal ideas and their relationships i.e. events, behaviors, space, time, etc. Other kinds of ontology are a particular instance of such type of ontology.
2. Domain ontology. It is a professional ontology that describes the concepts in a specific field and its relationship.
3. Task ontology. It expresses the ideas of a specific task or behavior as well as the relationship between them.
4. Application ontology. It describes the concepts that depend on specific areas and tasks including the relationship among concepts.

There are many useful benefits to construct an ontology. In computer science, the main objectives of structuring ontology are:

1. Capture knowledge in relevant fields.
2. Supply a collective cognition of knowledge in the field.
3. Determine collectively approved vocabularies in the field.
4. Provide a well-defined relationship between these terminologies and vocabulary from various levels of formalization pattern.

III. Methodology

We express our AI tutor as a knowledge graph-based QA system. To build the tutor, we construct a knowledge graph and design the QA system's architecture. AI tutor's techniques are displayed below including Spacy, OpenNLP for NLP data processing, neo4j graph database to store knowledge and JADE, a multi-agent development platform.

A. Knowledge Graph Construction

Fig. 4 shows the knowledge graph construction. Several questions are to be defined in building a knowledge graph. The first question is the domain of AI tutor. The QA robot is defined based on computer science domain rather than open domain.

The second layer is a model layer to extract knowledge from unstructured data to generate a knowledge graph. The knowledge graph domain has been defined in the request layer, and search for resources about computer science knowledge to extract information. Nevertheless, no models can extract knowledge such as entity and relation from computer science domain text so the relation type based on the specific domain was self-defined before extracting knowledge as shown in Table I.

According to relation type, the system extracts entity and the relationship between entities as SPO (Subject-Predicate-Object) triple. SPO is a summary of a sentence by extracting critical concepts of the original text. It matches the keywords based on defined rules to generate a shorter version of a sentence. Then, the original sentence is transformed into three relations: Subject, Predicate, and Object stored as two nodes and one relationship in the knowledge graph. SPO is used instead of original text for knowledge construction because it represents a more precise and efficient structure. Besides, the brief structured triple is powerful for further operations such as information querying. Finally, the knowledge graph stored in a graph database and the QA system will connect the database to search for answer to questions in the application layer.

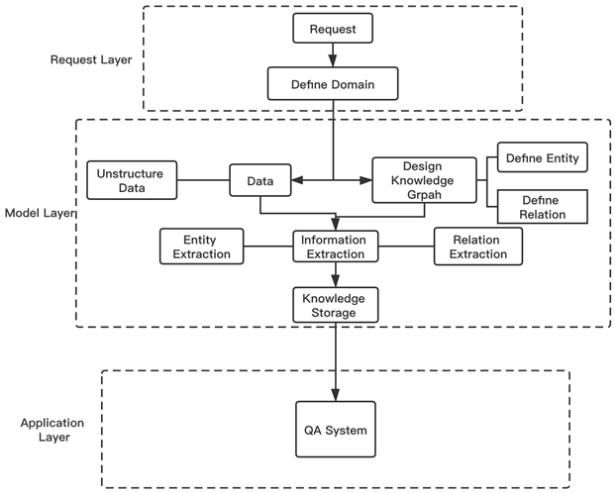

Fig. 4 Knowledge Graph Construction

TABLE I
COMPUTER SCIENCE DOMAIN ONTOLOGY ENTITY RELATION

Relation Type	Definition	Example
Definition	The definition of vocabularies and phrases	< Ontology, is, a relevant logical and graphical model for knowledge representation>
Reason	The reason of something	< Domain ontology is more useful to build intelligent application, because domain ontology is less abstract but more specific>
Developer	The developer or designer of something	< Dong, develop, HowNet>
Drawback	The drawback of something	<WordNet, is strictly limited to, English based application only>
Has_part	The contents of something	< Five different levels in primitives, contains, The epistemological level>
Use_to	The function of something	< knowledge representation, is used by, knowledge engineers or domain experts to study and verify the knowledge>
Example	The example of something	<Lexical ontology, such as, WordNet and HowNet>
The_same_thing	Two things are the same but with different name	<Top-level ontologies, also known as, upper ontologies>

B. QA Systems' Architecture

Fig. 5 shows the architecture of computer science domain knowledge graph-based QA system.

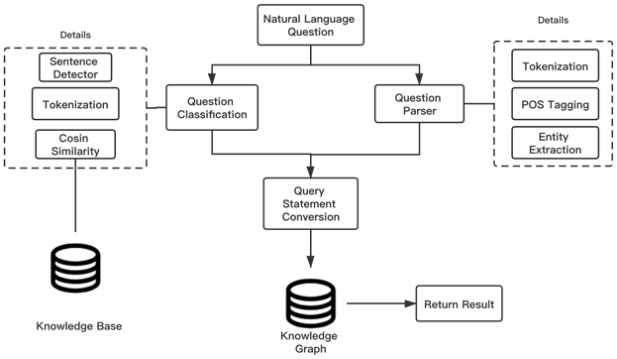

Fig. 5 Knowledge Graph-based QA System's Architecture

For users' questions, the sentences are to be processed to extract useful information to search for corresponding knowledge in the knowledge graph.

There are two steps to process the natural language question before query: question classification and question parser. Question classification can detect the question's type according to another knowledge base, and the type of question will be regarded as the relationship between entities in knowledge graph. The knowledge base is used to store common sense e.g. the knowledge of wh-word such as what, why and how are stored in a particular knowledge base. The system will also extract questions' entities in question parser and translate the extracted entities and relations from question into a query statement to match with the information from computer science domain knowledge graph and return results.

C. Spacy and OpenNLP for NLP Implementation

In this system, Spacy and OpenNLP are used to address the natural language process. Spacy is a python-based industrial toolkit to implement common NLP tasks.

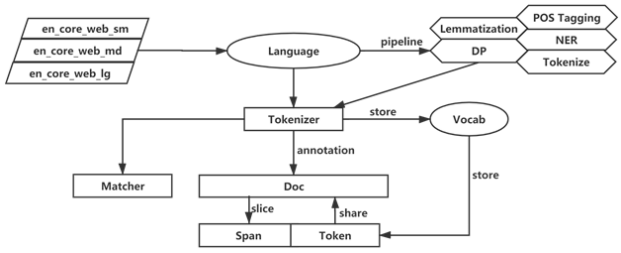

Fig. 6 Spacy Architecture

Fig. 6 shows the architecture of Spacy. Spacy contained many trained machine learning models as shown in parallelograms. After loading a model, an NLP object is created to set up as a pipeline to preprocess the text. To be more intuitive, Fig. 7 shows the function of pipeline that integrates and governs NLP tasks as hexagons illustrated in Fig. 6.

Fig. 7 Spacy Pipeline

The internal pipeline supports Tokenization, Lemmatization, POS Tagging (Part of Speech Tagging), NER (Named Entity Recognition), DP (Dependency Parsing) and other NLP tasks for a sentence.

After passing the pipeline, three classes can briefly implement into the remaining functional operations: Vocab, Doc, Matcher. Vocab is used to store tokens sequences and are shared with class Doc. Doc stores the tokens annotation. Therefore, object doc can be regarded as a container that consists of meaningful information to support further and more complex tasks such as text extraction. Besides, the Class span is a segment of Doc which can be used to implement more specific and accurate extraction. Each token has many annotations inside the Doc such as:

1. ORTH, TEXT. The original form of the token in a sentence is used to exactly match the text.
2. LOWER. The lowercase form of the token.
3. LENGTH. The number of letters of the token.
4. IS_ALPHA, IS_ASCII, IS_DIGIT. Determine the token belongs to the alphabet, ASCII characters, or digits.
5. IS_LOWER, IS_UPPER, ISTITLE. Determine the token belongs to lowercase, uppercase, or title case.
6. IS_PUNCT, IS_SPACE, IS_STOP. Determine the token belongs to punctuation, space, or stop word.
7. LIKE_URL, LIKE_EMAIL. Determine the token belongs to a URL or email.
8. POS. The speech part of the token.
9. LEMMA. Root of the token.
10. DEP. The dependency of token in a sentence.

Based on these annotations, it is sufficient to define many rules to extract the text. [7] A pattern is an extractor that integrates the defined rules. By matching the rules in patterns, class Matcher can extract the tokens stored in class Vocab.

OpenNLP is a Java-based tool to perform basic NLP tasks. After instantiating a trained model by loading a binary file, the specific NLP tasks can be implemented. OpenNLP is powerful and efficient in implementing NLP tasks in Java program. Based on this, it is useful to process tricky questions asked by users. It expands more accurate text extraction and constructs reasonable knowledge.

D.Neo4j Graph Database for Knowledge Graph

Neo4j database was used to store knowledge extracted from the text and API was used to refresh graph and mining knowledge from ontology.

There is no isolated information but interrelated domains around us. Entities and relations can be extracted from information, and entities combined with other entities to form a knowledge graph. Graph databases use the property graph model is shown in Fig. 8 to organize data as nodes, relationships and properties. Nodes represent entities in the graph and relationships provide directed, named connections between two nodes. Graph databases are better in managing highly connected data and complex queries as compared with traditional databases.

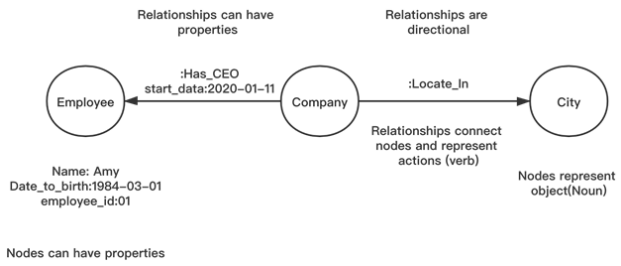

Fig. 8 Property Graph Model

Neo4j is the most popular graph database nowadays. It provides an efficient platform to store property graph model, drivers for other programming languages, and a query language similar to SQL to process data in the database. That means that neo4j is easy to represent connected data and fast to retrieve, traverse, and navigate more connected data. Most importantly, Neo4j database supports native Java API which is a simple API to perform database operations, and Cypher Java API is a powerful Java API to execute more difficult database operations.

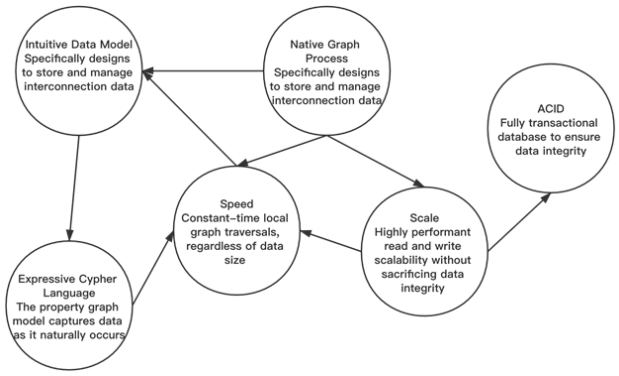

Fig. 9 Neo4j Features

E. Multi-Agent with JADE Platform

Our system is implemented as a multi-agent development platform. JADE is a Java-based multi-agent development framework. It provides a run-time environment to integrate different operations of agents effectively. Each agent is a computer system based on software or hardware which can proactively execute the tasks individually.

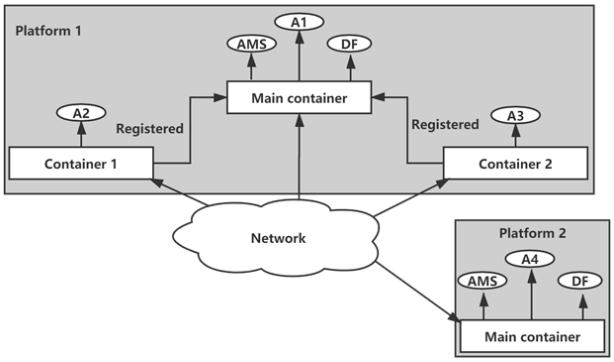

Fig. 10 Structure of JADE Platform

Fig. 10 shows the structure of JADE platform. A JADE platform can have multiple containers, and a container consist of multiple agents. Containers can be located at various hosts. There is only one container called the main container in a JADE platform. When other containers are launched, they must be registered in the main container. In addition, JADE provides many essential components:

1. RAM (Remote Management Agent). It can manage agent remotely, and provide a graphical interface to manage and control the agent platform.
2. Dummy Agent. It is used to monitor and debug, consisting a GUI and a potential JADE body with the graphical user interface. Information can be organized, sent to other

agents, and a list of sent or received messages that can be displayed with a time stamp. Further, the dialogue between agents can be recorded and rehearsed.

3. Sniffer Agent. It can capture information when passing messages between agents and display the process and message content in UML sequence diagrams.
4. Introspector Agent. It can detect agent's life cycle, the exchanged information, and the behavior is executed.
5. DF GUI. It is a graphic interface with a JADE directory service.
6. Log Manager Agent. It is an agent that can set the running log information.

JADE platform supports many types of an agent's behavior. An agent can choose to execute the tasks once, circularly, periodically, sequentially, in parallel, in a finite state machine or so on.

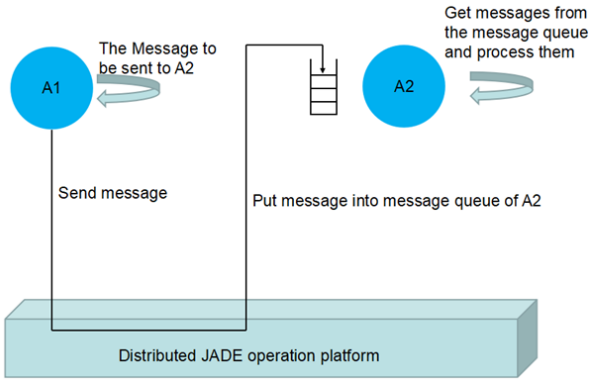

Fig. 11 Asynchronous transmission mechanism

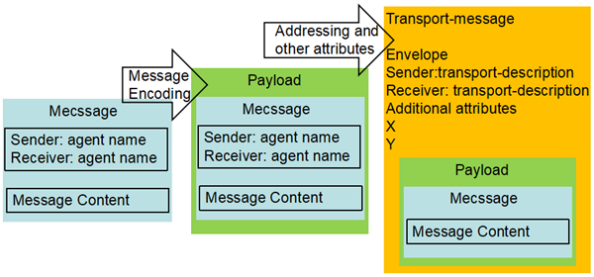

Fig. 12 Message encapsulation

The communication between agents adopts an asynchronous transmission mechanism. As shown in Fig. 11, each agent owns a mailbox to receive messages from other agents. Once there is a message, the system will notify the corresponding agent. The agent calls for the action() method to respond. Fig. 12 shows the message encapsulation process. Before the message is added to the transmission queue, the message is encapsulated in the envelope. In the interval of the message, the content is encoded in ACL which is a medium or tool used by an agent to deliver its viewpoints.

IV. System Implementation and Experimental Test

A. MVC Framework and JADE Platform

AI tutor is implemented as two systems: learning and QA systems. One is for understanding language and training knowledge. The other is for generating language. Fig. 13 shows the architecture of AI Tutor. Overall, it is implemented in an MVC (Model-Viewer-Controller) framework.

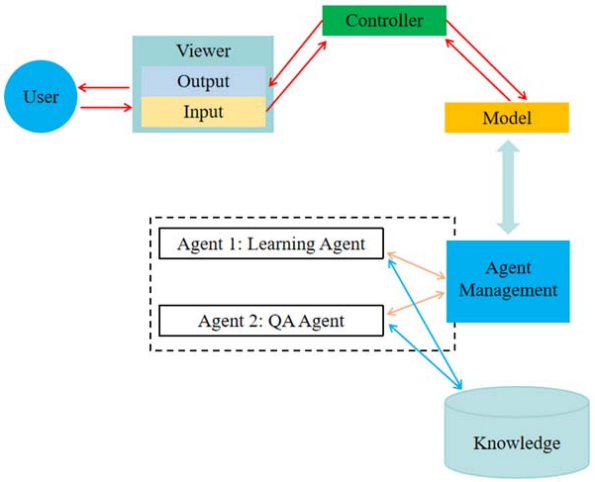

Fig. 13 AI Tutor's Architecture

The Viewer is a GUI (Graphical User Interface) chart window as shown in Fig. 14 which contains input and output windows. Swing and AWT class in Java were used to arrange the graphical windows and buttons to obtain the questions from users, produce output answers, and call agent to perform the QA function. For the output viewer, the original display format was replaced by HTML. HTML5 format was used to make the output interface more diversified.

The controller is responsible for receiving and transferring messages from Viewer and calling the model functions which are used to process the language. An agent manager in model was set up to launch, call, and connect the agents. As shown in Fig. 15, two intelligent agents were deployed: A learning agent to extract information from the book and then to create, maintain, and update the knowledge graph. QA agent to query information and answer generated language.

B. Learning Agent

The learning agent allows system to understand semantic meaning of the inputted text. As shown in Fig. 16, the learning agent is integrated into three layers:

Level 1 is a request layer to prepare for extraction, where Spacy toolkit was imported and loaded into the trained model. Further, sentences were detected and split from the original text and the separated sentences were stored into a list.

Level 2 is an extraction layer. A thread pool was created to implement multiple threads. After receiving separated sentences from the request layer, it preprocesses an individual sentence inside a thread. The Spacy toolkit was used to perform the NLP tasks i.e. tokenize, lemmatize, part of speech tagging,

named entity recognition, and dependency parsing. Each thread is equipped with a defined matching pattern which responds to a specific relationship. Then the pattern was applied to the tokens in class Doc. This layer performs eleven relationship categories, and multiple matching rules were defined for each relationship.

Fig. 14 Graphical User Interface of AI Tutor

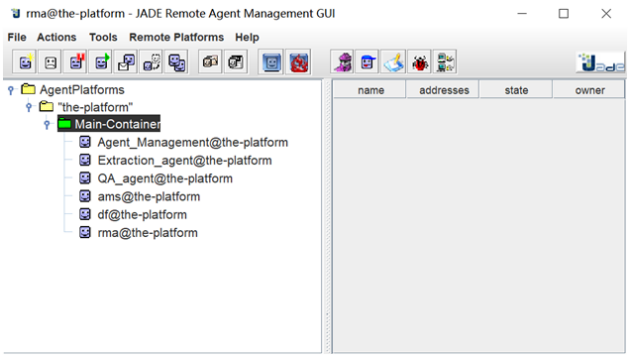

Fig. 15 Agent Management

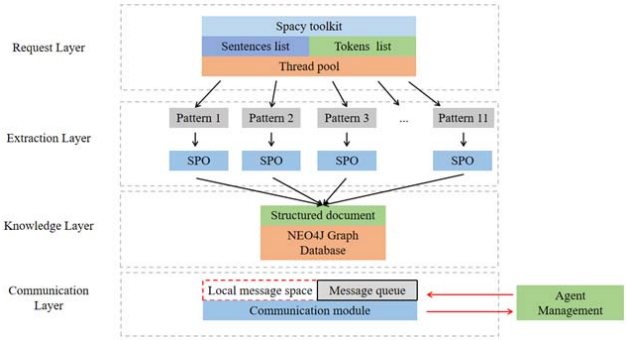

Fig. 16 Learning Agent's Architecture

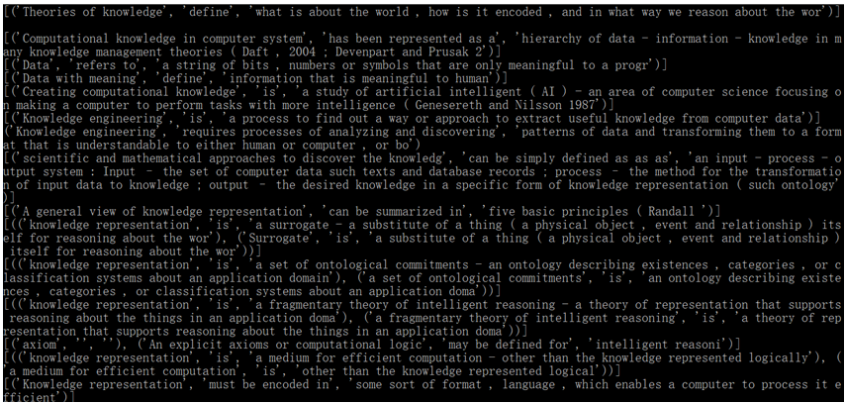

Fig. 17 SPO Extraction's Result

Finally, as shown in Fig. 17, sentences are extracted as corresponding to SPO triples. These SPO triples will be integrated and output as a structured document as shown in Fig. 18 for level 3.

Level 3 is a knowledge layer. From layer 2, a structured document was obtained. The information about SPO information for each sentence was achieved by reading the document. Based on the triples, the knowledge graph was maintained. In this system, Neo4j was used to store and represent the SPO relationships. By instantiating a graph database factory and loading the local graph database, Neo4j console was connected. In Neo4j, one SPO is represented as two nodes and one connecting relationship. The operations to generate and update a Neo4j knowledge graph are as follows:

Step 1 is judging existing nodes based on the name property. If there is no existing node whose name is equal to that of the current node, the current node will be imported as a new node and the corresponding triple will be imported as well. The result

is shown in Fig. 19. If there is an existing node whose name is equal to that of the current node, it will go to step 2.

<Subject>The entire Cyc ontology<Subject>
<has_part>contains<has_part>
<Object>hundreds of thousands of terms with relationship among the terms to model human consensus rea

<Subject>The OpenCyc ontology<Subject>
<has_part>contains<has_part>
<Object>a knowledge server to serve for its Cyc knowledge base , an inference engine , and it also defines (

<Subject>The OpenCyc ontology<Subject>
<definition>is<definition>
<Object>an upper - ontology available for defining some lower level ontology knowledge such as domain sp

<Subject>lexical ontology<Subject>
<definition>is<definition>
<Object>an ontology describing linguistic knowledge , and it tries to model the word meaning by ontologica

<Subject>Lexical ontology<Subject>
<example>such as<example>
<Object>WordNet and Miller<Object>

Fig. 18 Structure SPO Document

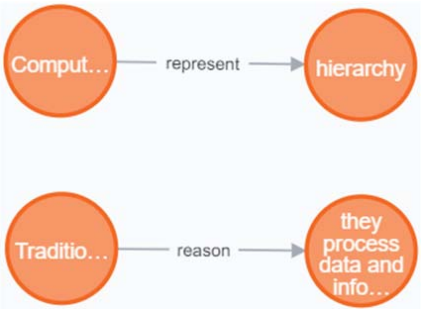

Fig. 19 Individual SPO

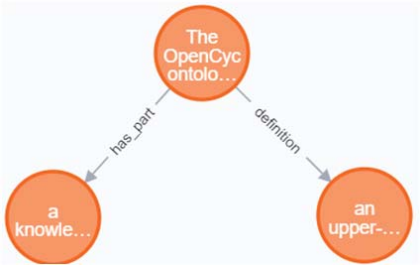

Fig. 20 Connected SPO

Step 2 is the situation that current node conflicts with the existing node. The merging operation will be applied to combine two triples and the result is shown in Fig. 20. Besides, we query the node information by CQL (Cypher Query Language) was used to command insert, delete, retrieve, and update data in Neo4j graph database. Therefore, as long as one of the Subjects or Objects is equal to the existing node, the new and existing ones to form different clustered ontologies. The result is shown in Fig. 21.

Level 4 is a communication layer. The agents can communicate with each other. Each time the agent management sends a command or another agent sends a message to this agent, the command message will temporarily be stored in the message queue. The communication module will remind the agent to execute the command or receive message. When the agent sends a message to another, the message will be encapsulated in the local message space and then transformed.

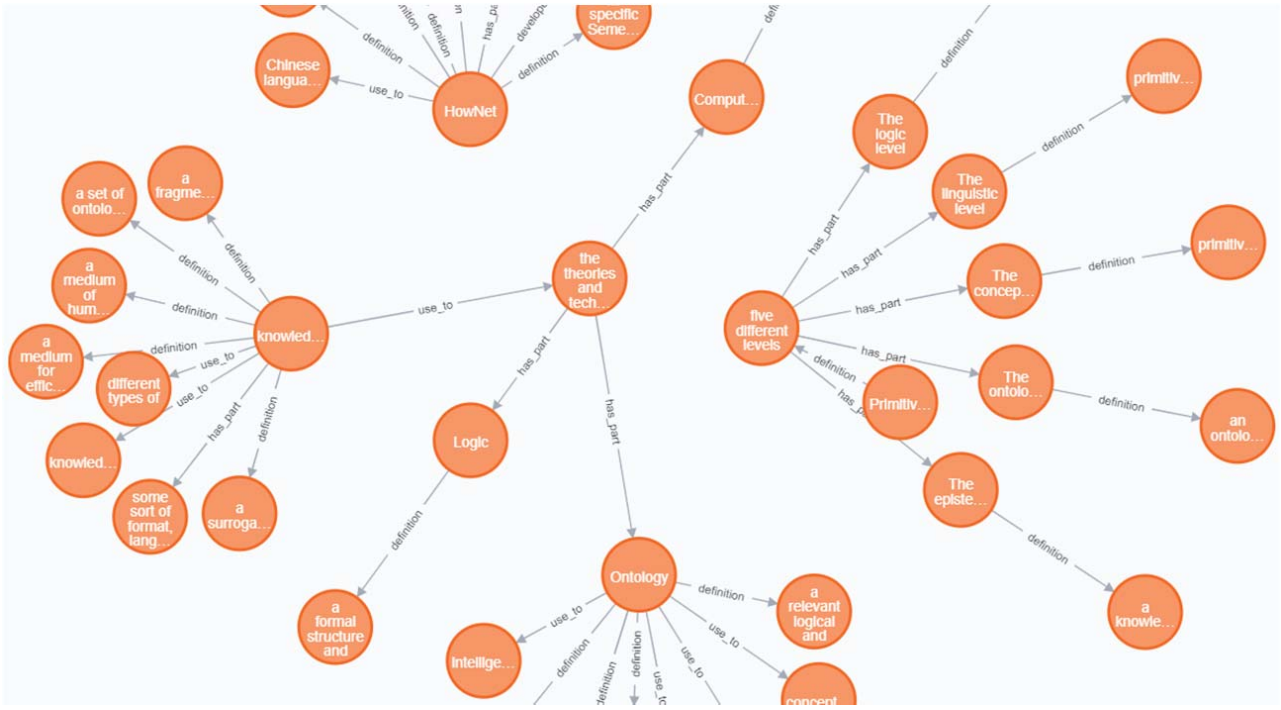

Fig. 21 Ontology Graph

C. QA Agent

The system has generated a knowledge graph after reading a book in learning agent, and the tutor agent will answer questions according to knowledge graph. The process of the

agent receives inputs and generates output as shown in Fig. 22.

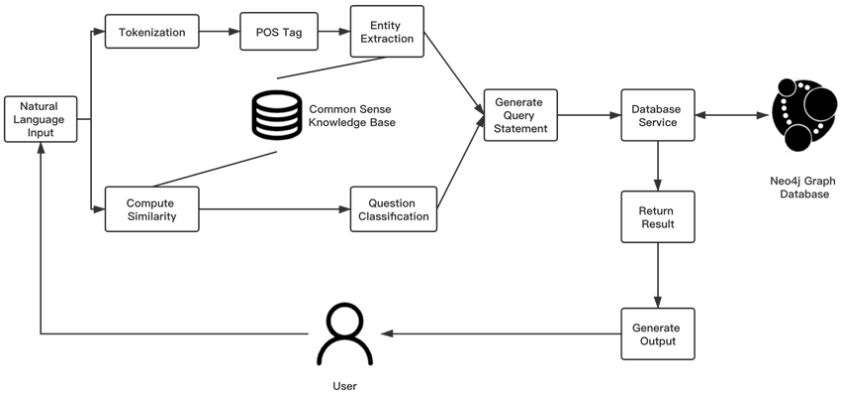

Fig. 22 QA Agent

The QA system is developed in the model layer. The input sentence will be processed in two directions: one is to extract sentences' entities; the other is to detect the question type. In order to classify the question type by interrogative words, a specific knowledge base was created to store common sense, such as the phrase "what is" means the definition of something, the word "why" represents the reason etc. The sentences will be split into words and calculated the words frequency in each sentence (the input sentence and the phrases in common sense knowledge base). The sentences have become two vectors: v1 and v2. After that, the system finds the corresponding word frequency of common string between two vectors notated as Common1, Common2, and computes the cosine similarity by (1). After calculating input sentence's similarity and phrases in knowledge base, the system will figure out type of phrases with the highest cosine similarity. More specifically, the input sentence has been labeled the type of sentence used to match the answer in knowledge graph later. Simultaneously, the agent will extract the entity from input sentence. Java OpenNLP API was used to tokenize the sentence, assign a POS tag to each token, and filter out the sentence's nouns as entities. The result of entity extraction and question classification is shown in Fig. 23.

$$\frac{\sum(Common1*Common2)}{\sqrt{\sum v1^2}*\sqrt{\sum v2^2}} \quad (1)$$

```
&gtWhat is the definition of knowledge?
What is the definition of knowledge?
WP VBZ DT NN IN NN . definition
knowledge
definition knowledge
0.9128709291752769 definition
```

Fig. 23 Entity Extraction and Question Classification

The agent obtained the entity and question type, then a Cypher query statement (a query statement like SQL in Neo4j) will be generated by the pattern. Neo4j Java API was used to connect Neo4j database to execute the generated Cypher query to match the corresponding nodes and relationships in knowledge graph and return the properties of the matched entities containing the name, contents of the nodes to the agent. The agent generates answers according to the return data and transmits them to the controller and viewer to display results to users as shown in Fig. 24.

Fig. 24 displays some questions related to content of the book which the agent has read. Since common-sense ontology stored the knowledge of interrogative words, the agent can detect question's type accurately from users even they ask questions in different ways. For example, "What is the definition of" and "What is the meaning of" means users want to ask an object's definition.

V. Conclusion

Our project is an artificial intelligent tutor that can serve as a teacher in school. To achieve the knowledge understanding function and answer generating function, we implemented a JADE platform project which can manage learning and QA agents. In the learning agent, Spacy and OpenNLP toolkit were used to perform the NLP tasks. Further, Spacy was used to match and extract SPO triples from a text. The extracted SPO triples will be constructed as an ontology, which we apply Neo4j, a graphical database was used to store and represent knowledge. In QA agent, MVC framework was used to receive question and output answer. The cosine formula was also implemented to classify processed question resulting the accuracy of querying information was significantly improved. In summary, we found that the proposed architecture is an appropriate research to support teaching assistance and will have great potential for further development. Source code can be accessed via Github: https://github.com/Pakhofan/AI_Tutor.

VI. Future Work

At present, our project is an elementary system. We will apply more advanced technologies to promote accuracy and efficiency. For example, a neural network will be implemented to improve the ability of questions classification and apply more querying algorithms to better deal with complex questions. We will also collect more data to enhance defined matching patterns to improve extraction performance. Further,

we will implement a powerful, intelligent agent to accurately recognize text's structure which can generate more undefined matching patterns based on accumulation automatically and efficiently.

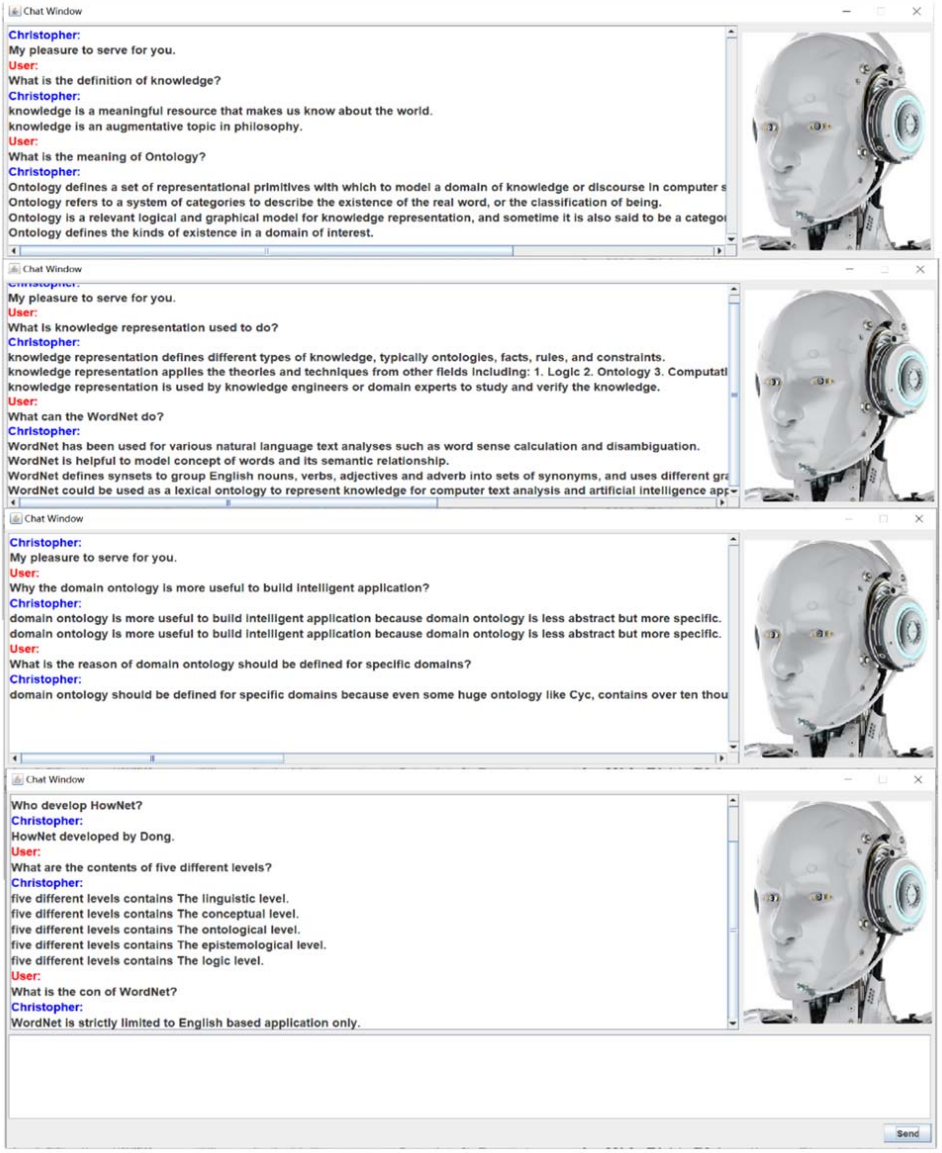

Fig. 24 The GUI of AI Tutor

ACKNOWLEDGMENT

The authors would like to thank for UIC DST for the provision of computer equipment and facilities. This project is supported by UIC research grant R202008.

REFERENCES

[1] R. S. T. Lee, *Fuzzy-Neuro Approach to Agent Applications.* Berlin: Springer, 2006, pp. 56-68.
[2] Z. Memon, A. H. Jalbani, M. Shaikh, R. N. Memon, and A. Ali, "Multi-Agent Communication System with Chatbots," *Mehran University Research Journal of Engineering & Technology*, vol. 37, pp. 663-672, July 2018.
[3] F. Bellifemine, G. Caire, and D. Greenwood, *Developing Multi-Agent Systems With JADE.* England: John Wiley & Sons Ltd, 2007.
[4] B. Hettige and A. S. Karunananda, "Octopus: A Multi Agent Chatbot," in *Proc. 8th International Research Conference, KDU*, 2015, pp. 41-47.
[5] R. Kumar and C. P. Rosé, "Architecture for Building Conversational Agents that Support Collaborative Learning," *IEEE Trans. Learning Technologies*, vol. 4, pp. 21-34, January-March 2011.
[6] E. H. Y. Lim, J. N. K. Liu, and R. S. T. Lee, *Knowledge Seeker- Ontology Modeling for Information Search and Management.* Berlin: Springer, 2011.
[7] R. M. Reese, *Natural Language Processing with Java.* Birmingham: Packt, 2015.

Yingqi Cui graduated from Bejing Normal University – Hong Kong Baptist University United International College (UIC) awarded B.Sc. (Computer Science and Technology) from Hong Kong Baptist University in 2021. His research interests covering Natural Language Processing, Deep Learning and Neural Network and Recommendation System.

Changran Huang graduated from Bejing Normal University – Hong Kong Baptist University United International College (UIC) awarded B.Sc. (Computer Science and Technology) from Hong Kong Baptist University in 2021. His research interests cover Natural Language Processing, Mini-program in Wechat platform.

Raymond Lee attained his B.Sc. (Physics) from Hong Kong University in 1989, M.Sc. (IT) and PhD (Computer Science) from Hong Kong Polytechnic University in 1997 and 2000 respectively. Dr Lee had worked at the Department of Computing of Hong Kong Polytechnic University as Associate Professor 1998 - 2005. During the past 20 years, Dr. Lee has published over 100 publications and the author of 8 textbooks and research monographs covering the fields of artificial intelligence, quantum finance, e-commerce, pattern recognition, intelligent agents and chaotic neural networks. Dr. Lee is now the Associate Professor in Beijing Normal University-Hong Kong Baptist University United International College (UIC).