The DialPort Portal: Grouping Diverse Types of Spoken Dialog Systems

Tiancheng Zhao⋆1, Kyusong Lee⋆1,2, and Maxine Eskenazi1

1 Language Technologies Institute, Carnegie Mellon University
{tianchez,max}@cs.cmu.edu,
2 Pohang University of Science and Technology
kyusonglee@postech.ac.kr

Abstract. This paper describes a new spoken dialog portal that connects systems produced by the spoken dialog academic research community and gives them access to real users. We introduce a distributed, multi-modal, multi-agent prototype dialog framework that affords easy integration with various remote resources, ranging from end-to-end dialog systems to external knowledge APIs. To date, the DialPort portal has one chatbot and has successfully connected to the multi-domain spoken dialog system at Cambridge University, the NOAA (National Oceanic and Atmospheric Administration) weather API and the Yelp API.

1 Introduction

The advent of Siri and other agents has generated interest in spoken dialog research. These applications have sparked the imagination of many and led them to believe that speaking to intelligent agents is useful. The research community needs to profit from this interest by creating a service for the general public that can gather real user data that can be used to make dialog systems more robust and can also be used to carry out comparative studies. Industry already has access to large data sets and sometimes to pools of real users that are viewed as strategic competitive resources and so not shared with the research community. Yet much fundamental research remains to be done, such as signal processing in noisy conditions, recognition of groups of difficult users (like the elderly and non-natives), management of complex dialogs (such as multi-party meetings, negotiations, and multimodal interaction), and the automatic use of meta linguistic information such as prosody. It is extremely difficult for any one group to devote time to collecting a significant amount of real user data. The users must be found and kept interested and the interface must be created and maintained. One data gathering portal that all dialog systems can be connected to gives potential users a variety of interesting applications, much in the way that virtual assistants do not only provide information about scheduling. The DialPort portal was created for this purpose.

One notable effort in collecting real user data for the community as well as giving access to the system to run studies is the Lets Go System [12] from the Dialog Research Center at Carnegie Mellon (DialRC). But research is carried out in other areas, beyond simple form filling. Just as one research group cannot attract a diverse pool of regular

⋆ Both authors equally contributed to this work
users, one group cannot cover all of the possible applications, such as virtual humans and robots with multimodal communication. Thus the goal of DialPort is to attract and maintain a pool of real users for a group of spoken dialog applications. DialPort will be attractive if it combines both information-giving and entertainment. In that respect, the addition of chatbots will be welcome. Not only can chatbots catch out of domain (for the information-giving systems) questions and provide an answer, but they can also be addressed on their own as a source of conversation.

The first year goal is to create the portal and link it to other systems. Once the working portal can give a variety of useful information and some entertainment, a service such as Prefinery\(^3\) will be used to attract the real users. These services solicit potential users, giving bonuses for signup and usage as well as for getting friends to sign up. In this paper, we present the DialPort portal that will link many different research applications and will provide real user data. Section 2 discusses related work; Section 3 discuss the core modules and explains the how various systems and resources can be connected to the DialPort; Section 4 reviews current progress and Section 5 concludes.

2 Related Work

In order to simplify the development process of new dialog systems, SDS development frameworks have been proposed by both academia and industry. Popular frameworks from academia include VoiceXML [10], the CSLU toolkit [16], Olympus [2], Trindikit [8], Opendial [7] and the Virtual Human Toolkit [5] and many others. Recently several services have been released from industry. For example, Microsoft released the Language Understanding Intelligent Service (LUIS) [21] which helps software developers create machine-learning powered language understanding for specific domains. The HALEF (Help Assistant Language Enabled and Free) framework from ETS leverages different open-source components to form an SDS framework that is modular and industry-standard-compliant [14].

Different from past work, DialPort strives to discover how to combine various spoken dialog systems (i.e developed with any SDS frameworks and serving arbitrary types of purposes) and resources (e.g structured database, web APIs) into a single SDS, creating a homogeneous user experience and a single stream of real user data. The most relevant work related to this question is the research in multi-domain dialog systems. Past approaches usually followed a two-stage framework [6, 11], in which the first stage classifies the domain and the second stage forwards the user’s request to the relevant single-domain dialog manager. This method has shown promising results for scaling up dialog systems to handle multiple domains. DialPort differs from previous frameworks in this area by proposing the concept of a multi-agent dialog system. This system combines both goal-driven and non-goal-driven dialog agents that have been independently developed by different research teams. The task of DialPort is to judiciously assign the user’s utterance to the most relevant dialog agent and to carry out complex nested conversations with real users. The long term goal is to enable any research group to connect their SDS to DialPort using a lightweight integration protocol. DialPort makes it easy

\(^3\) https://www.prefinery.com/
for real users to access many state-of-the-art dialog system services all in one place through a universal web-based entry point.

## 3 System Description

Figure 1 presents the system architecture, comprised of three sections: User Interface, DialPort, and Remote Agents/resources.

![System Architecture Diagram](image-url)

**Fig. 1.** The overall architecture of DialPort. In session arrows indicate the user inputs are directly forwarded to the remote agent that is in focus. The solid boxes and arrows are already deployed while the dashed ones are in progress as of this writing.

### 3.1 User Interface

The user Interface is the publicly available front end for real users\(^4\). It is in charge of both the visual and audio representing each dialog system. The audio interface uses the Google Chrome Speech ASR API to transform the user’s speech into text and the Google Chrome TTS API to convert DialPort’s text output into speech. The visual representation uses WebGL Unity 3D. Skylar is the virtual agent for DialPort and for the Cambridge University dialog system agent at present. It interacts with users and has 3D animated embodiments powered by the Unity 3D Engine\(^5\). We have also tried using one agent for each separate dialog system. Jasmin, a librarian, was created for the Cambridge system. She spoke when the control was turned over to Cambridge. But there are several issues in having a separate agent for each system:

- It would be expensive to have well-developed characters created for each system.
- The screen could get too full if there are, say, 10-15 agents appear in the screen.

\(^4\) [https://skylar.speech.cs.cmu.edu](https://skylar.speech.cs.cmu.edu)
\(^5\) [unity3d.com/](https://unity3d.com/)
But there are reasons to have separate agents:

- Each agent has a manner of interaction that gives it its unique character.
- We need to give credit to the system that is being called into use.

Thus, in the future we intend to use one main character, Skylar, and many talking heads. If we find a new system has full body agent gesture, we will create a new agent.

### 3.2 DialPort

DialPort is scalable and distributed. Its central message broker is ActiveMQ, a well-known open source message broker. ActiveMQ allows us to easily connect multiple components in order to create a larger system. Building on ActiveMQ, DialPort has four main modules: the HTTP API Server, the Natural Language Understanding (NLU), the ReinForest Dialog Manager (DM) [22] and the Natural Language Generation (NLG).

With the exception of the ReinForest DM, the modules are RESTful (Representational state transfer) web services: they do not consider any state information when handling requests. All contextual information about a dialog is maintained by the ReinForest DM. The HTTP API Server is the front gate of DialPort. It converts the incoming HTTP messages into proper ActiveMQ messages and sends them to the NLU. The NLU outputs a semantic frame that contains the original utterance along with: entities, an intent and a domain. Given the user input annotated by the NLU, ReinForest updates its internal dialog state and generates the next system response. Then the NLG is responsible for transforming the system response to its natural language surface form.

### 3.3 Remote Agents and Resources

Easy integration with remote agents is a major contribution of the proposed architecture. We define a remote agent as any external autonomous dialog agent. We further divide the remote agent into speech remote agent and text remote agent.

**Speech Remote Agent:** this is a self-sustaining spoken dialog system that only has a public audio API. Therefore, a speech remote agent expects audio streaming input and returns an audio clip that contains the system’s spoken response, which allows fully incremental sophisticated SDS. DialPort does not presently support this type of remote agent due to the difficulty of dealing with real-time audio streaming amongst remote servers. This will be dealt with when connection to a system of this type is proposed. When DialPort hands over control to a speech remote agent, the user interface will directly forward the audio to the remote system and thus bypassing the internal modules of DialPort until the control is handed back.

**Text Remote Agent:** this is a turn-based dialog system that provides an HTTP API, which inputs the ASR text output of the latest turn and returns the system response in text form. It should be noted that even end-to-end spoken dialog systems can belong to text remote agents as long as they can provide a text API that bypasses its audio front end. The Cambridge SDS [4] in Figure 1 is one example. It has its own VoIP audio server and also provides a HTTP API server that directly connects to its NLU, DM and NLG. Therefore, when the Cambridge system connects with DialPort, the latter sends
the transcribed speech to Cambridge’s text-based API and bypasses its VoIP server. When DialPort hands over the control to a text remote agent, DialPort is still in charge of turn-taking, ASR and TTS, but bypasses the NLU, DM and NLG of DialPort, until the session with the remote system is finished.

*Remote Resources* are knowledge resources, e.g. a database of bus schedule or a web API. DialPort is in charge of all of the dialog processing (NLU/DM/NLG) and uses the remote resources as knowledge back-end same as traditional goal-driven SDS [12].

### 3.4 Group Diverse Types of Agents

The agents connected to DialPort include both information-giving (goal-driven) and entertaining agents. The current ReinForest DM decides the next responsible agent based on the dialog state and hand-crafted rules. Each goal-driven remote agent has a list of subscribed topics, e.g. food or weather, that it is capable of dealing with. Then if the dialog state indicates that the users are looking for certain topics, the DM will assign the conversation floor to the agent that subscribes to the desired topic. If there are multiple agents subscribing to the same topic, the current mechanism will randomly choose an agent. Developing more sophisticated strategies of resolving such conflicts, for instance choosing the one with better user feedback, is part of our future research.

On the other side, non-goal-driven agents (e.g. a chatbot) usually do not have explicit topics that they address, so at present that they are chosen whenever the user utterance is outside of all the domains covered by the goal-driven agents. The current chatbot is formulated as a text remote agent that expects an utterance from the user and returns the system response with a confidence score. The confidence score is calculated on the current users input. In the future, we should also consider long-term discourse history into the chatbot. As assigning the users to the correct domain, it is an open research problem that when to trigger a chatbot. The current hand-crafted strategy is as follows. When a user’s input cannot be handled by ReinForest DM such as low ASR/SLU score and the confidence of the chatbot is high, such as out-of-domain utterances (e.g., "you are smart"), factoid questions (e.g., who founded Microsoft?, how much is an iPhone?), the dialog policy triggers the chatbot to generate the next system response. The chatbot is used for error handling strategy, so the chatbot will return the floor to ReinForest DM every turn, even if chatbot can potentially continue the conversation. However, in the future, other chatbots can join to DialPort to handle specific topics such as movie discussion, gaming and etc. Thus, a user can talk with the specific chatbot agent continuously without getting intervened and the function of those chatbots will be beyond error handling strategy.

### 3.5 Integration Protocol

This section describes the integration protocol for linking a remote system into DialPort. Since the *speech remote agent* is not supported at present, the protocol only concerns the *text remote agent* and the *remote resources*.

**Text Remote Agent:** The participating research group needs to implement the following two high level API functions. DialPort currently supports HTTP (GET/POST) calls for connecting to *text remote agents.*
– **NewCall**(id, s₀): The input parameters include the user id and current dialog state s₀. The output is the first system response. The initial state s₀ enables the remote agent to skip redundant questions that were already asked in the previous conversation. DialPort calls this function to initialize a new session with the remote agent. Also, it is up to the remote research group how they use s₀, so the remote agent can operate totally independently. The exact format of s₀ can be customized if needed.

– **Next**(id, utt): The input is the users’ utterance and the output is the system’s response and an end-of-session flag. After **NewCall**, DialPort continues to call **Next** to obtain the next system response until the end-of-session flag is true. Thus, the remote agent has complete autonomy during its session.

**Remote Resources** are functions that output a list of matched entities, given a list of input constraints. Therefore, any common database format (e.g. SQL) or service API (e.g. Yelp API) can be a knowledge remote agent. The resource provider would need to give specifications of all the input parameters (both required and optional) needed for the database search and specify any dependencies among the input parameters. DialPort will then construct slot-filling dialog domains that are powered by the corresponding remote resources.

**Data Sharing:** The purpose of DialPort is to collect and share real user data among all participating parties. So when a text remote agent finishes its session, it should be responsible for sending a dialog report along with the response to the last **Next** call. The report should contain all the essential information about the conversation, such as the utterance at each turn. The final report format will be found on the DialPort website. Speech data that is collected will be made publicly available by the group who collected the data. The resulting text corpora will be available at DialPort website.

### 4 Integration Example

#### 4.1 Cambridge SDS as a Goal-driven Text Remote Agent

The first academic system that was connected to DialPort is from Cambridge University. The Cambridge SDS is a slot-filling dialog system that provides information about restaurant and hotels in Cambridge England. Therefore, when the users request information about restaurants or hotels in Cambridge England, the Skylar will try to hand over the control to the Cambridge SDS. The Cambridge Dialog Research group implemented an HTTP API server that implements the **NewCall** and **Next** function defined in Section 3.5. **NewCall** expects in a JSON body that contains the user ID and the initial domain: restaurant or hotel. Also **Next** expects the N-best ASR outputs for the last turn and returns a JSON that contains the system response in text-form and a boolean flag that indicates whether or not the conversation with Cambridge is finished.

From the perspective of users, figure 2 shows the system when it was being tried with Jasmin. Skylar gives information about the weather and restaurants other than in Cambridge England. When the domain changed, the Jasmin avatar appeared. When the dialog on the connected systems topic is over, the user is handed back to Skylar. The transition is seamless from the users point of view. When the user is speaking with Jasmin, the logo in the background changes to the Cambridge log to indicate which system the user is speaking to (Figure 2).
4.2 Chatbot as a Non-goal-driven Text Remote Agent

When the users’ requests are not covered by domains of all the goal-driven dialog systems, DialPort uses a chatbot for non-understand error handling [3] and keeping user engaged. The current implementation uses an example-based chatbot, because the precision of the response can be controlled by a similarity threshold (Table 1-b). We use the publicly available large knowledge base, Freebase\(^6\) created by Google, to extend coverage. For example, if a user asks about a person, a location or the definition of a word, by using the Freebase ID extracted from the DBpedia spotlight and the Freebase property “common.topic.description”, the system can find the requested information. Therefore, the non-understanding error handling policy queries the chatbot agent with the out-of-domain user input and the example-based chatbot calculates the similarity scores using sent2vec [13] (rather than a traditional vector space model). If the similarity score is over 80%, the system response is selected from the chatbot agent. Otherwise ReinForest follows a deterministic error handling strategy which first asks users to ”rephrase their request” and then provides more instructions if the error cannot be recovered.

4.3 Seamless Switching

There are several issues involved in seamlessly switching from one dialog system to another.

---

\(^{6}\) [http://freebase.com](http://freebase.com)
1. The systems must be able to switch in a timely manner so that there is little chance of barge-in or repetition when users thinks the system has not heard them.
2. If one system is not available, Skylar is responsible for answering the request, if possible. If Cambridge is not available, the user will be told that the Cambridge agent is out of the office and in the same turn will get the information they asked for.
3. Starting a conversation with a new agent means that Skylar must introduce that agent and relinquish the floor as that agent puts out its first turn.
4. Ending a conversation with an agent means that Skylar has to send its first utterance during the same turn as the end turn of the remote agent. But it also means that the remote agent, which has control of the dialog, has to have mechanisms to know when to relinquish the floor. It has to recognize a “thank you” as the end of a dialog, but it also has to realize when the user has changed topics (and so the topic is out of domain for this agent and Skylar needs to be given the information in order to figure out where to direct the user’s query). Some system developers may choose to allow Skylar to intervene during their dialog.

4.4 Yelp Services as Remote Resources

Yelp provides a public API \(^7\) that allows for restaurant search. DialPort formulates the problem into a slot-filling dialog domain with two slots: user location and preferred food type. The NLU is done via Stanford Named Entity Recognizer Toolkit [9]. Dialog management is handled by the multi-domain dialog manager ReinForest.

5 Conclusions and Future Directions

We propose a novel shared platform, DialPort, which can connect to many research systems enabling them to test new application ideas and gather real user data. In this paper, we have described the architecture of the user interface, DialPort, the virtual agents, and the (non)goal-driven dialog managers and we have reported the current progress of the DialPort project. An important purpose of this paper is to encourage our colleagues to link their systems to DialPort so that we can help them to collect real user data.

In terms of future work, DialPort is being linked to more SDS and it will start to attract users to these systems as soon as it passes a series of stability tests and when several other remote agents in order to broaden interest.

6 Acknowledgements

This work was funded by NSF grant CNS-1512973. The opinions expressed in this paper do not necessarily reflect those of NSF. We would also like to thank Alan W Black for discussions on this paper.

\(^7\) https://www.yelp.com/developers/documentation/v2/overview
References


