Metacognitive Guidance in a Dialog Agent

Elizabeth McNany*, Darsana Josyula*†, Michael Cox*, Matthew Paisner* and Don Perlis*
† Bowie State University, Bowie, MD, USA
* University of Maryland, College Park, MD, USA
{beth, darsana, mcox, mpaisner, perlis}@cs.umd.edu

***Abstract*—The paper discusses the benefits of metacognitive guidance for a natural language dialog agent. These capabilities may be included directly in the agent or through a general purpose external module. We report on the specific case of handling pause time in dialog, using a metacognitive loop within the agent, and discuss future experiments implementing guidance for this example also using the general module.**

Keywords-metacognition; dialog management

I. INTRODUCTION

In human dialog, if the listener does not understand the speaker, the listener will typically notice the problem and take some action to address it. He may ask the speaker to repeat the statement, ask for clarification, or ignore the anomaly in hopes of determining the meaning later from context. In this way normal flow of conversation may be maintained. Similarly, if the speaker does not respond, several scenarios may have occurred: the speaker has left the conversation; the speaker forgot; or the speaker is thinking, and does not yet have a response. The listener then has to choose an appropriate course of action. This may include reminding the speaker by repeating the question, waiting a bit longer for an answer, or deciding the speaker is no longer responsive and ending the conversation.

When faced with such anomalies in dialog, humans very effectively *note* the anomaly, *assess* how to deal with it, and *guide* a response strategy into place. We call this the N-A-G process [2], and have modeled it in our artificial dialog agent, Alfred.

N-A-G is an example of so-called *metacognition*. Normal cognition entails reasoning with knowledge available to the agent; meta-cognition is reasoning about this reasoning. For instance, in the previous example, the speaker reasons that after asking a question the listener will respond; after this fails, the speaker must then decide (via logical reasoning, presumably) where his or her logic failed and thus how to recover from the anomaly. N-A-G is an essential component of a larger reasoning process, which we call the metacognitive loop (MCL) [13], [3]. To employ MCL, a system must (i) have expectations about what it will observe; (ii) note deviations from those expectations, reason about, and respond to them; and (iii) adjust those expectations as suggested by experience; item (ii) is simply the N-A-G process.

We have implemented several applications with MCL built in; and we more recently implemented a general purpose MCL module, which we call GP-MCL. GP-MCL is designed to be attached to many types of systems and includes facilities to handle many types of anomalies - from physical or external errors such as unexpected obstacles or user input, to logical or internal errors based on incorrect assumptions. Because it is created with no specific system in mind, GP-MCL is flexible enough to deal with failures in a variety of possible situations.

We are exploring the possibility of applying the general guidance provided by MCL to the particular case of the dialog agent Alfred. The specific application we examine is that of an expected pause time between utterances in a conversation. We also propose future experiments in a 2x2 design, of location of anomaly versus location of anomaly handling, detailed later in Section VI.

II. RELATED WORK

Work in linguistics focusing on conversational competence has shown the importance of meta-reasoning and error handling in natural dialog. Based on Chomsky's [6] notion of linguistic competence, Hymes introduced the idea of communicative competence [8]. Hymes identified four characteristics of communicative competence, namely if the utterance is: formally possible (that is, grammatical), feasible with available means, context-appropriate, and actually performed. Canale [5] expanded on this idea, also with four types of communicative competence: grammatical, sociolinguistic, discourse (cohesion of utterances within a dialog), and strategic (strategies used when communication breaks down). Of these four, sociolinguistic and strategic were identified by Savignon [12] as necessary and sufficient conditions for communicative competence. Some examples of strategies given in Tarone [14] include approximation, circumlocution, repetition, asking for help, and abandoning the utterance.

Even prior to these findings in linguistics, Rieger [11] concluded that sufficient meta-reasoning when handling natural language can overcome deficiences in other language skills. McRoy [10] echoed this, including the ability to deal with mistakes as a central component of reasoning and linguistic capabilities. These mistakes were defined as inconsistencies between actual utterances and expectations of the dialog

Copyright (c) IARIA, 2013. ISBN: 978-1-61208-273-8

participant in [7]. More recently, Anderson and Lee [1] found that more than half of dialog management is meta-linguistic, dealing with mistakes and references to previous conversation. Clearly, the ability to handle anomalies in conversation is a crucial part of natural dialog management.

III. ALFRED

Alfred is a dialog agent which acts as an interface between a human user and a task-oriented domain [9]. It accepts English sentences as input and parses them into appropriate commands, based on the particular domain and information in its knowledge base (KB). Alfred is designed to be a general agent and flexible enough to handle a variety of different scenarios. For each domain, Alfred has a dictionary listing the possible commands and objects, as well as specifying the command syntax for that domain.

To implement the N-A-G cycle, Alfred maintains a set of expectations regarding time, content and feedback; i.e, *when* a certain predicate is expected, the expected *values of parameters* in a predicate, and expected *predicate*. The expectations are represented as $expectation(\alpha, t1, t2)$, where $t1$ and $t2$ are values for time steps, and α is a predicate. That is, α is expected between time steps $t1$ and $t2$, or if $t2$ is zero, sometime after $t1$. If $t1$ is zero, α is expected to be true from the current time step, Now, until $t2$. If both times are zero, α is expected to always be true. $alpha$ is of the form $predicate(param\_list)$ where $param\_list$ can contain values, variables and predicates. Any parameter in the $param\_list$ can be set to _ to indicate that the specific parameter can take any value, that is, the value does not matter. If a parameter is set to a specific value then that value is expected for that parameter. On the other hand, if the parameter is set to a variable, then that variable is assumed to be universally quantified.

One specific expectation in Alfred is that of a pause time. During a typical dialog, Alfred will respond to the user, asking questions or informing them of a completed task. The user will then reply to carry the conversation forward. Thus, when Alfred speaks to the user, it has the expectation of a response within a reasonable time frame. If the user does not respond within this time frame, Alfred notes an expectation violation and may then take steps to respond to this violation.

In Alfred, the expected pause time is represented as $expectation(pause\_less\_than(100), 0, 0)$. An expectation of the form $expectation(pause\_less\_than(100), 0, 0)$ translates to expecting the predicate $pause\_less\_than(100)$ to be universally true across all time steps. As a result, Alfred asserts $pause\_less\_than(100)$ in its KB. However, when the current pause exceeds 100, $not(pause\_less\_than(100)$ gets asserted. As a result, a contradiction gets asserted in the Alfred knowledge base, and the corresponding formulas get distrusted. The contradiction handler built in to Alfred will then deal with the particular contradiction based on the type of violation that has occurred.

When the expectation is not met, Alfred interprets it as an indication of a failure: *noting* the problem, *assessing* the situation, and *guiding* a response strategy into place. In the example of a too-long pause, Alfred may say "Please tell me what to do now." When the expectation is met, the corresponding violation is removed from the knowledge base; when it is not, Alfred will attempt another response strategy until the issue is resolved.

IV. GP-MCL

GP-MCL has been used in a variety of other applications [3], [4], which indicates to us that it can be useful as a general component of any host system. As an addition to an existing system, it may monitor the host's expectations and attempt to resolve anomalies —but as a general component, it must handle many types of expectation failures and possible responses. We have developed a set of abstracted ontologies that aim to cover all potential categories of failures and responses [4]. They correspond to the steps of the N-A-G strategy: an ontology of *indications* for noticing expectation failures, an ontology of *failures* to categorize and assess the causes, and an ontology of *responses* to choose an appropriate strategy to guide the system in recovering.

A. Ontologies

The indications ontology corresponds to the Note phase, and includes definitions of nodes to represent various types of sensor and expectation failures (reading did not change, reading is out of range, etc.) and indications from the host (counter increased, current state, etc.). This layer is the entry point for information from the host into GP-MCL's reasoning system.

The second layer is the failures ontology, which roughly corresponds to the Assess phase. These nodes connect to the indications layer and represent the various types of errors possible in the host AI. The classes include categories for physical error, sensor noise or misconfiguration, and knowledge errors.

The responses ontology is the final layer, with connections from nodes from the previous. These nodes determine which course of action to recommend to the host, analogous to the Guide step of the N-A-G cycle. The classes are related to the action required, and whether it pertains to a physical error or knowledge error. Responses include recommendations to run a diagnostic, reset a sensor, rebuild models, fix the knowledge base, try again, or even ask for help.

B. Architecture

Given the general ontologies and nodes described, systems of arbitrary complexity may be constructed to handle errors in the host. Links between nodes are specified and given default weights, forming a chain of reasoning from the initial

Copyright (c) IARIA, 2013. ISBN: 978-1-61208-273-8

failed expectation to a possible solution. The ontologies and their linkages form a directed graph which can be viewed as a Bayes net. Conditional probability tables associated with each node allow computation of probabilities for responses.

The host may send information to a GP-MCL server over a TCP/IP interface using a socket interface. After initialization of the nodes, sensors, and expectations for the host, the host is responsible for sending periodic updates of the salient values. When updates are received, GP-MCL will respond with suggestions, if any, for the host to implement or ignore. Finally, the host replies to GP-MCL indicating whether it implemented the suggestion and if the action was successful.

GP-MCL stores certain information about the state of the host, including sensor states and type of expectation failure, as well as meta-information like previous ontology states and number of failed or successful repairs. This allows GP-MCL to update the probability tables such that it learns which responses are better for particular types of expectation failures and does not repeatedly suggest an action that fails.

V. Ongoing Application

In the initial design of Alfred, with minimal implementation of MCL, a too-long pause time was handled by asking the user, "Please tell me what to do now." This continues periodically each time the pause limit was exceeded, until the user responded or exited the program. However, if the user is always slow to respond or has left the conversation, asking again and again after the same pause time is not productive. Alfred must notice that it is repeatedly initiating the same response for an expectation violation (actual pause exceeded expected pause time) without making progress in the conversation, and hence consider an alternate response to the violation.

To achieve this, Alfred may keep track of the success of its repeated questioning. If the response is successful, the violation is removed from its KB. Otherwise, the original violation remains in the KB, and so when the same violation occurs again Alfred may evaluate candidate responses and choose an alternative. For example, the framework previously described may be extended to change the typical pause time dynamically when the user changes or responds at a slower pace. For instance, if the typical pause duration is t, a pause violation occurred at time $t1$, and removal of the violation occurred at $t1 + t + 5$, then Alfred's metacognitive reasoning can retract its expectation regarding a pause duration of t and assert an expectation of pause time $t + 5$.

An example in a different experimental set up involves using Alfred as an interface to direct trains. In this setting, we have implemented the monitoring of the success of initiated responses and evaluation of candidate options before immediately initiating the same response again. For instance, if a user requests "send the Chicago train to New York", Alfred may choose Metroliner as the candidate, a train that is currently in Chicago. However, if the user replies "No" and repeats the same request, Alfred evaluates its options, notices that its previous first choice, Metroliner, was an unsuccessful response, and instead chooses Northstar, a train that starts at Chicago. In this way Alfred is able to "learn" which entity is meant by "the Chicago train", instead of repeatedly choosing the same, incorrect train as a response to the user's request.

VI. Planned Experiments

Alfred's current set of expectations are based on its view of the current world. If the external world changes or Alfred's view of the external world changes, Alfred has to update its expectations. The mechanism for revising existing expectations may be implemented in either Alfred alone or with the attachment of GP-MCL to Alfred. This mechanism has two components: (i) noticing the need for revising expectations, and (ii) updating the expectations to match reality. The first component is implemented by keeping track of expectation violations, responses initiated to deal with the violations, and success of the responses initiated to deal with each violation. The second component is implemented by assessing how far Alfred's expectations deviated from the actual observations and making adjustments accordingly.

Expectation violations can be categorized as *internal errors* —ones due to internal model error, or *external errors* —ones due to changes in the external world. Internal errors refer to incorrect assumptions in Alfred's KB, which may be corrected by adjusting said expectations; external errors are anomalies that the agent cannot control, such as the user leaving the conversation. Although both types must be accounted for in a complete implementation, we have chosen to address these issues independent of each other using two different techniques: (i) through a metacognitive handler built in to Alfred, and (ii) via an external method such as GP-MCL.

If expectation violations are handled by internal MCL, then it is easy for the internal MCL to access any part of Alfred's KB, and hence, making changes to any part of Alfred's KB becomes easier. However, since the internal MCL would share the same resources as the underlying deliberative component, the metacognitive processing could potentially slow down the normal deliberative processing of Alfred. On the other hand, utilizing an external module like GP-MCL for handling expectation violations may involve additional overhead costs of connecting it to the host system, but may result in a more flexible system overall. In future work, we will examine this tradeoff between including MCL directly in Alfred versus externally through the GP-MCL framework to handle one or both types of anomalies.

With these variables we have a 2x2 experimental design, shown in Table I. Each number 1 - 4 in the table represents a potential experiment, combining techniques for handling the two categories of errors. Hence, set-up 1 handles both

Copyright (c) IARIA, 2013. ISBN: 978-1-61208-273-8

TABLE I. FUTURE EXPERIMENTAL SETUPS.

	Expectation Error	User Error
Internal Handler	1, 2	1, 4
External Handler	3, 4	3, 2

internal and external errors by a metacognitive process built internally within Alfred, while set-up 3 handles both externally; setups 2 and 4 are combinations of the two. In set-up 2, internal errors are handled by the internal MCL, while external errors are handled by GP-MCL, and in set-up 4, internal errors are handled by GP-MCL and external errors are handled by the internal MCL. The ongoing application described previously thus corresponds to set-up 1 in Table I: implementing MCL within Alfred to handle both internal and external errors.

Many particular capacities can be tested with the above experimental setups, from pause time to word disambiguation, new words and/or meanings, etc. We are currently nearly finished with a Wizard-of-Oz pilot study of pause time in set-up 3: external MCL with both expectation- and user-error handling. We anticipate a complete implemented study of this and much more in the near future.

VII. CONCLUSION

Metacognition is a neccessity for natural dialog, and different strategies may be used to implement that capability. In the specific case of pauses in dialog, there is an expectation that the user will respond within a specified time. A failed expectation may be caused by an external (user) or internal (knowledge) error; both must be accounted for. However, these two types of anomalies may be handled either with logic internal to Alfred itself, or by the external module GP-MCL. Implementing MCL within Alfred requires keeping track of previous actions, to detect repeated failures and thus know when to adopt a new strategy. Future work will explore different methods of including MCL within Alfred's dialog system and the benefits and tradeoffs involved.

ACKNOWLEDGMENT

This material is based upon work supported by ONR Grant # N00014-12-1-0430 and ARO Grant # W911NF-12-1-0471.

REFERENCES

[1] M. Anderson and B. Lee, "Metalanguage for dialog management," in 16th Annual Winter Conference on Discourse, Text and Cognition, 2005.

[2] M. Anderson and D. Perlis, "Logic, self-awareness and self-improvement: The metacognitive loop and the problem of brittleness," Journal of Logic and Computation, vol. 15, no. 1, 2005, pp. 21-40.

[3] M. Anderson, T. Oates, W. Chong, and D. Perlis, "The metacognitive loop I: Enhancing reinforcement learning with metacognitive monitoring and control for improved perturbation tolerance," Journal of Experimental and Theoretical Artificial Intelligence, vol. 18, no. 3, 2006, pp. 387-411.

[4] M. Anderson et al., "Ontologies for Reasoning about Failures in AI Systems," in Proceedings, Workshop on Metareasoning in Agent-Based Systems at the Sixth International Joint Conference on Autonomous Agents and Multiagent Systems, 2007.

[5] M. Canale, "From communicative competence to communicative language pedagogy," in Language and Communication, J. Richards and R. Schmidt, Ed., New York: Longman, 1983, pp. 2-27.

[6] N. Chomsky, *Aspects of the theory of syntax*. Cambridge, MA: MIT Press, 1965.

[7] G. Hirst and S. McRoy, "The repair of speech act misunderstandings by abductive inference," in Computational Linguistics vol. 21 no. 4, 1995, pp. 435-478.

[8] D. Hymes, "On Communicative Competence," in Sociolinguistics: Selected Readings, J. B. Pride and J. Holmes, Ed., Harmondsworth: Penguin Books, 1972, pp. 269-293.

[9] D. Josyula, "A Unified Theory of Acting and Agency for a Universal Interfacing Agent," Ph.D. dissertation, University of Maryland, College Park, 2005.

[10] S. McRoy, "Abductive Interpretation and Reinterpretation of Natural Language Utterances," Ph.D. dissertation, University of Toronto, 1993.

[11] C. Rieger, "Conceptual Memory: A Theory and Computer Program for Processing the Meaning Content of Natural-Language Utterances," Ph.D. dissertation, Stanford University, 1974.

[12] S. Savignon, Communicative Competence: Theory and Classroom Practice, Reading, MA: Addison-Wesley Publishing Co., 1983.

[13] M. Schmill, et al., "The Metacognitive Loop and Reasoning about Anomalies," in Metareasoning: Thinking about Thinking, M. Cox and A. Raja, Ed., Cambridge, MA: MIT Press, 2011, pp. 183-198.

[14] E. Tarone, "Some thoughts on the notion of communication strategy," in TESOL Quarterly vol. 15 no.3, 1981, pp. 285-295.

Copyright (c) IARIA, 2013. ISBN: 978-1-61208-273-8