Knowledge Acquisition Module for Conversational Agents

Pauline Mak[1], Byeong-Ho Kang[1], Claude Sammut[2], Waleed Kadous[2]

University of Tasmania, Tasmania, Australia,
{Pauline.Mak,Byeong.Kang}@utas.edu.au
[2] University of New South Wales, Sydney, Australia,
{claude,waleed}@cse.unsw.edu.aubhkang@utas.edu.au

Abstract. The focus of traditional conversational agents is placed on natural language processing and understanding the needs of the user. These agents are typically implemented for specific domains such that domain knowledge and conversations are built manually. Domain knowledge of these agents are encoded as conversational content which causes a poor separation between the two distinct types of knowledge. This problem is magnified by the lack of knowledge acquisition tools and, as a result, agents find it difficult to adapt to different domains and to update existing knowledge bases.
The framework proposed in this paper aims to rectify this problem by building a module to handle knowledge acquisition. The module acquires knowledge through a case-based methodology called Ripple Down Rules (RDR); a technique that has been employed successfully across a host of expert system.

1 Background

Intelligent conversational agents have been the ultimate goal in the realm of human computer interactions. This has motivated a large number of agents of varying complexity to surface in recent years [7]. One of the earliest, and perhaps most well known, example of such agents is Eliza [12]. Eliza engages a conversation through simple scripts that prompt the speaker with questions derived from the latest utterance. While this approach allows one to converse with Eliza, the resulting dialogue becomes a series of repetitious questions. For an agent to be of any use, it should be able to provide some kind of service, such as giving an expert advice in a field of specialisation. This is the type of agent that this paper will focus on and in particular, the customisation of such agents.

Contemporary conversational agents look to the fields of linguistics and memory for more realistic models of agents [4]. From this, many different types of parser and speech generation systems are built. For instance, the Phillips train system [1] is a conversational agent that is connected to a public telephone network which can provide train timetabling information through conversation. This agent essentially treats the user input as a query into the timetable database. A similar system by Zelle and Mooney [13] can learn to translate sentences into

queries for obtaining geographical information about the United States. Both of these system have a predefined database that stores the knowledge of the agent. While this is good for systems where most facts are atomic, this is not suitable for storing expert knowledge. This problem solving knowledge is based on heuristics which cannot be suitably expressed in a flat database.

There are even more ambitious agents that do not only incorporate speech, but also body language, such as emotions and spatial awareness, as exemplified by Rea [3] and Max [6]. These agents are able to convey subtleties such as facial expressions which are important non-verbal cues for expressing the context in which spoken information is given.

Recent developments in the Cyc project allows expert to update the knowledge base through a web-based application [2]. The system has tools such as dictionaries and analogies to introduce new facts into the knowledge base. The system is also sufficiently intelligent to generate related facts which are implied by the user input. While this system provides one of the most innovative method of knowledge acquisition, it only applies to adding facts in a common-sense knowledge base. While this can provide a solid back-end for general knowledge, the architecture is not suitable for retaining specific domain knowledge.

In summary, contemporary conversational agents are focused on the problem of providing a natural and adaptive way to interact with the end user. Little has been done on simplifying the process of updating the knowledge of the agents. This paper proposes a knowledge acquisition module that will allow conversational agents to learn domain specific knowledge, thus, creating a flexible framework for customising specialised agents. The subsequent sections will demonstrate how this structure works.

2 Framework

The module builds on an existing conversational agent and knowledge acquisition methodology. A special data structure known as frame is fundamental to the development of the module. The following sections shall introduce these elements briefly.

2.1 Conversational Agent

The module for knowledge acquisition is built on existing conversational agent technologies. First and foremost, the conversational agent that is augmented with the module is Probot. It is an extension to the programming language iProlog, an enhanced version of standard Prolog [10].

Probot is a turn-based conversational agent, where the user and the agent alternate speaking. There is no natural language processing in Probot, instead user input is matched against patterns defined in files known as scripts. Context of these patterns are separated through the use of topics; this allows the same patterns with varying meanings to coexist within the agent without ambiguity [11]. It is possible to define multiple topics for the agent, however, the agent can only handle one topic at a time.

2.2 Frames

Frames are structures (first proposed by Marvin Minsky [8]) used for describing a typical situation, such as entering a room. Each aspect of the situation is represented by a slot. Minsky believes that one understands dialogue through finding frames that describe the content most appropriately. He also believes that there is a hierarchy of frames, where each subsequent level in the chain of inheritance contains increasing amount of details about the topic of interest, much like subclasses in the object oriented paradigm. A frame is also able to maintain a list of questions that can be asked when certain values are needed.

There are two types of knowledge that operate in a conversational agent. One is conversation knowledge and the other, domain knowledge. These types of knowledge are modular, and use a single data structure known as a frames though these are maintained through separate procedures.

Frames can also react to events through the use of daemons. These are similar to event handlers in languages such as Java. iProlog has defined a number of daemons that are called when a variety of events, such as the instantiation of a frame, occurs.

Frames are ideal data structures for the knowledge acquisition module as it supports flexible data storage as well as containing elements of conversation in a modular manner.

2.3 Knowledge Acquisition

The knowledge acquisition methodology adopted for this project is the Ripple Down Rule (RDR) approach. It was first devised by Paul Compton [5] as a consistent approach to updating large knowledge bases [9].

The core assumption of this methodology is that experts make judgments on a case by some implicit reasoning. Different conclusions are given as a result of variations in these cases. Experts are better at identifying the variations in cases that distinguishes one from another.

Knowledge is stored within a rule tree, where each node is a conclusion as well as a set of conditions that it must satisfy. As a new case is given to the knowledge base, it will be tested against rules within the tree. Conclusion is given by the last node that the case completely satisfies. If the expert disagrees with the conclusion, new rules can be written to correct the knowledge base. New rules branches from the last correct node, that is, the node that had the incorrect conclusion. As the rest of the tree is not affect by this change, RDR gives a consistent approach to incremental learning.

Updates are only required when the expert disagrees with the conclusion the knowledge base has drawn. In practice, each update requires the specification of a case, a set of rules that identifies the case and a new conclusion that is different from the conclusion generated by the knowledge base. These elements will form parts of the conversation that the agent must undertake for extracting knowledge from the expert.

3 Module

The module is consisted of two knowledge acquisition components; domain and conversational knowledge. Both uses the frame hierarchy to store information required by the knowledge acquisition process and the details of which will be examined in detailed in the following sections.

3.1 Knowledge Acquisition Frame Hierarchy

The knowledge acquisition frame (KA frame) is used for storing information required for a single update. There are a number of fixed elements that a KA frame as dictated by RDR and is discussed previously in Section 2.3. KA frame should contain three slots, where each item is represented by a frame and are essentially subframes. As a result, a KA frame will have the following subframes: case frame, conclusion frame and condition frame, as illustrated in Figure 1. Shaded boxes indicates that the frame is domain dependent.

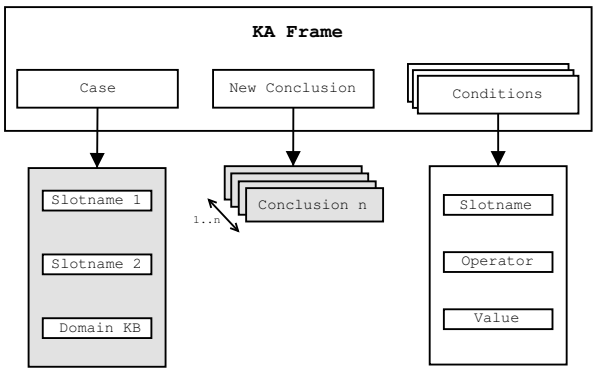

Fig. 1. KA Frame

The case frame describes features of a situation from which an expert can draw a conclusion, such as symptoms of a patient. This frame is domain dependent, and must be remodeled for every knowledge base the agent is introduced to. This task is the responsibility of the developer that is adapting the agent to the new domain, of course, with sufficient consultation with the experts.

The case frame also houses the domain knowledge base. This is a logical choice as the knowledge base must have access to the features of a case in order to draw conclusions from it. Due to daemon inheritance, the knowledge base will only need to be defined in the generic case frame and all instance frames will have access to it. Therefore, any new cases created will also use the most recent version of the knowledge base. This inheritance structure complements RDR;

consistency allows all existing and future cases to share the same knowledge base without the need to restructure.

Conclusions are possible classifications that an expert may give to a case. This set of values can be stored as flat values, however in some cases, it may be represented more adequately by frames. For example, the agent may learn how to select cars for a given customer. Conclusions in this case would be better represented by a car frame rather than just the model of the car. This frame is domain and implementation independent, and must be recreated for each new knowledge base.

Conditions are responsible for holding rules related information where rules specify distinguishing properties of a case. These properties are predicates that will either evaluate to true or false. Due to the frame structure, these predicates are required to know which slot in the case frame is to be tested. Therefore, a condition frame needs to have three slots: slot name, operator and comparison value. In the simplest case, a slot will be verify whether it holds a specific value. However, there are other data types that requires more than just equality, such as numeric and date comparisons. It is possible to create new definitions of comparison frames if these new frames contain the three slot (slot name, operator and comparison value.)

It is possible for an expert to define multiple rules for the same case. This is made possible by allowing the KA frame to take a list of condition frames. When added to the knowledge base, the list will be coalesced with the boolean function AND.

Once all the data is collected, knowledge can be updated in the knowledge base by calling a predefined function *add_rdr*. This will add the new rule at the appropriate location in the RDR tree.

This hierarchy covers the basics for knowledge acquisition, however, consideration should be given to understanding expert terminology for specifying these rules and cases. The next section shall explain in detail how these terminologies can be obtained by the agent and how these two hierarchies cooperate.

3.2 Conversation Acquisition Frame Hierarchy

The conversation acquisition frame hierarchy allows an agent to translate what an expert has said into an internal representation. Previously in Probot, users had to create scripts for recognising patterns in speech. This was a simple yet functional approach. Unfortunately, each slot is interpreted differently and requires a new set of patterns. As frames are populated with more slots, scripts have to be written for each new slot; resulting in a proliferation of scripts. The hierarchy proposed here is a simple solution for solving this problem by merging scripts and internal data representations into the frame structure.

Knowledge acquisition in this module uses RDR for storing patterns that an agent may need to recognise and it is similar to the hierarchy described in the previous section. However, this is restricted to conversational knowledge which gives greater control over how the knowledge is stored and manipulated and as a result more fixed elements can be defined. As specified by the RDR paradigm,

the conversational knowledge acquisition hierarchy should contain a case, rule and conclusion. Any utterance from the user is treated as a case. Rules are patterns that can be matched over the case. Finally conclusions are translated values for slots in the frame.

Any slot that requires user speech input needs to have the infrastructure for accessing utterances from the user as well as a location to store rules that are applied to the input. An additional slot is added to the frame structure, *utterance*, which stores user input and is updated with every utterance. Presently, the *utterance* slots for every frame throughout the hierarchy will have the same sentence to interpret. However, it is possible to include some text manipulation as the input is interpreted by the framework and will be an area for future exploration. The knowledge base is stored in another fixed slot, *interpret* in which, the rules for interpretation are stored here.

The infrastructure alone is not enough for specifying the rules; there needs to be a way to qualify the patterns. A new function, *u_match*, is defined in iProlog that will allow the following type of patterns to be matched, based on a list of given words, *a*, to match against:

- *all*: the utterance must have every word in the list *a*.
- *any*: the utterance must contain at least one word that is in the list *a*.
- *has_phrase*: the utterance must contain the phrase *a*.

This is not intended to be a comprehensive list of operation that can be performed on speech text, though they are acceptable for most circumstances, particularly in keyword matching. Further operations can be added to this function by adding new definitions to *u_match* in iProlog.

The acquisition of these patterns is done manually, by which, the user has to explicitly initiate the process. However, logically it should also be possible for the process to be instantiated when the system finds that none of its frames can interpret the user input.

Part of the vision of creating this module is to allow frames to be reusable. As each frame is responsible for interpreting data for itself, it is possible to reuse the same conversational knowledge base. For example, gender may be a subframe of another frame, such as person. The agent then learns some rules for defining the gender of a person. The agent may need to be adapted to other domains at some point later, which requires information on interpreting the gender, say, for animals. There are commonalities in the language used between the domains and it is here that modularity of the proposed hierarchy is key. Rules learned from the previous domain can be applied to the new domain. Of course, new rules have to be created for domain specific terminologies, in addition to incorrect conclusions as frames are used in different contexts. RDR excels in incremental knowledge acquisition, which reinforces its necessity in this module.

4 Dialogue Control

The goal of the dialogue control component is to make expert-agent conversations natural. Most customer inquiry phone lines require users to say or enter specific

answers when promoted with a fixed set of questions, which leads to a menu driven dialogue rather than a conversation.

It is important to first examine what kind of interaction the agent must engage in for knowledge acquisition. It is essentially a task for filling in data, which, when completed, the agent is able to update the knowledge base with. As described in previous sections, the agent needs a fixed number of elements. However, it is not necessary for the users to give all of the information in a fixed order. Probot uses scripts to control the direction and the content of a conversation. This is not very practical if flexibility is a requirement, as it is not possible to anticipate all combinations of which order a user may specify facts. As a result, previous implementations of the knowledge acquisition module follows the menu driven dialogue model.

A more adaptive approach is to allow users to specify slot values in no fixed order. It is imperative for the old script system to change. As the conversation is centered around extracting information from the user to fill in the frame, a frame is now bounded to a script. The purpose of a script is to direct the overall conversation, such as, how the agent should react once all the necessary information is given, whereas frames now handle the translation of dialogue to some internal knowledge representation. The top-most frame of the hierarchy will be associated with the script. Since some information may be stored within subframes, it is necessary to pass information down the structure such that subframes can also interpret the user input for values. In order to do so, the frame must know which slots are subframes. This leads to the inclusion of an additional slot called *subframes* where the slot name and the subframe type is paired. In the future, slots that uses frame will automatically tagged as a subframe and will eliminate the need for the extra slot. Scripts can then use this information to create subtopics. Subtopics link a subframe to the top-most frame, and thus, provide a channel of communication for passing user input. Every input is filtered through the frame hierarchy. As each frame is able to process user input, slots can be filled in any order.

Scripts must be able to verify that an entire frame has been filled successfully. There may be elements in frames that are optional and it is therefore necessary to specify which slot is a requisite. A new daemon, *required*, is introduced which uses any iProlog predicate to check whether a value has fulfilled the requirements. This works in conjunction with a new function, *has_req*, which applies the predicate defined in *required* to determine whether a frame is complete. When applied to subframes, this will allow the entire hierarchy to be checked. In terms of knowledge acquisition, this will guarantee that the user has provided enough information for the knowledge base to be updated. This forms the last element needed for flexible conversation.

5 Conclusion

Current conversational agents are primarily focused on realistic simulation of human interactions, such as the use of body language and behaviour models.

However, development into agent customisation has been negligible. This paper has demonstrated a framework that is a structured method for agents to learn domain and conversational knowledge.

It is made possible through the separation of conversational and domain knowledge to provide a consistent approach for customising an agent. Complementing this architecture is the use of RDR. The methodology supports incremental learning in a consistent manner. This is significant as the system is aimed to be adaptive and can gain knowledge while in use.

A crucial part of the module is the use of frames. This structure encapsulates data in a manner that allows knowledge to be reusable. The cost to adapt an agent to a similar domain will be reduced. Frames also contribute to flexible conversation flow, which would otherwise become a menu driven dialogue.

In summary, the module detailed in this paper is a novel approach to solving the problem of customising conversational agents. It is capable of learning conversational and domain knowledge in a flexible and reusable manner.

6 Future Work

This framework has all the elements required for knowledge acquisition for both domain and conversational knowledge, though, the work thus far is of a preliminary nature.

Gauging the success of a conversational agent is a difficult manner, as there are no standard metrics for measuring the performance. As RDR has already proven to be a successful methodology for knowledge acquisition, there is no need to test whether the module can acquire knowledge. Instead, further work needs to be done on testing how the entire framework cooperates when adapting to a specific domain. Aside from testing on one domain, there is also a need to adapt the agent to multiple domains - this way, the re-usability of the framework can be revealed.

Acknowledgments

This research has been funded by the Smart Internet Technology CRC and the authors would like to express thanks for the support received throughout the production of this paper. Also, thanks to Adam Berry for proof reading this paper.

References

1. H. Aust, M. Oerder, F. Seide, and V. Steinbiss. The philips automatic train timetable information system. In *Speech Communication*, volume 17, pages 249–262. 1995.
2. David Baxter, Jon Curtis, Dave Schneider, Robert Kahlert, Pierluigi Miraglia, Peter Wagner, Kathy Panton, Gavin Matthews, and Amanda Vizedom. An interactive dialogue system for knowledge acquisition in cyc. In *IJCAI-03 Workshop on Mixed Initiative Intelligent Systems*, pages 138–145, 2003.

3. J. Cassell, T. Bickmkore, M. Billinghurst, L. Campbell, K. Chang, H. Vilhjalmsson, and H. Yan. Embodiment in conversational interfaces: Rea. In *CHI99*, pages 520–527, 1999.
4. J. Cassell and M. Stone. Living hand to mouth: Psychological theories about speech and gesture in interactive dialogue systems. In *Proceedings of the AAAI 1999 Fall Symposium on Psychological Models of Communication in Collaborative Systems*, pages 34–42, 1999.
5. P. Compton and R. Jansen. A philosophical basis for knowledge acquisition. In *3rd European Knowledge Acquisition for Knowledge-Based Systems Workshop*, pages 75–98, 1989.
6. S. Kopp, B. Jung, N. Lessmann, and I. Wachsmuth. Max - a multimodal assistant in virtual reality construction. *KI-Kunlnstliche Intelligenz*, 4:11–17, 2003.
7. M. McTear. Spoken dialogue technology: Enabling the conversational user interface. *ACM Computing Surveys*, 34(1):90–169, March 2002.
8. M. Minsky. *The Psychology of Computer Vision*, chapter A Framework for Representing Knowledge. MIT Press, 1981.
9. P. Preston, G. Edwards, and P. Compton. A 2000 rule expert system without a knowledge engineer. In *Proceedings of the 8th AAAI-Sponsored Baff Knowledge Acquisition for Knowledge-Baed Systems Workshop*, 1994.
10. C. Sammut. iprolog programmer's manual. http://www.cse.unsw.edu.au/ claude/teaching/AI/notes/prolog/index.html.
11. C. Sammut. *Electronic Transactions on Artificial Intelligence*, chapter Managing Context in a Conversational Agent. Linkoping University Electronic Press, 2001.
12. J. Weizenbaum. Eliza - a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(36-45), 1966.
13. J. M. Zelle and R. J. Mooney. Learning to parse database queries using inductive logic programming. In *Proceedings of the 14th National Conference on Artificial Intelligence*, pages 1050–1055. AAAI Press/MIT Press, 1996.