Ontology-based Inference for Information-Seeking in Natural Language Dialog System

Abstract – Many natural language dialog systems have been developed with Relational Data Base (RDB) as a machine-readable knowledge source. However, RDB has some problems for answering the questions which need complex domain-specific information. In addition to the typical problems of RDB such as dependency and redundancy problems, limitations of meaning representation and storing various domain knowledge problems also exist. To solve the problems of RDB, we adopted ontology concepts as a knowledge representation method. Ontology knowledge has some advantages about representing and querying information. In this paper, we developed ontology-based approach for information-seeking to improve traditional RDB-based natural language dialog systems. This is more flexible than RDB-based dialog system for answering to complex questions. To implement our system, we designed hand-crafted ontology schemas for an Electronic Program Guide (EPG) application and we populated ontology instances from web pages semi-automatically. We believe that our preliminary evaluations show the possibility of our new system.

I. INTRODUCTION

As the needs of human-machine interface have increased, many natural language dialog systems are developed. Most of previous systems for information-seeking generally can respond to the user inputs with Relational Database (RDB). The RDB is composed with simple relational tables which have domain knowledge to generate answers in information-seeking tasks. However, with only RDB, they cannot answer to complex questions because they need complicated knowledge inferences.

To overcome the limitations of RDB, there has been substantial previous work on ontology-based interactive question answering (QA). Some interactive QA systems already have been used domain-specific ontologies to support the answering to complicated questions. Ceusters et al. suggested that the ontology could take a role to develop question answering systems, especially for question analysis and answer processing [1]. AquaLog adopts triple-based data model [2]. The triple-based model is adequate to represent queries and knowledge base. CuteAid use topic ontology concept to build content database [3]. In this system authors define their own ontology format which is relatively simple compared with general Resource Description Format (RDF)-based ontology.

In this paper, we also propose a natural language dialog system that can infer answers from ontologies as domain knowledge for information-seeking of complicated questions which cannot be answered with simple domain information. The remainder of the paper is as follows. Section II presents the previous system architecture that uses RDB-based knowledge base and its limitations. Section III presents our new system which is based on ontology domain knowledge. Section IV gives an inference process that is realized on the proposed system and example scenario. Section V shows implementation issues. Section VI presents the experiments and analysis. Section VII discusses the conclusion of the paper and future works.

II. TRADITIONAL RDB-BASED DIALOG SYSTEM

A. Overall Architecture

Most of traditional dialog systems are using RDB as their knowledge base. The natural language dialog systems typically consist of the main components shown in Fig. 1. When the user speech input is entered, the spoken input is first processed through Automatic Spoken Recognition (ASR) module and the textual input is analysed by Natural Language Processing (NLP) (e.g. morphological analysis and part-of-speech tagging). Then, Natural Language Understanding
(NLU) module maps the pre-processed input to a meaning representation (e.g., semantic frame) in which speech act, main goal and domain-specific named entities are extracted by semantic parser or statistical model. After extracting the semantic frame, Dialog Management (DM) module serves many roles which include discourse analysis, knowledge database (e.g., RDB) query, and system action prediction based on the semantic frame and discourse history. Finally, Natural Language Generation (NLG) and Text-To-Speech (TTS) module generates the system utterances with system actions and RDB results. Among these modules, the DM is one of the central components within natural language dialog system. The major role of the dialog management is to select correct system actions based on observed evidences from the results of NLU. In addition, the domain-specific knowledge is required to generate the right system response at NLG module. Traditional RDB-based dialog systems store this knowledge as RDB0 tables and access with Structural Query Language (SQL) query.

In our previous work, we also developed RDB-based dialog system, POSSDS-EPG [4], as shown in Fig. 2.

B. Problems of RDB-based Approaches

Traditional natural language dialog systems have used RDB as a domain-specific knowledge database. However, it has some problems to reason about complex questions. First is the dependency problem. When new elements or features are inserted into RDB, we should consider dependencies between tables and columns. This problem is also occurred for the deletion case. In general, RDB has plural tables and each table has many elements and columns. Thus, inserting or deleting new tables or columns with maintaining consistency is not easy. Second is absence of meaning information. In RDB schema, we can just guess the meaning of each column from its label because RDB cannot represent the meaning information between tables and columns. The structural relations of tables and columns cannot be also found in RDB. In other words, RDB schema is not intuitive to represent the relations of objects. Third is the redundancy problem. As more tables are inserted, redundant elements are exist as a necessity. Finally, there is complicated SQL problem to search. Because the statement contains many table joining and it needs the knowledge about the relations and dependencies between tables and columns.

III. ONTOLOGY-BASED DIALOG SYSTEM

To address the above problems of traditional RDB-based dialog systems, we use domain-specific ontologies as a representation method for domain knowledge. The ontologies are based on RDF format which has subject, object, and predicate [5]. Predicate means the relation between subject and object. This RDF forms ternary tuple <subject, predicate, object>, and a graph can be represented with these tuples. Fig. 3 shows an example of a graph representation of ontology on movie and TV program domain.

A. Advantages of Ontology

Due to the distinctive features of ontology, many limitations of RDB can be solved by using ontology-based knowledge source.

Basically, an ontology consists of several tuples. Therefore, when we want to insert some new elements, adding new tuples is enough regardless of any dependencies. In our approach, constructing ontology schema in ontology representation is very important task. The design of ontology schema can be thought as a hidden cost applying our approach. However, once the ontology schema is well-designed, then the process of inserting tuples is quite easy.

Other advantage for using ontology is that the relation and classes of objects can be efficiently represented by ontology. The expression power of ontology is superior to that of RDB because the classes and relations have hierarchical structures. In ontology, classes have the structure such as union, complement, and intersection. Relations have properties such as transitive, symmetric, functional, and inverse relations. With these hierarchies, ontology can have strong semantics.

In addition to above advantages, ontology query is more efficient than RDB query. Ontology querying does not need to join several tables because the ontology inference can be executed with considering the structure of relations and classes.

Consequently, the higher expressive power and meaning representation enable ontology to replace RDB as new domain
Knowledge successfully.

B. Building Ontology

To develop ontology-based systems, there are two steps: 1) building ontology schema and 2) instance population.

As mentioned above, ontology has a graph form which has many (subject, predicate, object) tuples. Ontology schema is the basic frame of the ontology. Thus the classes and relation structures are defined on the schema. We construct the basic TV program ontology schema and other schemas according to the detail domains (e.g., drama, sports, news). All schemas are designed manually. Fig. 4 shows an example of ontology schema used in our system.

Next, we should populate ontology instances on its schema. Instance population means the process of inserting instances on the ontology schema. In other words, inserting the real objects and their relations according to the pre-defined frame is the task of instance population. Instance population is important work in ontology building. Although building ontology schema is done manually, populating instances is done easily by semi-automatic information extraction as following procedures. First, we extract instances from Electronic Program Guide (EPG) web site. The web resources are preprocessed and transformed to the two objects and binary relation form using pattern-based sequence alignment method [6]. After that, the extracted information of objects and relations is converted Web Ontology Language (OWL) format [7] and inserted into the ontology schema. Fig. 5 shows the instance population. Fig. 5-1 is the instances which are extracted from the web resources. They are converted to OWL format as shown in Fig. 5-2 and populated into ontology schema. Fig. 5-3 shows a part of the populated instances.

IV. INFERENCE PROCESS

After building the ontologies, Knowledge Manager (KM) module accesses domain-specific ontologies to obtain domain knowledge. In this process, an inference engine confirms reasoning on the ontologies. We explain the whole progress of inference process in this section.

When the ASR output is transferred to SLU module as user input, SLU detects the domain-specific semantic slots and the user intention (e.g., dialog act and main goal). Then, according to SLU results, DM predicts the next system actions given discourse history and the current semantic frame. In addition, KM generates query statement to acquire information for answering to the user question. With the generated query statement, the inference engine executes inference process on ontologies. The inferred results and system actions are transformed to system utterance by NLG module. The overall flow of ontology inference progress is shown in Fig. 6.

A. Example Scenario

---

Fig. 4. An example of ontology schema.

---

Fig. 4-1. Instances (objects and relation) to be populated

<table>
<thead>
<tr>
<th>Class: Actor</th>
<th>Class: Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lily</td>
<td>Kate</td>
</tr>
<tr>
<td>Fox</td>
<td>Jack</td>
</tr>
<tr>
<td>Kim</td>
<td>Sun</td>
</tr>
</tbody>
</table>

Fig. 5. Converting to OWL format.

Fig. 5-1. A sample of populated instances.

---

1 EPG web site: http://www.epg.co.kr
We show an example scenario which needs an inference process. To answer for the request, “Let’s watch Wayne Rooney.”, the system should know that Wayne Rooney is a football player and whether there is a football match in which Wayne Rooney plays now. The previous system that uses RDB as knowledge base is impossible to make reasonable answers because RDB (TV program schedule DB) does not contain the specific domain knowledge such as sports team members or player information. In our system, NLU extracts the named entity and its semantic slot, “Wayne Rooney” and <person_name>, and also detects the main goal, “search_channel”. Then KM tries to search the person name at many domain-specific ontologies (e.g. actor ontology, sports player ontology, singer ontology). After KM find knowledge that “Wayne Rooney” is a football player from sports player ontology, it generates query statement and the inference engine infers with the query that there is a football match in which Manchester United Football Club takes part and Wayne Rooney is a member of it. Then KM returns the match information to the NLG and NLG generates the system utterance with the system actions and the inference results. Fig. 7 shows the example scenario.

V. IMPLEMENTATION

We use Web Ontology Language (OWL) as popular ontology language to build ontology as domain knowledge [7]. OWL is one of the ontology representation languages for describing the classes and relations as shown in Fig. 5-2. Among the three levels of OWL, We adopt OWL Description Logics (DL) to support description logic level inference. The <subject, predicate, object> tuple can be expressed very well with OWL because OWL is a kind of vocabulary extension of RDF.

To query the ontology, we use SPARQL, one of RDF query language, which provides compatibility with OWL-based ontology schema [8]. SPARQL is designed for querying the triple-based graph data format.

The inference engine is one of the important modules in our system. We take Pellet as the inference engine [9], [10]. Pellet is open software engine for research purpose and supports SPARQL querying and ontology consistency check. 

Our ontology-based framework was integrated into the previous dialog system, POSSDS-EPG [4]. To support inference processes, we defined more main goals, semantic slots of named entities, and query statement generation rules. The details of the additional elements are shown in Table I.

VI. EXPERIMENTS

A. Experiments Setting

Evaluation of ontology-based dialog system is problematic as it deals with a large set of factors(e.g. noisy environment and error propagation), all of them strictly depending on specific applications. In this section, we try to design some experiments to evaluate our system by humans. First, we made 10 EPG tasks to be done by real users. All of them cannot be completed with previous RDB-based dialog system. It needs to process ontology inference with ontology knowledge. Each task is supposed to be successful if a user achieves its goal. The 8 test volunteers participate in our experiments. We do not provide any guide for novice user.
TABLE I  ADDITIONAL ELEMENTS FOR ONTOLOGY-BASED SYSTEM IMPLEMENTATION

<table>
<thead>
<tr>
<th>Type of elements</th>
<th>POSSDS</th>
<th>Proposed system</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main goals</td>
<td>13</td>
<td>17(+4)</td>
</tr>
<tr>
<td>Semantic slots</td>
<td>7</td>
<td>15(+8)</td>
</tr>
<tr>
<td>Query generation rules</td>
<td>-</td>
<td>23</td>
</tr>
</tbody>
</table>

For each task, the user can enter utterance and receive system response at maximum 5 turns. If the turns are over 5, then the task is determined as failure. We evaluate whether each task is success and the Task Completion Rate (TCR) for it. Also we record the total turns used per each task. After finishing the experiment, the participants are asked for their thought about our proposed system. The questionnaire is rated with five star scales and they are acquired from [11]. The questionnaire is appended at Appendix. Because we focus on the ontology-based inference performance, we do not consider the performance of ASR and TTS.

B. Results and Analysis

Table II shows the results of experiments. The average user turns for all tasks is 3.0 and the average user turn for success tasks is 1.2. This means that not more than 2 utterances are needed to complete the user’s goal. For success tasks users takes very few turns. For fail tasks the average user turn is 5 because we limit the maximum turns as 5. TCR for the proposed system is 0.64. This result cannot be compared directly with other researches because we consider only the complicated tasks which cannot be solved by the previous RDB-based dialog systems.

The questionnaire results are given in Fig. 8. This results can be thought as the user satisfaction (USAT) evaluation. When the USAT is transformed in the five star scales, the average of our system is 2.93. We think that this result is acceptable because our system should execute complex inference to search information of interest. From “Task Ease” and “User Expertise”, we can guess that the users could select their next utterance and get information which they want easily. On the other hand, relatively low scores are found at “Interaction Pace” and “Future Use”. The failure cases of tasks might cause some users’ dissatisfaction.

VII. CONCLUSIONS

In this paper, we introduced the ontology inference concept to our natural language dialog system for information-seeking. Traditional RDB-based knowledge is totally replaced to ontology-based knowledge. For ontology building, we designed ontology schemas manually and populated ontology instances semi-automatically. The KM could infer on the domain-specific ontology to answer the complicated questions. The evaluation results show the possibility of our approach and necessity of future work to improve our system.

We think that we need more scenarios which can be done with more detail inference processes. This allows us to broad the coverage of question answering. And we can reduce some SLU errors which are caused with more utterance patterns by collecting more SLU training data. As constructing more robust dialog system for tasks which can be solved with inference processes, the user satisfaction will be improved.

ACKNOWLEDGMENT

This research was supported by the MKE(Ministry of Knowledge Economy), Korea, under the ITRC(Information Technology Research Center) support program supervised by the IITA(Institute for Information Technology Advancement) (IITA-2008-C1090-0801-0045).

REFERENCES

APPENDIX

The questionnaire:
1) Task Ease: In this conversation, was it easy to find the message you wanted?
2) Interaction Pace: Was the pace of interaction with the system appropriate in this conversation?
3) User Expertise: In this conversation, did you know what you could say at each point of the dialogue?
4) System Response: How often was the system sluggish and slow to reply to you in this conversation?
5) Expected Behavior: Did the system work the way you expected him to in this conversation?
6) Future Use: From your current experience with using the system to get your email, do you think you’d use the system regularly to access your mail when you are away from your desk?