Rapid Prototyping of Form-driven Dialogue Systems Using an Open-source Framework

Svetlana Stoyanchev
Interactions Corporation
New York, USA
sstoyanchev@interactions.com

Pierre Lison
Language Technology Group
University of Oslo, Norway
plison@ifi.uio.no

Srinivas Bangalore
Interactions Corporation
Murray Hill, USA
sbangalore@interactions.com

Abstract

Most human-machine communication for information access through speech, text and graphical interfaces are mediated by *forms* – i.e. lists of named fields. However, deploying form-filling dialogue systems still remains a challenging task due to the effort and skill required to author such systems. We describe an extension to the OpenDial framework that enables the rapid creation of functional dialogue systems by non-experts. The dialogue designer specifies the slots and their types as input and the tool generates a domain specification that drives a slot-filling dialogue system. The presented approach provides several benefits compared to traditional techniques based on flowcharts, such as the use of probabilistic reasoning and flexible grounding strategies.

1 Introduction

Dialogue systems research has witnessed the emergence of several important innovations in the last two decades, such as the development of information-state architectures (Larsson and Traum, 2000), the use of probabilistic reasoning to handle multiple state hypotheses (Young et al., 2013), the application of reinforcement learning to automatically derive dialogue policies from real or simulated interactions (Lemon and Pietquin, 2012), and the introduction of incremental processing methods to allow for more natural conversational behaviours (Schlangen et al., 2010). However, few of these innovations have so far made their way into dialogue systems deployed in commercial environments (Paek and Pieraccini, 2008; Williams, 2009). Indeed, the bulk of currently deployed dialogue systems continue

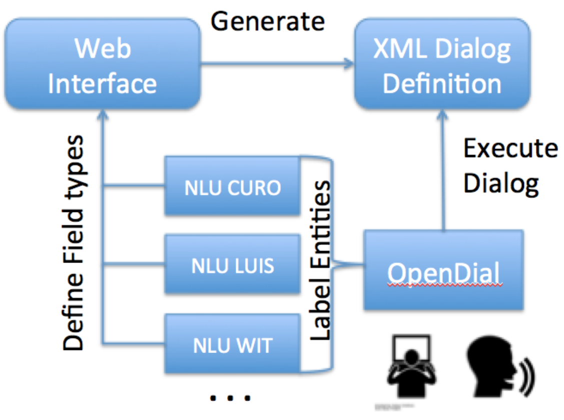


Figure 1: Architecture overview.

to rely on traditional hand-crafted finite-state or rule-based approaches to dialogue management using commercial or proprietary tools generating VoiceXML. The key reasons for this *status quo* are the need for the dialogue designer to (1) retain control over the system's behaviour, (2) ensure the system can scale to large numbers of users, and (3) easily author and edit the system's internal models. These features supersede their shortcomings. While authoring a system-initiative dialogue is quick and easy to maintain, authoring a user-initiative dialogue system in VoiceXML often results in large interdependent code bases that are increasingly difficult to maintain. Furthermore, these dialogue systems cannot capture multiple state hypotheses nor optimise the dialogue by learning from previous interactions.

Meanwhile, various dialogue authoring frameworks have been developed in academia to facilitate the development of dialogue systems by authoring state update rules (Bohus and Rudnicky, 2009; P. Lison, 2015). In particular, OpenDial, an open source dialogue system framework based on a information-state architecture, allows system developers to easily specify and edit dialogue behaviours, which is a crucial requirement for commercial conversational applications. However, designing and maintaining dialogue systems using these tools remains a challenge.

In order to address this challenge and lower the entry barrier for authoring dialogue systems, we demonstrate a web-based tool that allows a user to create a form-filling dialogue system automatically by simply specifying a form template. The architecture of the system is shown in Figure 1. The dialogue designer using the authoring tool specifies a form template – as a list of slots associated with their corresponding semantic types – as input. The tool compiles the form template into a specification to drive the dialogue management framework, OpenDial. Natural language understanding is provided through the use of cloud-based APIs. The authoring tool satisfies the requirements of maintaining control of the dialogue flow with the use of probabilistic modelling techniques, thus allowing simple authoring of mixed-initiative slot-filling dialogue systems.

Our target audience includes both researchers and industry practitioners. A fully-functional spoken interface to a system, such as hotel reservation, airline booking, or mortgage calculator, can be generated using the tool by a non-expert in dialogue systems. The generated domain specification can be further edited by the system developers in order to integrate more advanced functionality such as escalated per-field prompts or customized language generation.

The rest of this paper is structured as follows. The next section presents the web-based tool, the generated OpenDial domain file, and the software bridges to external NLU services. Section 3 describes a preliminary evaluation, while Section 4 relates the system to previous work.

2 System

We rely on OpenDial as underlying framework (P. Lison, 2015) for dialogue management. OpenDial has been previously used for human–robot interactions, in-car driving assistants, and intelligent tutoring systems (Lison and Kennington, 2016). It is also a popular platform for teaching advanced courses on spoken dialogue systems.

2.1 Form-to-System Generation

We created a web-based tool that generates an (XML-encoded) OpenDial dialogue domain from a form specification. The web tool allows the dialogue designer to configure any number of form fields by specifying a field name, a corresponding semantic type, a natural language question for eliciting the field value, an implicit confirmation sentence, and a optional set of constraints between the slots. Figure 2 illustrates the interface for defining a form for hotel reservations with four fields: *location, arrival, duration,* and *departure*. Fields can also be marked as "optional", and can be mutually exclusive with other fields (for instance, the "duration" of a hotel stay and its "end date"). It should be noted that the *NL Question* and *NL Implicit Confirmation* can reference the values of previous slots, such as e.g. "When are you arriving in *location*". This enables the dialogue designer to implement implicit grounding strategies. When the form is submitted, the authoring tool generates the corresponding domain file.

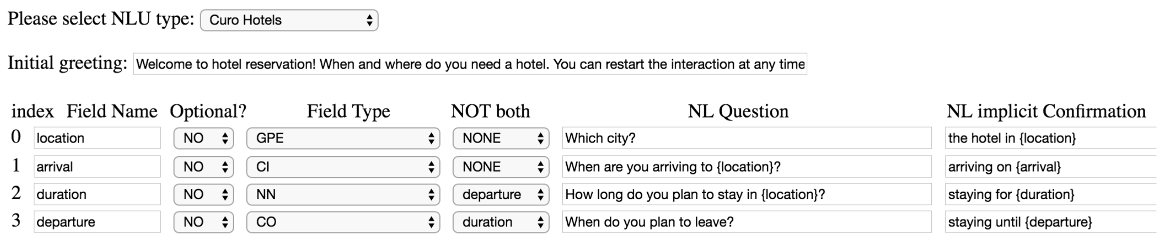
2.2 Domain file

OpenDial stores domain-specific information in a *domain file*, which is encoded in XML. The domain file specifies the following information:

- The initial dialogue state.
- A collection of domain models, which are themselves composed of probabilistic rules.
- General configuration settings, such as settings for the cloud-based NLU.

The dialogue state is represented as a Bayesian Network, allowing for explicit capture of uncertainty. For slot-filling tasks, the state variables capture the values for each slot, the recent dialogue history, a list of slots that are already filled and grounded, and a (possibly empty) set of mutual exclusivity constraints between slots. This dialogue state is regularly updated based on user inputs and subsequent system responses.

The probabilistic rules are expressed as *if-then-else* blocks associating logical conditions to probabilistic effects (see (P. Lison, 2015) for more details). The domain file generated by the web-based tool is composed of about fifteen rules responsible for (1) updating the slot values given the user inputs, (2) selecting the most appropriate system actions based on the current state, and (3) mapping these high-level actions to concrete system responses. The (probability and utility) parameters of these rules are initially fixed to reasonable

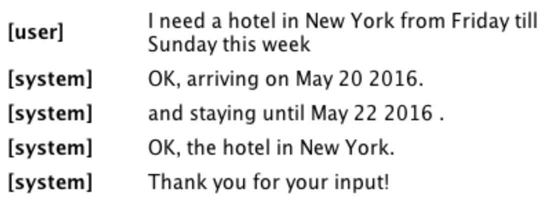


Please select NLU type: Curo Hotels

Initial greeting: Welcome to hotel reservation! When and where do you need a hotel. You can restart the interaction at any time

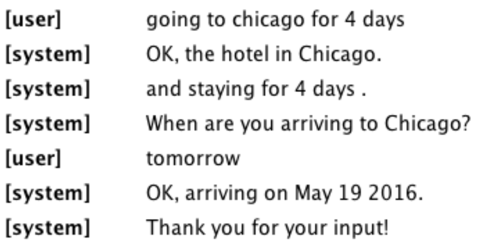
index	Field Name	Optional?	Field Type	NOT both	NL Question	NL implicit Confirmation
0	location	NO	GPE	NONE	Which city?	the hotel in {location}
1	arrival	NO	CI	NONE	When are you arriving to {location}?	arriving on {arrival}
2	duration	NO	NN	departure	How long do you plan to stay in {location}?	staying for {duration}
3	departure	NO	CO	duration	When do you plan to leave?	staying until {departure}

Figure 2: Form for generating a dialogue with hotel information domain.



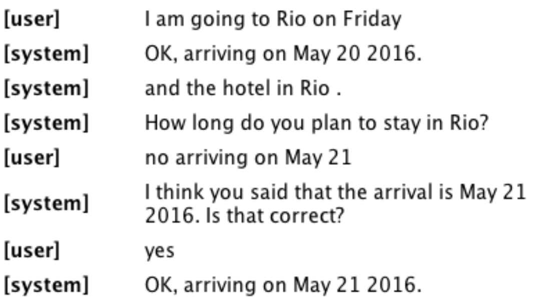
[user]	I need a hotel in New York from Friday till Sunday this week
[system]	OK, arriving on May 20 2016.
[system]	and staying until May 22 2016 .
[system]	OK, the hotel in New York.
[system]	Thank you for your input!

(a) Filling the form in one turn.



[user]	going to chicago for 4 days
[system]	OK, the hotel in Chicago.
[system]	and staying for 4 days .
[system]	When are you arriving to Chicago?
[user]	tomorrow
[system]	OK, arriving on May 19 2016.
[system]	Thank you for your input!

(b) Filling the form with multiple turns.



[user]	I am going to Rio on Friday
[system]	OK, arriving on May 20 2016.
[system]	and the hotel in Rio .
[system]	How long do you plan to stay in Rio?
[user]	no arriving on May 21
[system]	I think you said that the arrival is May 21 2016. Is that correct?
[user]	yes
[system]	OK, arriving on May 21 2016.

(c) User correction and grounding.

Figure 3: Dialogue Examples in the hotel reservations domain.

defaults, but the user is free to modify the values of these parameters (or estimate them from data if such interaction data is available).

The generated dialogue domain allows for mixed-initiative interactions where a user can choose any order and combination of fields for filling the form, including a single turn (Figure 3a) or in multiple turns (Figure 3b). In addition, the dialogue manager includes correction and grounding capabilities (Figure 3c). The user may interact with the system using either text inputs or speech (using third-party APIs such as Nuance or Curo for speech recognition and synthesis).

2.3 Natural Language Understanding (NLU)

In slot-filling applications, the main objective of natural language understanding is to label the user utterance with (application-specific) semantic entities. The entities identified through NLU can then be exploited by the dialogue manager to fill the fields of the form which in turn drives the next response. The mapping between NLU labels and state variables is established through the field types specified in the form (Figure 2).

We extend OpenDial to access cloud-based NLU services through HTTP endpoints. When a user selects an *NLU type* from the list of supported services, the values in the *Field Type* dropdown boxes for each field are populated with the NL labels in the selected NLU module. To add support for a given NLU service, we create a corresponding OpenDial module configured with the service's HTTP endpoint and session parameters. This module processes the output *json* file returned from the HTTP request to the service and extracts assigned semantic labels.

We have implemented NLU modules for publicly available cloud services from Microsoft and Facebook[1] and for a proprietary Curo NLU. This enables dialogue designers with a range of alternative NLU solutions, from using the already supported cloud NLU services to implementing their own NLU module in OpenDial.

3 Evaluation

For a preliminary evaluation, we asked five researchers from the lab to use the web interface and generate a dialogue system using pre-loaded hotel reservation form, evaluate it by running opendial as end-users, and explore the web interface by creating new systems. All of the participants were able to generate a hotel reservation form-filling in-

[1]https://www.luis.ai/, https://wit.ai

terface successfully. The participants were asked to fill out the hotel reservation form using multiple dialogue paths. On average the participants attempted four distinct dialogue paths and successfully completing three of them. All of the participants agreed that the tool *provides an effective method of generating a voice interface for a form* and four of the participants indicated that they would use it for generating spoken interfaces in the future (one was neutral).

4 Related Work

Several web-based NLU services have been recently launched by companies such as wit.ai (now part of Facebook), Microsoft, Nuance and api.ai [2]. These services provide cloud-based solutions for creating NLU for systems with simple web-based interfaces and active learning capabilities. Some of these tools have now been extended with basic dialogue management functionalities. These platforms can be employed by novices with no programming or speech experience to author and deploy spoken interfaces.

Similar to these commercial solutions, the presented authoring tool aims at lowering the entry barrier for dialogue developers wishing to quickly create functioning dialogue systems. Our solution is intended for both dialogue system researchers and commercial companies that still predominantly use VoiceXML-based platforms and have restrictions on transferring customer data to third-party services. We hope that both target audiences will benefit from the ability to deploy the system on a proprietary server, with full control over the dialogue flow, easy access to third-party ASR, NLU and TTS components, and ability to perform probabilistic reasoning and optimize dialogue policies from data.

5 Conclusions and Future Work

We have presented a web-based authoring tool to facilitate the creation of slot-filling dialogue systems. Based on a simple form template, the tool generates an XML domain file that specifies the system behavior, while intent recognition is delegated to a cloud service, either third-party or proprietary. We aim at bridging the gap between spoken dialogue research and conversational systems deployed in the industry by providing an open-sourced tool that combines simple authoring, full control of the dialogue flow with the ability to optimize from historical interaction data.

As future work, we wish to extend the tool to handle multiple forms and design an interface for describing system behaviors in a multi-form system. We also intend to use the system for research on clarification strategies and evaluating benefits of joint ASR and NLU processing in dialogue.

[2]http://wit.ai, https://www.luis.ai, https://api.ai, https://developer.nuance.com/mix.

References

D. Bohus and A. Rudnicky. 2009. The RavenClaw dialog management framework: Architecture and systems. *Computer Speech & Language*, 23(3):332–361.

S. Larsson and D. R. Traum. 2000. Information state and dialogue management in the TRINDI dialogue move engine toolkit. *Natural Language Engineering*, 6(3-4):323–340.

O. Lemon and O. Pietquin. 2012. *Data-Driven Methods for Adaptive Spoken Dialogue Systems: Computational Learning for Conversational Interfaces*. Springer.

P. Lison and C. Kennington. 2016. OpenDial: A toolkit for developing spoken dialogue systems with probabilistic rules. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (System Demonstrations)*, Berlin, Germany.

P. Lison. 2015. A hybrid approach to dialogue management based on probabilistic rules. *Computer Speech & Language*, 34(1):232 – 255.

T. Paek and R. Pieraccini. 2008. Automating spoken dialogue management design using machine learning: An industry perspective. *Speech Communications*, 50(8-9):716–729.

D. Schlangen, T. Baumann, H. Buschmeier, O. Buß, S. Kopp, G. Skantze, and R. Yaghoubzadeh. 2010. Middleware for Incremental Processing in Conversational Agents. In *Proceedings of the 11th SIGDIAL meeting on Discourse and Dialogue*.

J. D. Williams. 2009. Spoken dialogue systems: challenges, and opportunities for research (invited talk). In *Proc IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU), Merano, Italy*.

S. Young, M. Gačić, B. Thomson, and J. D. Williams. 2013. POMDP-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.