Configuration of Dialogue Agent with Multiple Knowledge Sources

Ridong Jiang, Rafael E. Banchs, Seokhwan Kim, Luis F. D'Haro, Andreea I. Niculescu, Kheng Hui Yeo
Department of Human Language Technology, Institute for Infocomm Research, Singapore 138632
E-mail: {rjiang, rembanchs, kims, luisdhe, andreea-n, yeokh}@i2r.a-star.edu.sg

Abstract— **Knowledge base is a key component of a dialogue agent which determines its usability, performance as well as intelligence. Usually one type of knowledge source is tailored towards specific application or task. One challenge for developing multiple-domain dialogue agents is how to represent the domain knowledge in their respective ways for the best efficiency and easy construction of queries from natural language. There are a variety of knowledge sources available for building up dialogue agents such as relational database, ontology, knowledge base (KB), search index, artificial intelligence markup language (AIML) and the World Wide Web, etc. In this paper, we present a systematic way of configuring dialogue agents supported by multiple knowledge sources. With the proposed dialogue framework, we provide a complete solution for the development of various dialogue systems with the support of different dialogue management techniques, multiple protocols for cloud-enabled dialogue services, pluggable infrastructure for component reuse and service enhancement, strong scripting functions in both XML and Python. We have constructed various dialogue agents with this framework. Results show that these agents are robust and the development cycle can be considerably reduced.**

I. INTRODUCTION

Speech interfaces are an intuitive, flexible and natural means of communication between users and machines. This is especially true for small devices like hand phones and hand-hold intelligent devices where clicking, scrolling and typing with fingers are not convenient. With the advancement of natural language processing technologies, more and more voice enabled applications and products are emerging in our daily life. The well-known applications include Apple Siri – an intelligent personal assistant, Google Now – a virtual assistant, Microsoft Cortana – a voice based virtual assistant, Nuance Nina – an intelligent multichannel virtual assistant, etc. Successful voice enabled commercial applications are backed by rich knowledge sources which might contain local restaurants and weather forecast information, the traffic on your commute home, or when your flight is about to take off. For instance, Google Now is powered by its knowledge graph; Cortana draws knowledge from its search engine Bing while Siri taps into Wolfram Alpha knowledge base and other third part services. A key challenge in building such a system is how to handle the knowledge base behind the scene [1]. The competition between these kinds of systems is in fact very much depending on their knowledge bases because knowledge base is the fundamental ingredient which determines the system's usability, coverage, performance as well as the intelligence. Of course, understanding of semantics and the context and then translating the natural language into the correct query also play a very important role in the development of chatbots and dialogue agents.

Depending on the requirements and application domain, dialogue agents can support a broad range of applications in business enterprises (e.g. Customer Service, E-commence, Tour guide), education (e.g. E-learning, tutor, speech therapy) [2], healthcare (e.g. Appoint booking, medical question answering) and entertainment (e.g. game, joke telling, chatbot) [3]. Usually one type of knowledge source is tailored towards specific application or task. For instance, AIML is widely used as knowledge base for chatbots which are usually modeled for providing chatting functions. For question answering, it is quite common to treat it as a task of information retrieval which is based on vector space model, where questions and answers are represented in question answer pairs and the document is represented as a vector of index terms [4, 5]. DBpedia, Freebase and various ontologies have become very practical and successful in representing real-world entities like people, places and things across domains. These structured knowledge bases stored in Resource Description Framework (RDF) can be retrieved and manipulated by SPARQL query language. VoiceXML is commonly used in many industries with a variety of successful commercial applications where information and dialogue is modeled as finite-state automata or form filling (frame-based dialogue model) [6]. In addition, knowledge sources for dialogue agents can also be any other forms such as relational database [7, 8], First Order Predicate Logic [9], as long as they are expressive enough and suitable for these applications.

In this paper, first we discuss the task and domain representation with different knowledge sources (section II). Next, we present the proposed configurable system architecture for multi-domain dialogue system (section III). Then, we describe the construction process with a case study (section IV). Finally we review some related work (section V) followed by conclusion and future work (section VI).

II. DOMAIN AND KNOWLEDGE SOURCE

Every response from the dialogue agent needs information from system knowledge source or dialogue model. For instance the information "London is the capital of Great

Britain" can come from any kind of representation of knowledge base such as an ontology. While the confirmation and repeating of prompts may be generated by dialogue models. For the case of information from knowledge base, it can be viewed as a process of information retrieval. This process can broadly be divided into two steps: first mapping the natural language into a query statement or predefined question answer pairs either through pattern matching or information indexing; next getting the desired information from knowledge source based on the mapping results by executing the query statement or picking the answer from mapped question answer pair. There are several application scenarios when handling domain models and their knowledge sources.

A. Single domain and single knowledge source

Every domain model holds knowledge of the world that is talked about [10]. The simplest case is that the dialogue task involves only one domain and one source of knowledge as shown in Fig. 1.

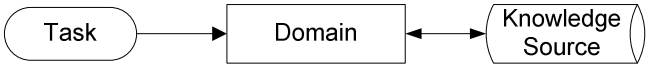

Fig.1 Single domain – single knowledge source

For this single domain with a single knowledge source scenario, the knowledge source fully represents the domain to provide all possible information services for the specific dialogue task. A typical example is natural language question-answering system that uses a file of frequently asked questions (FAQ) as its knowledge base [11]. These kinds of systems first try to compile a list of question answering pairs or directly extract FAQ from web sites and save them into a file to create the knowledge base for the dialogue system [12]. The FAQ file is the only knowledge source for the information query. Next, the answer retrieval can be carried out by using the vector space model to calculate the similarity between the user utterance and the FAQ pairs in the knowledge base. Hence for the construction of FAQ based dialogue agent, it is necessary to have an indexing module which is based on the vector space model for information retrieval. In addition, authoring tools for extracting FAQ as well as creating index for information retrieval will definitely be helpful.

Another example which can be considered to fall into this category is the chatbot. The chatbot relies on the pattern matching knowledge base such as AIML. Chatbots are very popular virtual agents with various applications in E-commerce, entertainment and education, etc. [13]. In this case, the single knowledge base is AIML which is composed of a number of knowledge units – categories. Every unit contains question pattern and an answer template. For the configuration of AIML based chatbots, an AIML interpreter must be provided. There are many AIML interpreters available under a free or open source license.

B. Single domain and multiple knowledge sources

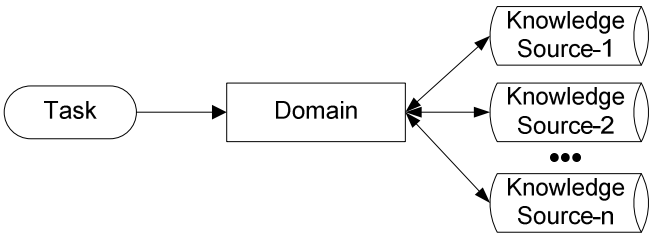

Fig.2 Single domain – multiple knowledge sources

A more complicated scenario is a dialogue agent that is able to respond to questions in one domain but needs the support of multiple knowledge sources as shown in Fig. 2.

One good example of this scenario is the receptionist agent [14]. The task of a receptionist commonly includes greeting visitors and callers (greeting knowledge), answering visitor inquiries about the company (company information knowledge), and directing visitors to a place within the company as well as other facilities in the office building such as coffee shops, food courts, supermarkets, etc. (facility information knowledge). The different types of knowledge required in this case could be best represented in different forms. Greeting knowledge can be naturally rendered as AIML. Company information knowledge may include a lot of FAQ about the company and hence vector space model based information retrieval would be ideal for representing and handling this type of knowledge source. For facility information knowledge, there are many attributes to be described. For instance, name of a restaurant, its unit number and floor number in the building, category, description and contact number could be useful for the receptionist. It will be very efficient to use a relational database to store all the information especially when there are a lot of facilities in a big office building.

To construct a dialogue agent in a single domain with multiple knowledge sources, different modules for handling the respective knowledge sources are essential. A good practice is to reuse the relevant modules which are developed for single domain and single knowledge source as discussed in previous section. This will greatly help reducing the system development cycle. However, a function module which is able to coordinate the use of different knowledge sources must be developed. The key issue to be resolved is how to identify the right knowledge source to get the desired answer. The simplest way of processing is to sequentially try out all knowledge sources by the order of information contained. The more complex way could be using a classifiers to identify the correct knowledge source based on the user's input.

C. Multiple domain and single knowledge source

In contrast to the single domain with multiple knowledge sources as discussed in the last section, multiple domains with single knowledge source as shown in Fig. 3 is also quite popular in today's dialogue agents.

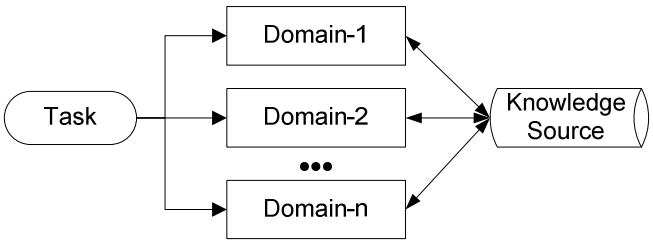

Fig.3 Multiple domains – single knowledge source

A typical example for this scenario is to use ontology as knowledge repository to serve multiple domains. Here we refer to the case where ontology is the only knowledge source for the dialogue agents although multiple ontology files may exist in the system. There is a flux of ontologies on the internet and the total number of which is still rapidly increasing in recent years. Many ontology based dialogue systems have been developed and quite a number of them deal with question answering [3, 16-17]. The popular data model for storing objects defined in the ontology is the graph model, Resource Description Framework (RDF), which represents data as a collection of triples in the form of "subject-predicate-object". Standard SPARQL language is used to retrieve and manipulate data stored in RDF. To construct the dialogue agent with knowledge base in format of RDF triplestores, one particular big challenge is to derive a structured query from a given natural language question. The user needs to know the SPARQL query language as well as the ontologies used to express the triples he/she wants to query on. In addition, response time is also a crucial issue, especially when there is enormous amount of data in the store.

D. Multiple domain and multiple knowledge sources

The most complicated dialogue agents would be the ones dealing with multiple domains with multiple knowledge sources as shown in Fig. 4.

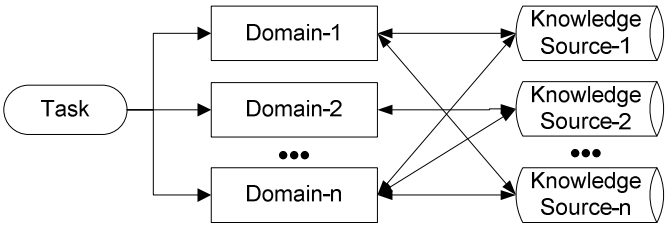

Fig.4 Multiple domains – multiple knowledge sources

In the domain and knowledge source relationship, one domain could have one or more knowledge sources. In the meantime, one knowledge source could be shared by multiple domains or tasks. For instance, an AIML based greeting knowledge base can be commonly shared by different agents or the same agent when working in different domains.

The typical examples are various intelligent personal assistants such as Apple Siri, Google Now, etc. Using Siri as example, it retrieves information from multiple knowledge sources depending on user's question and the context. For instance, if the question is related to mathematical computation or scientific fact, the answer would likely come from Wolfram Alpha. For movie listings and other movie related information, the answer could be from Movie Tickets and Rotten Tomatoes. There are many challenges for construct dialogue agents for multiple domains with multiple knowledge sources [18, 19]. Besides the challenges as stated in previous sections on single domain-single knowledge source and single domain-multiple knowledge sources, the system also has to spot the right domain and identify the correspondent module to process the user's input.

III. SYSTEM ARCHITECTURE

In order to make the system configurable for multiple domains and multiple knowledge sources, we take an object oriented approach, loose component coupling, event-driven communication paradigm and plug-and-play strategy to design and develop the proposed dialogue framework. All communication components and knowledge base manipulation modules are reusable and extensible. The overall system architecture is shown in Fig. 5.

The system is broadly composed of following functional blocks: user interface, communication middleware, Apollo spoken dialogue management engine, natural language process components and knowledge manipulation components.

A. User Interface

The user interface is the client of the dialogue service. It can be a web browser, an Android or iPhone app, or a computer program which can "talk" to the dialogue services in one of the supported communication protocols. The input can be either speech or text. One dialogue service can support multiple users for different dialogue tasks.

B. Communication Middleware

The communication middleware is the bridge to link the user interface with the spoken dialogue system through text. The supported communication protocols include WebSocket, Hypertext Transfer Protocol (HTTP) and TCP/IP. Messages are encoded in JSON format. All the supported protocols use the unified messages format for communication. Hence, switching from one protocol to another does not affect the message handling in both the client side (user interface) and dialogue server side. If low-latency and bidirectional persistent communication is required, WebSocket protocol can be selected (for web browsers, it must be a HTML5 compliant browsers). The HTTP protocol is supported by a HTTP plug-in. With this plug-in, the framework will turn the dialogue framework into a HTTP server providing dialogue service through commonly used HTTP request methods: "GET" and "POST". In addition, Restful dialogue service can also be configured through the supported HTTP protocol. TCP/IP is a convenient socket communication protocol which

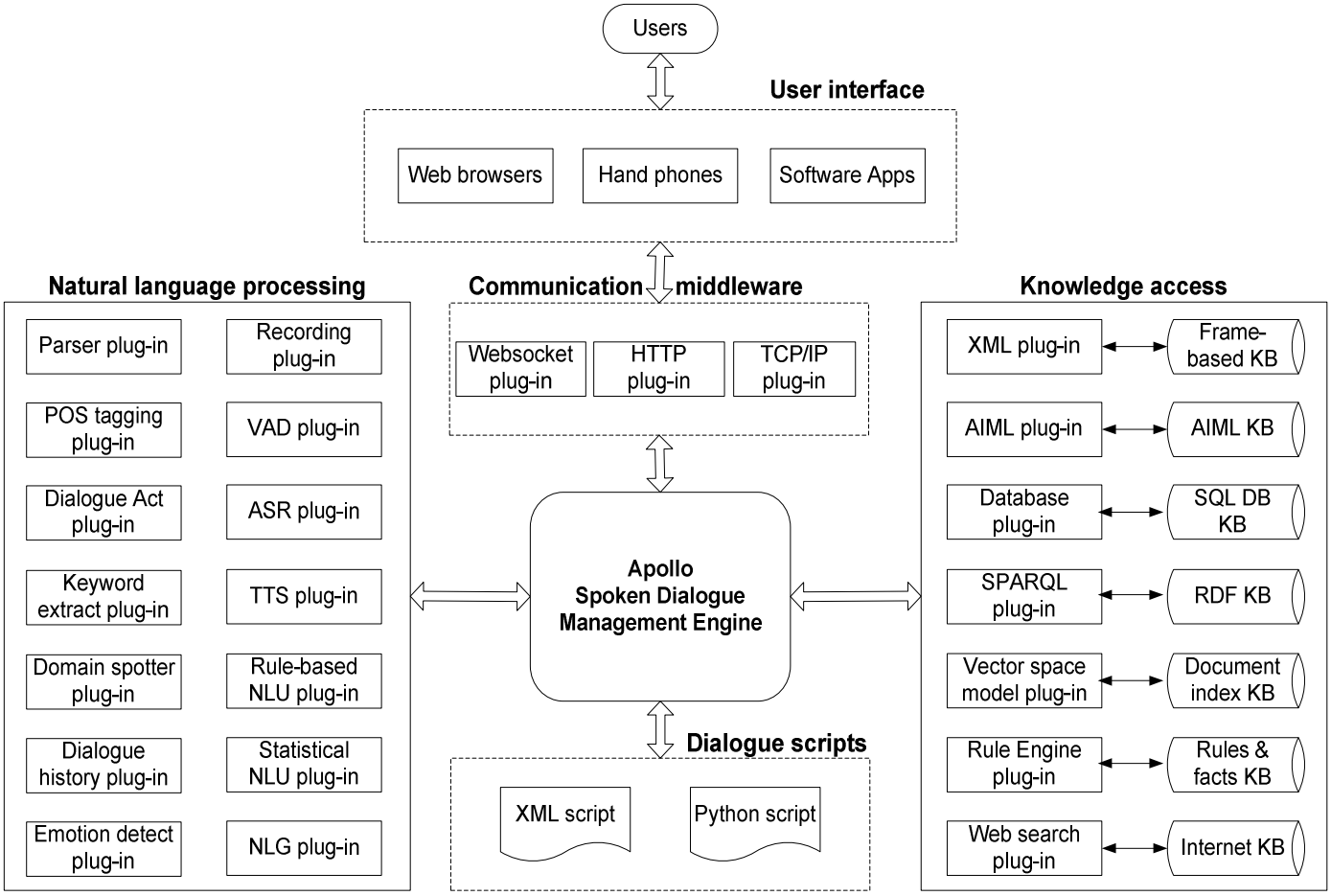

Fig.4 System Architecture of Apollo Dialogue Framework

is supported by most of the computer applications or hand phone apps.

The messages transmitted between the clients and the dialogue servers are text messages which are encoded in JSON format. The message sent to the server should include its action (the purpose of the message) and the parameters for this action. For instance, the following message is a normal query message sent to the dialogue server:

```
{
    " action " : " query "
    " parameters " :
    {
        " input " : " Good morning ",
        " type " : " normal "
    }
}
```

When the message reached the communication plug-in, one more field will be added to this JSON object by the plug-in. That is the unique client identifier (UCID). The UCID consists of client IP address and the socket number which is automatically assigned by the server. The UCID will be included in every message when the query is processed by the dialogue manager and related components. This is to facilitate the dialogue system to identify the right client when the response is produced by the framework.

C. Apollo Spoken Dialogue Management Engine

The dialogue management engine is the core functional module in the framework. It is composed of a message centre, a XML script engine and a plug-in manager.

The message centre is an event-driven messaging engine for the effective message routing. All messages from different sources are passing through the message hub. Every message is represented in a unified form and can be dispatched and handled in the same way.

The XML script engine supports XML scripting which can be used to drive the engine for state based (finite-state machine) and frame based dialogue management. In the meantime, the XML script is also used by the message centre for message manipulation such as sending and receiving messages. In addition, the dialogue management engine also supports Python scripting through Python plug-in development. This opens the door for the framework to make use of the rich resource of natural language processing developed in Python.

The plug-in manager is used to manage the plug-ins in the framework. The function includes the dynamic loading and unloading of plug-ins, the indexing of a plug-in by its type or

name. The system allows multiple plug-ins with the same type to work in parallel. For instance, it is possible for two different speech recognition engines work simultaneously.

Plug-ins development is one of the most important features of the Apollo dialogue framework. Currently, it supports the plug-in development in C++ and Python. All functional modules in the framework are plug-ins. Various plug-ins have been developed and can be reused for the construction of different spoken dialogue systems.

D. Natural Language Processing Components

The natural language processing modules are the fundamental components in building a spoken dialogue agent. For instance, automatic speech recognition (ASR), text to speech (TTS), natural language understanding (NLU) and natural language generation (NLG), etc. are basic elements when building a new dialogue agent. In addition, when further linguistic processing is needed on the user's utterance, for instance, syntactical analysis, semantic analysis, domain spotting, and emotion detection, the developed plug-ins as shown in Fig. 5 can be reused.

E. Knowledge Accessing Components

As discussed in section II about the domain and knowledge source, there exist different kinds of knowledge sources for the representation of domains. Different knowledge representations may have their own features and expressive advantages. For a complex dialogue system, given the heterogeneity of the system, a single knowledge representation and knowledge source may not be adequate to meet all the needs. We developed a number of knowledge accessing plug-ins for the support of different knowledge sources as shown in Fig. 5.

1) XML Plug-in: This is the system built-in component for the support of state-based and frame-based dialogue management. The task representation will be described in either different states or frames in XML depending on the dialogue management technique to be used. For instance, to represent a hotel booking task, a class object HOTEL can be defined in the XML. Under hotel class, we can define the required slots and their constraints: check-in date, check-out date, and room type (with "single", "double", "twin" constrains), etc.
2) AIML Plug-in: This module will be able to interpret AMIL and hence, AIML knowledge source is supported through this plug-in.
3) Database Plug-in: This is a plug-in for relational database access. There are cases when a relational database is the most suitable for information representation and storage, e.g. staff database of a big company. We have MySQL plug-in for MySQL database. For other relational database, we can select the ODBC (Open Database Connectivity) plug-in.
4) SPARQL Plug-in: A Python plug-in which is able to retrieve and manipulate data stored in RDF format. With the support of this plug-in, the dialogue framework will be able to query information from a public SPARQL Endpoint such as DBpedia.
5) Vector Space Model Plug-in: This module is often used for question answering with predefined question answer pairs. The engine uses the notion of a term space, where each example question is represented as a vector in a high-dimensional space. The engine projects a query into this term space and calculates the distance from the query vector to all the questions vectors in a dataset. Those example questions that are within a certain threshold distance are added to search result set and their correspondent answers are considered possible answers to the user's query.
6) Rule Engine Plug-in: This module is a full-fledged knowledge based system developed with the open source CLIPS – a tool for building expert systems [21]. The CLIPS rule engine is fully integrated into the dialogue framework with extended functions in knowledge bases for sending messages to the framework. The dialogue framework is also able to dynamically add facts or rules to the rule engine. With the support of this plug-in, the framework is able to do reasoning with rules defined in system knowledge bases.
7) Web Search Plug-in: A Python plug-in which is able to call search engine API such as Google Web Search API. Just like the search in web browser, this module accepts user's search keywords or sentence and sends it to search engine. Then using HTTP protocol it fetches web pages from search engine which are deemed as search results. Finally the titles, the content and related URL are extracted from the result web page. This module provides the dialogue framework an additional knowledge source using World Wide Web.

F. Dialogue Scripts

The dialogue script is part of the developed dialogue agent. As described in subsection C, the script can be XML, Python code or any mixture of XML and Python code.

IV. CONFIGURATION OF DIALOGUE AGENTS

With the proposed dialogue framework as well as the pool of reusable components, the effort needed for the construction of a new dialogue agent can be significantly reduced. There are a variety of supported knowledge sources for the quick construction of new dialogue agents. The business logic and communication between different components can be scripted in either XML or Python. With the help of this framework, dialogue agent developers can focus more on the data preparation. Furthermore, the quality and the robustness of the new developed dialogue agents are better than those agents developed from scratch.

A. Configuration Guidelines

Depending on the dialogue agent's requirements and knowledge sources available, followings are the general

guidelines and steps for the construction of a new dialogue agent.

1) Based on the dialogue agent's task requirements, first we have to decide whether the system is a single domain of multiple-domain application. What are the available knowledge bases and data? The requirement knowledge source should be single source or multiple sources as described in section II.
2) Investigate the best form of knowledge representation to support the agent's task. For instance, if the functions include answering of frequently asked questions, vector space model KB could be a good solution. If any ontology must be supported, then SPARQL KB should be adopted. If a task oriented dialogue function must be provided, for instance, flight booking, then the XML plug-in should be picked.
3) Determine the user interface of the agent then pick the right communication protocols. The questions to be asked include: whether the agent is a web based system? Is persistent communication required? Should multiple communication protocols be supported? If persistent connection is needed between the server and the client, Websocket protocol can be used. In order to support multiple users with different communication protocols, then all the three components can be selected and run simultaneously.
4) Pick all the other related components to support the system configuration in all last steps. If multiple domains must be supported, then domain spotter should be included. In case of a database is one of the knowledge sources, one important step is to translate the user utterance into a formal SQL query. Further natural language processing must be performed. Hence parsing, POS tagging and dialogue act classification can be included for analysis purposes.
5) Develop some XML script or/and Python script to coordinate all modules working seamlessly to fulfil the required tasks of the dialogue agent. This may include the pipelined information flow, different message handling, response JSON object formation, etc.
6) Test and fine tune the developed agent.

B. Case Study

We have successfully deployed a number of spoken dialogue applications using this configurable dialogue framework. The applications range from single domain dialogue systems such as chat-oriented dialogue agent [22], restaurant recommendation [23] to multiple domain dialogue applications [24, 25]. In this case study, we discuss the construction of a more complex multiple domain dialogue agent with multiple information sources. The domains, information source and related knowledge access modules are listed in table 1.

For the development of multiple domain dialogue systems, one crucial issue is how to properly identify the domain based on the user's utterance and current context and then select the right processing engine and corresponding knowledge base so that the correct answer can be produced. In order to find the clue for identifying the appropriate domain, the user input can be analyzed with domain-independent analyzers, such as linguistic analysis, keyword analysis, etc. [26]. The challenge for solving this problem is to maintain both robustness and domain extensibility because the classification problem needs the support of collected data [27]. Currently we combine keywords extraction and pattern matching for domain selection. Domain switching is allowed when a task oriented dialogue is ongoing (e.g. in the middle of hotel booking, asking questions about facilities).

TABLE 1 MULTIPLE DOMAINS AND THEIR KNOWLEDGE BASE

Domain	KB Engine	Knowledge Base
Self-intro	AIML Engine	AIML KB
Facility info query	SQL Engine	Database
Company info query	Vector Space Model Engine	FAQ index KB
Hotel booking	Frame-based Engine	XML KB
Chat	Vector Space Model Engine	Movie index KB

Self-Introduction: AIML knowledge base is widely used for creating various chatbots. It is self-contained and easy for editing with any text editing tool. It is straightforward to use AIML to represent some questions about the dialogue agent itself. There are only limited questions and question patterns about the dialogue agent itself. Hence, AIML is a practical and feasible knowledge representation of this domain. The self-introduction questions include questions about the name, age, the creator of the agent, etc.

Facility Information: The agent is able to answer queries on the facilities inside the office building it resides. We collected about fifty entries of data which can be classified into fifteen categories such as Restaurant, Sports and Fitness, Healthcare and Beauty, Learning Centre, Supermarket, etc. Every entry comes with information on facility ID, name, category, tags, unit number, floor number, description and contact number.

For the database query using natural language, one critical issue is how to translate the natural language into formal database query. This includes the identification of fields to be queried (the SELECT clause), the condition which constrains the query (the WHERE clause). Sometimes temporal information is included in the utterance. Hence time event identification is also necessary. Unlike template based AIML knowledge base query, further lexical, syntactical and semantical analysis is needed to detect the query field, as well as the query constraints. Take following query as an example:

Is there any restaurant servicing western food on the second floor?

Here the query field is the restaurant name. The constraints are "restaurant", "western food" and "second floor" which correspond to the fields of category, tags and floor number. The correct query statement must be:

SELECT name FROM facility where Category = 'Restaurant' AND Tags like '%western food%' AND Floor=2

Besides the linguistic processing on keywords extraction, dialogue acts detection, role labeling from parsing relation

tree, using dictionaries created from database fields to help the identification of database fields is also an efficient approach. However, paraphrasing on the contents of database fields is necessary for robustness purpose.

Company Information Query: The company information query is a list of FAQ about Institute for Infocomm Research. We collected about one hundred question-answer pairs from our company web site and other related sources. All the questions are frequently asked questions possibly by visitors. For instance, "When was I2R founded?", "How many departments are there in I2R?", "What are the research areas of the Robotics programme?", etc. In order to improve the robustness of the system, we expanded the questions to about three hundreds and created vector space model index with these question-answer pairs.

Hotel Booking: The hotel reservation system does not link to any global distribution system (GDS) for real hotel booking. We just use it as a test case for frame-based dialogue management with our XML frame-based KB engine.

Table 2 shows the frame representation of hotel booking. The slots include city name, check-in and check-out dates, room type (single, double or twin, etc.), the star-rating of the hotel, the location of the hotel in the target city as well as the hotel name. The hotel location and name are informational, they are not compulsory information that users need to provide. For all other slots, users need to indicate the information so that the system can complete the task. Otherwise the system will continuously ask questions to seek the necessary information.

Compared with state-based dialogue system, frame-based system is more natural, flexible and supports mixed-initiative communication. Information for multiple slots can be provided in one sentence and the sequence of provided information does not affect the slot filling.

Chat: The chat dialogue is implemented as a back-off system of the agent. When the domain spotter fails to identify the domain or task as described before, the chat engine will take the input and try to provide a response. The data collection used for this chat engine consists of dialogues from 153 movie scripts, mainly belonging to the comedy, action and family genres [22].

TABLE 2 HOTEL BOOKING REPRESENTATION

```
<class name="HotelBooking" topic="hotel">
  <slot name="CityName">                        // City name
  <slot name="CheckinDate"/>                    // Date of check-in
  <slot name="CheckoutDate"/>                   // Date of check-out
  <slot name="RoomType"/>                       // Single, double, twin
  <slot name="StarRating"/>                     // Five/four/three stars
  <slot name="HotelLocation" type="info"/>      // Location in the city
  <slot name="HotelName" type="info"/>          // Name of the hotel
</class>
```

With the selected domain knowledge sources and reusable related components, the dialogue agent was constructed and tested with questions from different domains. Sample conversion between user and the system is shown in table 3.

In various demonstrations, the multiple-domain dialogue agent is able to respond correctly on both task-oriented dialogue (frame-based dialogue in this case study) and question answering. In case of error, most of the time, it was caused by the incorrect selection of domain. If the system could not get any response from the wrongly directed domain, the system will pass the utterance to back-off chat agent.

TABLE 3 SAMPLE CONVERSATION FOR MULTI-DOMAINS

S/N	Utterance	KB
U1:	Hello	
S1:	Hi, nice to meet you.	AIML
U2:	How old are you?	
S2:	I am quite young	AIML
U3:	Who is your master?	
S3:	Researchers from Human Language Technology Department, I2R	AIML
U4:	Is there any Japanese restaurant on the second floor?	
S4:	We have 1 Japanese restaurant here; namely: Shin Nichi Japanese Restaurant. Shin Nichi Japanese Restaurant locates on the second floor.	Database
U5:	Where are you now?	
S5:	Uh - The Taft Hotel	Chat
U6:	I need to book a hotel	
S6:	Welcome to use Hotel reservation service, please tell me your destination city, the dates of check-in and check-out	Frame KB
U7:	A four-star hotel in Shanghai	
S7:	What is the exact date of your arrival?	Frame KB
U8:	The 24th of March	
S8:	And what type of room do you want, single bed, twin beds or double bed?	Frame KB
U9:	Single bed	
S9:	How long will you be staying?	Frame KB
U10:	I will be staying for two nights	
S10:	Thanks for using our service. You reservation has been made for …	Frame KB
U11:	How many departments are there in your institute?	
S11:	There are nine departments in our institute	FAQ
U12:	Where are you right now?	
S12:	And I was wondering if I could buy you a drink or something -	Chat
U13:	OK, bye bye	
S13:	Good bye	AIML

Alternatively, clarification strategies can be applied.

C. Advantages

The advantages of the proposed framework come from its configurability and component reusability. These features make the framework scalable for simple dialogue systems as well as complex multiple-source, multiple-topic dialogue applications. In addition, it is light-weight and has short learning curve. The development cycle for new dialogue agents can be greatly reduced.

Our experience with different users is that most users are able to develop a fairly complex dialogue agent in one to two weeks' time. This includes the learning of dialogue script (XML) and system configuration. The users only need to know XML because the system configuration, rule-based language understanding and dialogue control are all based on XML. This shows great advantage when comparing to some open source systems which typically need one month time to get the system working. In addition, the developed dialogue agents are robust because of the reuse of dialogue engines and components.

V. Related Work

Building a spoken dialogue system is a complex task. This is especially true for a multiple-domain dialogue agent. Quite a number of dialogue frameworks have been developed over the years for reducing the efforts of building new dialogue agents. Notable projects include Galaxy communicator based dialogue systems, information state update based system, Chatbot service based systems, etc.

Galaxy II communicator based systems include Olympus dialogue system [28], AT&T Spoken dialogue system [29]. The Galaxy Communicator software infrastructure is a distributed, message-based, hub-and-spoke infrastructure optimized for constructing spoken dialogue systems. Each component is implemented as a separate process that connects to a traffic router - the Galaxy hub. The hub script provides every limited programming capability. Another widely used dialogue system uses an information state update based approach. Trindikit [30] is a toolbox for building dialogue managers based on an information state and dialogue move engine. DIPPER [31] is implemented on top of Open Agent Architecture and it comes with its own dialogue management component. This component is similar to Trindikit. ISUbased systems and requires an open agent architecture for communication and a non-free dialect of the programming language Prolog for information state update and dialogue control [32]. Pandorabots web service makes available tools for building, hosting, and deploying chatbots using AIML [33]. There are hundreds of thousands chatbots have been developed as shown in its website.

Among the developed dialogue frameworks, only a few of them are targeting multiple domain dialogue applications because of their complexity. First the framework itself must be able to manage multiple domain agents, provide flexible communication capabilities between dialogue managers and domain agents, and embed a powerful scripting engine for domain and knowledge source configuration. In addition, strong natural language processing functions such as lexical analysis, syntactical analysis, semantical analysis, and spoken language understanding must be provided.

Nobuo Kawaguchi et al. proposed a distributed architecture for multi-domain spoken dialogue systems [34]. The key concept of the architecture is distribution and integration of data fragments. They promote the system extensibility and scalability. Kazunori Komatani et al. developed a spoken dialogue system that can handle user requests across multiple domains. Such systems tried to satisfy two requirements: extensibility and robustness against speech recognition errors [35]. O'Neil et al. implemented an object-based, cross-domain, mixed initiative spoken dialogue manager with Java [18, 36]. The system communication is based on the Galaxy software hub. Cheongjae Lee et al. proposed an example-based dialog modeling (EBDM) framework for a multi-domain dialog system to simultaneously manage goal-oriented and chat dialogs for both information access and entertainment. All domain knowledge sources are based on semantic-based indexing and querying [19]. Compared with the existing multi-domain dialogue framework, we provide a complete solution for various dialogue systems development with different dialogue management techniques, multiple protocols for cloud-enabled dialogue services, pluggable infrastructure for component reuse and service enhancement, as well as strong scripting functions in both XML and Python.

VI. Conclusion and Future Work

To develop a multiple domain spoken dialogue application, we must have the following functional modules or components in place: dialogue manager, domain spotter, various modules supporting respective domain knowledge sources, fundamental speech and dialogue components (speech recognition, natural language understanding, natural language generation, speech synthesis, etc.), some lexical, syntactical and semantical analysis modules, as well as strong scripting capability to link and glue all modules together. In addition, if a cloud-enabled dialogue agent is required, communication middleware which can "talk" to dialogue service and various service clients such as web browsers, mobile apps, etc. must be provided. The spoken dialogue framework presented in this paper provides a complete solution to meet all above requirements for the quick configuration of multiple domain dialogue agents with multiple knowledge sources. All the components in the dialogue framework can be configured and reused. The supported knowledge sources include Frame-based XML, AIML, SQL database, RDF through SPARQL query, Vector space model based indexing, rules & facts knowledge base through expert system engine, as well as web search. New dialogue agent development can make use of any of the supported knowledge sources or any combination of them.

In the future, we will continue to strengthen some of the supported knowledge source. For instance, RDF knowledge source, currently supports only few facts query on some specific open SPARQL endpoints, such as DBpedia. In the

meantime, other type of knowledge sources will be integrated into the framework through new plug-in development.

REFERENCES

[1] Emerson Cabrera Paraiso, Andreia Malucelli, "Ontologies supporting intelligent agent-based assistance," *Computing and Informatics,* Vol. 30, pp. 829–855, 2011.

[2] D. Tsovaltzi and A. Fiedler, "Construction and Use of a Mathematical Ontology in a Tutorial Dialogue System". *IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, Acapulco, Mexico, 2003

[3] Martin Beveridge, John Fox, "Automatic generation of spoken dialogue from medical plans and ontologies", *Journal of Biomedical Informatics* 39 (2006) 482–499

[4] Robin D. Burke, Kristian J. Hammond, Vladimir Kulyukin, Steven L. Lytinen, Noriko Tomuro, Scott Schoenberg, "Question Answering from Frequently Asked Question Files", *AI Magazine*, Vol 18 Number 2 (1997), 57-66.

[5] Poonam Gupta, Vishai Gupta, ''A Survey of Text Question Answering Techniques'', *International Journal of Computer Applications* (0975 – 8887), Volume 53– No.4, September 2012

[6] Alexander lark, Chris Fox, Shalom Lappin, ''The handbook of computational linguistics and natural language processing'', July 2010, *Wiley-Blackwell*.

[7] Ana-Maria Popescu, Oren Etzioni, and Henry Kautz, "Towards a theory of natural language interfaces to databases". In *Proceedings of IUI-2003*, pages 149–157, Miami, ACM, 2003.

[8] Sachin Kumar, Ashish Kumar, Dr. Pinaki Mitra, Girish Sundaram, "System and Methods for Converting Speech to SQL", proceedings of International Conference on Emerging Research", *Computing, Information, Communication and Applications, ERCICA 2013*, pp: 291-298

[9] Kendal, S.L., Creen, M., "An introduction to knowledge engineering", London: Springer, 2007, ISBN 978-1-84628-475-5, OCLC 70987401

[10] Annika Flycht-Eriksson. ''A survey of knowledge sources in dialogue systems''. In *Proceedings of IJCAI-99 Workshop on Knowledge and Reasoning in Practical Dialogue Systems,* August, 1999, Stockholm, pages 41-48.

[11] Robin D. Burke, Kristian J. Hammond, Vladimir Kulyukin, Steven L. Lytinen, Noriko Tomuro and Scott Schoenberg, "Question Answering from Frequently Asked Question Files", *AI Magazine*, Vol 18 Number 2, 57-66.

[12] Valentin Jijkoun and Maarten de Rijkej, "Retrieving Answers from Frequently Asked Questions Pages on the Web". *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*. ACM Press, 2005.

[13] Chatbots directory: https://www.chatbots.org/

[14] Ryuichi Nisimura, Takashi Uchida, Akinobu Lee, Hiroshi Saruwatari, Kiyohiro Shikano and Yoshio Matsumoto, "ASKA: Receptionist robot with speech dialogue system", *Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems*, EPFL, Lausanne, Switzerland, Oct. 2002

[15] Shiyan Ou, Constantin Orasan, Dalila Mekhaldi, Laura Hasler, "Automatic question pattern generation for ontology-based question answering", *Proceedings of the Twenty-First International FLAIRS Conference* (2008)

[16] Vanessa Lopez, Victoria Uren, Marta Sabou, Enrico Motta, "Cross Ontology Query Answering on the Semantic Web: An Initial Evaluation", *K-CAP'09*, September 1–4, 2009, Redondo Beach, California, USA.

[17] S. Bloehdorn, P. Cimiano, A. Duke, P. Haase, J. Heizmann, I. Thurlow, and J. Voelker, "Ontology-based question answering for digital libraries". In L. Kovcs, N. Fuhr, and C. Meghini, editors, *Proceedings of the European Conference on Research and Advanced Technologies for Digital Libraries* (ECDL), number 4675 in Lecture Notes in Computer Science, pages 14-25, Berlin, Germany, 2007. Springer.

[18] Ian O'Neil, Philip Hanna, Xingkun Liu, Des Greer, Michael McTear, "Implementing advanced spoken dialogue management in Java". *Speech Comm*unication. 54, 2005, 99–124

[19] Cheongjae Lee, Sangkeun Jung, Seokhwan Kim, Gary Geunbae Lee, "Example-based dialog modeling for practical multi-domain dialog system". *Speech Communication*, 51, 2009, 466–484

[20] Emerson Cabrera Paraiso, Andreia Malucelli, "Ontologies supporting intelligent agent-based assistance". *Computing and Informatics*, Vol. 30, 2011, 829–855

[21] CLIPS expert systems: http://clipsrules.sourceforge.net/

[22] R.E. Banchs, H.Li, "IRIS: a Chat-oriented dialogue system based on the vector space model", in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, 2012

[23] Seokhwan Kim and Rafael E. Banchs, "R-cube: a dialogue agent for restaurant recommendation and reservation", *Proceedings of Asia-Pacific Signal and Information Processing Association – Annual Summit and Conference (APSIPA)*, 2014

[24] Ridong Jiang, Yeow Kee Tan, Dilip Kumar Limbu, Tran Anh Tung and Haizhou Li, "A configurable dialogue platform for ASORO Robots", *Proceedings of Asia-Pacific Signal and Information Processing Association – Annual Summit and Conference (APSIPA ASC)*, Xi'an, China, 2011

[25] Andreea I. Niculescu, Kheng Hui Yeo, Luis F. D'Haro, Seokhwan Kim, Ridong Jiang and Rafael E. Banchs, "Design and Evaluation of a Conversational Agent for the Touristic Domain", *Proceedings of Asia-Pacific Signal and Information Processing Association – Annual Summit and Conference (APSIPA ASC)*, 2014

[26] Injae Lee, Seokhwan Kim, Kyungduk Kim, Donghyeon Lee, Junhwi Choi, Seonghan Ryu, and Gary Geunbae Lee, A Two-Step Approach for Efficient Domain Selection in Multi-Domain Dialog Systems, Proceedings of the 4th International Workshop on Spoken Dialog Systems Ermenonville, November 28-30, 2012

[27] Satoshi Ikeda, Kazunori Komatani, Tetsuya Ogata, Hiroshi G. Okuno, Extensibility Verification of Robust Domain Selection against Out-of-Grammar Utterances in Multi-Domain Spoken Dialogue System, Proceedings of Interspeech, September 22-26, Brisbane Australia, 2008

[28] Dan Bohus, Antoine Raux, Thomas K. Harris, Maxine Eskenazi, Alexander I. Rudnicky, Olympus: "an open-source framework for conversational spoken language interface research". *Proceedings of HLT-NAACL 2007 workshop on Bridging the Gap: Academic and Industrial Research in Dialog Technology* (2007).

[29] Levin, E., Narayanan, S., Pieraccini, R., Biatov, K., Bocchieri, E., Di Fabbrizio, G., Eckert, W., Lee, S., Pokrovsky, A., Rahim, M., Ruscitti, P., and Walker, M, "The AT&T-DARPA Communicator mixed-initiative spoken dialog system", *Proceedings of the Sixth International Conference on Spoken Language Processing (ICSLP'2000)*, Beijing, China, pp. 122-125, 2000.

[30] Staffan Larsson, David R Traum, "Information state and dialogue management in the TRINDI dialogue move engine toolkit", *Natural Language Engineering*, Volum 6, Issue 3-4, 323 – 340, 2000.

[31] J. Bos, E. Klein, O. Lemon, T. Oka, "DIPPER: Description and Formalisation of an Information-State Update Dialogue System Architecture", *Proceedings of the Fourth SIGdial Workshop of Discourse and Dialogue*, 2003.
[32] Peter Ljunglöf, "Trindikit.py: An open-source Python library for developing ISU-based dialogue systems". In *Proceedings IWSDS'09, 1st International Workshop on Spoken Dialogue Systems Technology Workshop*, Kloster Irsee, Germany, 2009.
[33] Pandorabots: http://www.pandorabots.com/
[34] Nobuo Kawaguchi, Shigeki Matsubara, Katsuhiko Toyama, Yasuyoshi Inagaki, "An Architecture for Multi-Domain Spoken Dialog Systems", *Proceedings of the 5th Natural Language Processing: Pacific Rim Symposium (NLPRS'99),* pp.463-466, 1999.
[35] Kazunori Komatani, Naoyuki Kanda, Mikio Nakano, Kazuhiro Nakadai, Hiroshi Tsujino, Tetsuya Ogata, Hiroshi G. Okuno, "Multi-Domain Spoken Dialogue System with Extensibility and Robustness against Speech Recognition Errors", *Proceedings of the 7th SIGdial Workshop on Discourse and Dialogue*, pages 9–17, Sydney, July 2006.
[36] Ian O'Neill, Philip Hanna, Xingkun Liu and Michael McTear, "Cross domain dialogue modelling: An object-based approach", *Proceedings of ICSLP*, 2004, pp. 205–208.