MAi: An Intelligent Model Acquisition Interface for Interactive Specification of Dialogue Agents

Tathagata Chakraborti, Christian Muise, Shubham Agarwal, Luis A. Lastras
IBM Research AI, USA
{tathagata.chakraborti1, christian.muise, shubham.agarwal}@ibm.com, lastrasl@us.ibm.com

Abstract

The state of the art in automated conversational agents for enterprise (e.g. for customer support) require a lengthy design process with experts in the loop who have to figure out and specify complex conversation patterns. This demonstration looks at a prototype interface that aims to bring down the expertise required to design such agents as well as the time taken to do so. Specifically, we will focus on how a meta-writer can assist the domain-writer during the design process and how complex conversation patterns can be derived from simplifying abstractions at the interface level.

Sign up! An overview of the project can be viewed at: ibm.biz/mai-video. If you would like to take `MAi` for a ride, please sign up here: ibm.biz/mai-signup.

Designing Dialogue Agents: State-of-the-Art

Current enterprise-level automated goal oriented dialogue agents require significant expertise, time and effort to build (Sreedhar 2018). The process usually requires domain experts to sit down with engineers to construct complex interaction patterns in the form of explicit dialogue trees. This process quickly becomes intractable. Existing end-to-end solutions to chatbots, on the other hand, require little expertise to build (but a lot of data) and offer little to no control over the operational properties of the bot (Metz 2018). As a result, such data driven end-to-end techniques have been predominantly demonstrated in non-goal oriented settings.

Interactions on `MAi`

`MAi` attempts to address this problem in two ways, by –

– Reducing the expertise and effort required to specify a dialogue agent by abstracting away key dialogue components in the interface and providing iterative assistance during the design of the bot.

– Allowing the designer to examine the agent created by investigating the dialogue tree generated in the backend or by directly chatting with it.

The key features of the interface are shown in Figure 1. The domain writer can choose to design an agent either using the panel in the middle or the one on the left. The former allows specification of the dialogue model directly in terms of dialogue actions, their realizations in implementation (determiners), context variables they operate on and their relationships between themselves. The latter allows specification of the domain at a higher level of abstraction – i.e. in terms of the domain knowledge that the dialogue is supposed to occur in. The key insight here is that procedural knowledge of the dialogue process is largely (if not entirely) transferable across domains, and thus a generic dialogue agent can be bootstrapped on given domain knowledge. The designer can, of course, interactively refine this model further based on the needs of that particular domain. We anticipate that this modeling paradigm will not only reduce the effort in building robust chatbots for enterprise use but also make the design process of chatbots more easily accessible to a wider range of users (and thus to a wider range of uses).

The attached video walkthrough of the interface touches upon three use cases that build upon these concepts.

Dialogue Model Abstraction The interface (middle panel) requires the domain-writer to specify what information the dialogue agent would need to gather, and how (e.g. by asking versus hitting API endpoints on the cloud) and relationships among these variables in terms of pre/post-conditions of the actions. They can also specify if these actions need to have follow-ups (such as changing of the information or confirmation of the same in course of the dialogue). The backend compiles this specification into two forms – one to be consumed by the implementation of the bot in terms of what endpoints to hit for executing an action, the other to be used by a non-deterministic planner that can orchestrate the dialogue process. The domain-writer can also visually inspect the generated dialogue tree at any point to better understand the dependencies in the domain and identify unmodeled constraints or possible undesired dialogue patterns.

Bot in a Click The interface (left panel) also allows the domain-writer to only specify the high-level features of the domain (in terms of tasks in it and dependencies among them) and generate a basic dialogue agent directly from this. This mode of specification thus does not require any consideration of the dialogue patterns but rather the domain in which the agent is going to converse. The domain-writer has the option of using this mode, or the previous one which required direct specification of dialogue actions, or a mixture of both.

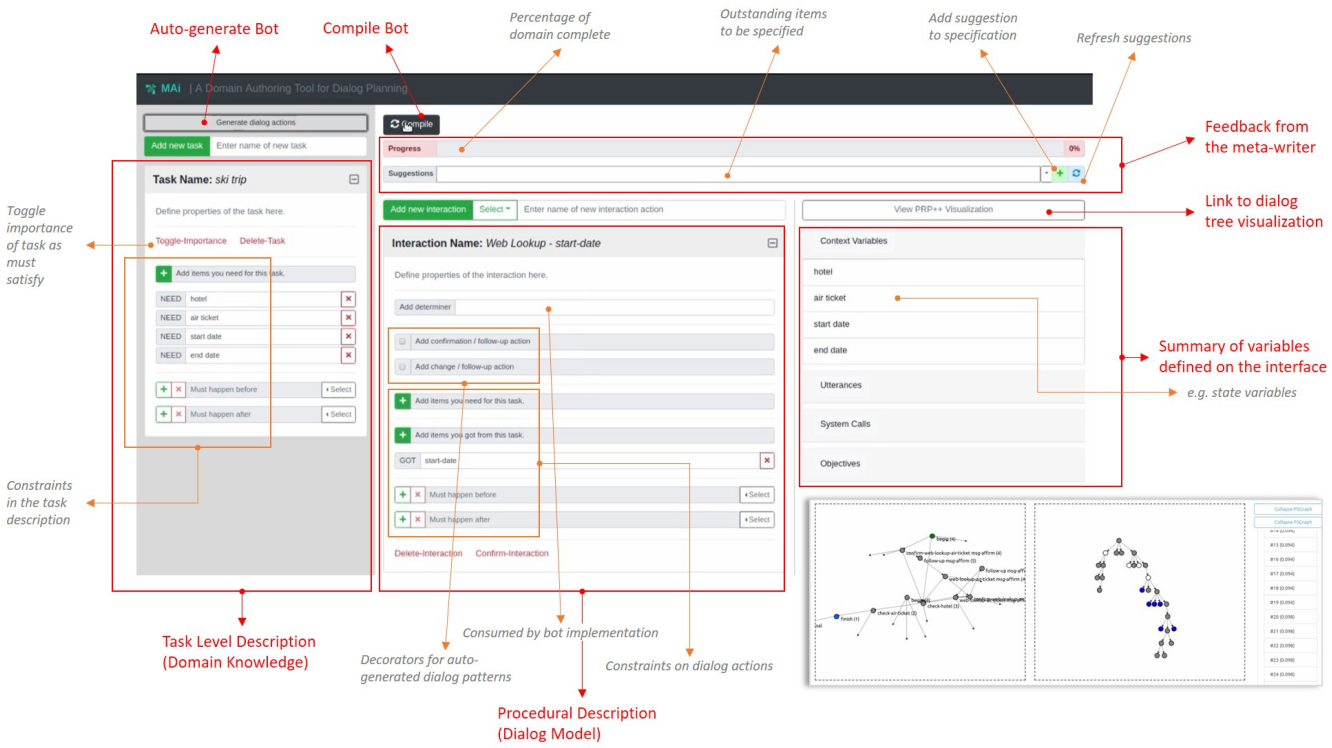

Figure 1: Different functionalities on MAi to aid in the bot specification process. Inset shows the generated dialogue tree.

The Meta-Writer The automated generation of the bot from the domain description is done by a "meta-writer" that casts the domain writing process as a planning problem. This meta-problem models the minimum requirements of a dialogue model given the nature of the domain specification and monitors for fulfillment of those basic requirements as the domain writer is constructing the bot, before the bot can be launched. If there are outstanding items to be modeled, it suggests possible dialogue actions to the user who can directly add them into the model of the agent. The meta-problem is the same model that is being used to generate the "bot in a click" as described above by directly populating the model of the bot with its suggested actions and allowing the domain-writer to edit them interactively.

Backend Technologies

The AI components in the backend is primarily built around existing technologies from the planning community.

– The meta-writer uses (Ramırez and Geffner 2009) to compile the domain writer's actions on MAi into observations that can be compiled into a meta-planning problem, as described before. It uses Fast-Forward (FF) (Hoffmann 2003) as the underlying planner.

– The execution model of the dialogue agent itself is realized through a non-deterministic planning domain (Muise, McIlraith, and Beck 2012). This allows for modeling of complex dialogue patterns and construction of large dialogue trees in a declarative fashion. Details of this are outside the scope of this discussion.

Demonstration Outline

Demonstration participants will be able to:

– Specify a dialogue agent from scratch;

– Interact with the MAi interface to iteratively refine the agent's model;

– Visualize the complex dialogue trees generated from the specification; and

– Interact / chat with the resulting bot.

A video of the model specification process is attached (https://ibm.box.com/v/aaai-19-demo-mai).

References

Hoffmann, J. 2003. The Metric-FF Planning System: Translating "Ignoring Delete Lists"to Numeric State Variables. *JAIR* 20:291–341.

Metz, R. 2018. Microsoft's neo-Nazi sexbot was a great lesson for makers of AI assistants. https://goo.gl/TF8DQx. MIT Technology Review.

Muise, C. J.; McIlraith, S. A.; and Beck, J. C. 2012. Improved Non-Deterministic Planning by Exploiting State Relevance. In *ICAPS*.

Ramırez, M., and Geffner, H. 2009. Plan Recognition as Planning. In *IJCAI*.

Sreedhar, K. 2018. What it takes to build enterprise-class chatbots. https://goo.gl/fRDkDn. Chatbots.