"LET'S GO, DUDE!" USING THE SPOKEN DIALOGUE CHALLENGE TO TEACH SPOKEN DIALOGUE DEVELOPMENT

Helen Hastie, Nicolas Merigaud, Xingkun Liu, Oliver Lemon

Department of Mathematical and Computer Science
Heriot-Watt University
{h.hastie, nm210, x.liu, o.lemon}@hw.ac.uk

ABSTRACT

Educational tools are essential in teaching the field of Spoken Dialogue Systems given the complexity and variety of disciplines involved. This paper describes DUDE, a Dialogue and Understanding Development Environment that enables researchers and students to efficiently create Information State Update (ISU) Spoken Dialogue Systems using large scale databases with minimal programming and grammar development. The experience of creating real Spoken Dialogue Systems that they can call through a VoiceXML platform, increases students' motivation and improves learning by grounding key concepts, including introducing students to ISU dialogue modelling.

1. INTRODUCTION

Industrial and academic approaches to researching and developing Spoken Dialogue Systems (SDS) have varied in terms of their objectives, size of systems built and research questions asked. Industry has typically focused on more short-term goals such as creating scalable and robust systems in a cost efficient manner. While, traditionally, academia has been more interested in exploring a larger variety of research questions with longer term scientific goals. Until recently, creating Spoken Dialogue Systems, particularly using large-scale industrial databases, required resources not available to academia and, therefore, academic Spoken Dialogue Systems were typically limited to small "toy" systems. Platforms such as DUDE (Dialogue and Understanding Development Environment) [1] have enabled researchers and students to create large-scale Spoken Dialogue Systems allowing them to better validate research results and increase the impact of their work.

DUDE is a development environment that automatically generates dialogue systems from a dialogue flow model and a database. These generated Spoken Dialogue Systems are then deployed on an industry standard VoiceXML platform. Specifically, the deployed system works by dynamically generating context-sensitive VoiceXML pages. The dialogue move of each page is determined in real time by the dialogue manager, which is an Information State Update (ISU) engine [2]. The main advantages that DUDE has over other similar tools and development environments is that it does not involve any programming or grammar writing and it compiles into an end-to-end Spoken Dialogue System that contains a mixed initiative ISU dialogue manager and can be called from a phoneline. DUDE has been used for rapid prototype development using business-user resources [1] and also as a teaching tool for the 2010 Spoken Dialogue Challenge (SDC).

Educational tools for the area of Spoken Dialogue Systems are essential given the number of modules that make up an SDS and their individual complexity. The field of Spoken Dialogue Systems crosses many disciplines and attracts students from diverse backgrounds: from Linguists interested in syntactic and semantic formalism, to Computer Scientists interested in architecture design, to Electrical Engineers interested in improving ASR/TTS. In addition, students vary significantly in programming skills, which makes teaching such a course on Spoken Dialogue Systems very challenging. A platform such as DUDE allows one to reach out to such a diverse audience in an effective and time efficient manner.

Through DUDE, the students are educated on the process of creating an SDS using real large-scale databases from industrial partners. The systems created can be

called from any phoneline, thus increasing the visibility of the students' systems and also boosting the confidence of the individual student in terms of their ability to develop a system in a short period of time that is commercially viable. In addition, vehicles such as the Spoken Dialogue Challenge allow systems to be rapidly deployed and tested using real users, which can be outside the scope of a Master's project.

In this paper, we look briefly at systems currently being used as teaching tools. We then describe the DUDE platform and the resulting Spoken Dialogue Systems. Throughout the paper, we refer to a case study system developed by a student for the 2010 Spoken Dialogue Challenge using the Pittsburgh Port Authority Transit database to enquire about bus times in and around Pittsburgh, U.S.A..

2. PREVIOUS WORK

[3] describes the Regulus Open Source Project which has been used in a course for students with limited computer programming skills. Regulus includes a number of resources including compilers, IDE, a Regulus resource grammar for English, documentation and example dialogue systems. Where DUDE automatically compiles the whole dialogue system (including speech recognition grammars, NLU and dialogue manager), Regulus provides one tool that automatically compiles CFGs by taking a unification grammar (UG) together with a supplementary lexicon and compiles a domain-specific UG using Explanation Based Learning. This is then compiled into a CFG, which can be used with a speech recogniser.

[4] describes Dialog OS, which is an educational tool aimed at helping students grasp the main ideas of finite-state-based dialogue modelling. The main strength of this platform is the GUI workspace that allows the user to add and remove nodes in the Finite State Automata (FSA). Although this tool goes part way to reducing development time, FSA dialogue modelling is still a lengthy process. We believe that our process of automatically creating an Information State Update Dialogue Manager (DM) through DUDE does not only reduce development time but also provides a more flexible Dialogue Manager.

Voxeo's Evolution Designer (http://www.voxeo.com) is a web-based graphical interface that allows one to design call flow, system prompts and user responses. The system is then linked to a phone number for testing.

Similarly to Voxeo, the CSLU toolkit [5] contains a drag-and-drop authoring environment called Rapid Application Developer (RAD) for rapid dialogue development integrating facial animation, ASR, NLU and TTS with other features such as word-spotting, barge-in, dialogue repair, telephone and microphone. Through RAD, the students specify prompts, recognition vocabularies and actions.

The advantages of DUDE over the above-mentioned systems is that DUDE not only provides automatic grammar compilation but also generates an end-to-end ISU dialogue system. ISU systems are more flexible and require less development time than FSAs (as in Dialog OS) and hand-coded dialogue flow (as in the Voxeo and CSLU toolkits). DUDE also provides further exploration work by allowing students to look at and modify the update rules of the ISU DM. Finally, DUDE requires no programming experience or knowledge of grammars.

System: Welcome to the Pittsburgh bus information line.
You can say start over anytime.
Where are you going to take your bus from?
User: Hi yeah I want to go from McKeesport
System: Sure. Where do you want to go?
User: Duquesne.
System: Did you mean Duquesne?
User: Yes
System: At what time do you want to take it?
Say next bus or a time for example 8.40pm.
User: Around 4.15pm
System: I'm going to search for buses around 4.15pm.
Is that OK?
User: Yes.
System: On which day?
User: Today
System: I have found one bus. It is at 4.15pm.
There is a bus number 61C from McKeesport.
It will arrive in Duquesne at 4.21pm.
System: Is there anything else I can do for you?
User: No thanks
System: Thanks for calling. Good bye!

Table 1. Example dialogue

3. THE 2010 SPOKEN DIALOGUE CHALLENGE

As a case study of using DUDE as a teaching tool, we look to the 2010 Spoken Dialogue Challenge where the challenge was to create a Spoken Dialogue System that allows the callers to enquire about bus schedules using a real database provided by the Pittsburgh Port Authority. Table 1 gives an example dialogue from a bus scheduling system created using DUDE. The original bus information system, known as "Let's Go", was developed at Carnegie Mellon University [6] and has been running 'live' for several years with real callers seeking bus times.

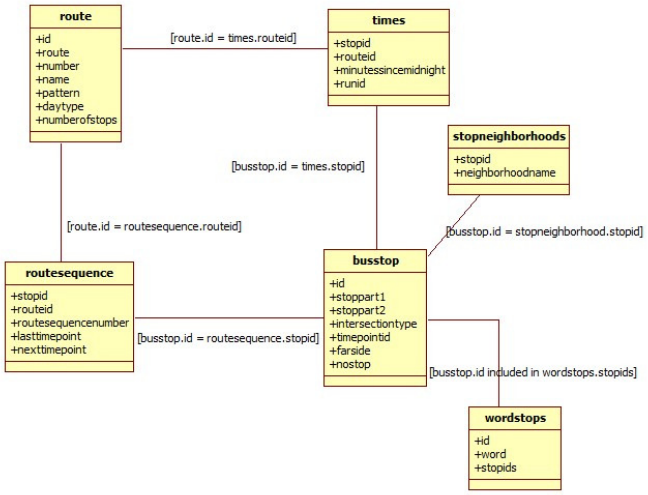

Fig. 1. Original database structure

The SDC database is an SQL database containing 6 tables as illustrated in Figure 1. The `busstop` table lists the bus stops in terms of an address, e.g. "Forbes avenue at Murray" is `busstop.stoppart1+busstop.intersectiontype +busstop.stoppart2.` `Route` refers to the bus route, e.g. `route.number` is 61C. The `times` table lists the times a bus leaves a `busstop`.

`Routesequence` links buses and `busstops` on the same route. To access a schedule for a specific bus, assuming that both the departure location and arrival location are `busstop` addresses, the SQL query would involve all of the above-mentioned tables. In addition, there are two other tables: the `stopneighborhoods` table represents the different neighbourhoods in Pittsburgh, e.g. "Shadyside"; finally `wordstops` are simple keywords associated with certain `busstops`, for example,
"Carnegie Mellon University" or CMU would be associated with the `busstop` address "Forbes Avenue at Morewood Carnegie Mellon".

INSERT INTO 'buses' VALUES (0,'MCKEESPORT', 'TRANSPORTATION MCKEESPORT CENTER BAY 2 MCKEESPORT_TAGN','4 15 A M','four twenty A M', 'DUQUESNE','GRANT SECOND DUQUESNE_TAGN', '4 21 A M','four twenty A M','61 C','McKeesport Homestead','I','SATURDAY');
INSERT INTO 'buses' VALUES (1,'MCKEESPORT', 'TRANSPORTATION MCKEESPORT CENTER BAY 2 MCKEESPORT_TAGN','4 15 A M','four twenty A M' ,'SECOND STREET AT GRANT', 'GRANT SECOND DUQUESNE_TAGN', '4 21 A M','four twenty A M','61 C','McKeesport Homestead','I','SATURDAY');
INSERT INTO 'buses' VALUES (2,'MCKEESPORT TRANSPORTATION CENTER AT BAY 2', 'TRANSPORTATION MCKEESPORT CENTER BAY 2 MCKEESPORT_TAGN', '4 15 A M','four twenty A M', 'SECOND STREET AT GRANT', 'GRANT SECOND DUQUESNE_TAGN','4 21 A M', 'four twenty A M','61 C','McKeesport Homestead','I',SATURDAY');
INSERT INTO 'buses' VALUES (3,'MCKEESPORT TRANSPORTATION CENTER AT BAY 2', 'TRANSPORTATION MCKEESPORT CENTER BAY 2 MCKEESPORT_TAGN', '4 15A M','four twenty A M', 'DUQUESNE','GRANT SECOND DUQUESNE_TAGN', '4 21 A M','four twenty A M','61 C','McKeesport Homestead','I','SATURDAY');

Table 2. Four example insert statements from the database

As the current version of DUDE does not handle multi-table databases, some pre-processing was needed. The 6 table database was transformed into a single large table where each line would represent a single bus trip at a specific time. Table 2 gives four example insert statements out of the 282,305 insert statements in the modified database.

A single line in the database says: "There is a [routenumber] from [departure location] at [departure time] on [daytype]. It will arrive in [arrival place] at

[arrival time]". In our case, the first example insert statement translates to "There is a 61C from MCKEESPORT at 4:15 AM on SATURDAY. It will arrive in DUQUESNE at 4:21 AM". The four example statements describe the same 61C bus leaving McKeesport at 4.15pm. This is to enable the user to say a mix of neighbourhood locations and bus stop locations. Concretely, for one specific bus there are the following statements:

- From `stopneighborhood` to `stopneighborhood`
- From `stopneighborhood` to `busstop`
- From `busstop` to `busstop`
- From `busstop` to `stopneighborhood`

4. THE DUDE PLATFORM

4.1. The Development Environment

The DUDE environment consists of a GUI that allows the student to design a task flow by creating, dragging and dropping nodes. There are two levels of nodes:tasks and slots. Figure 2 illustrates one such dialogue flow created using DUDE. For illustrative purposes this figure contains both the flow for the tasks (above) and for the slots (below). In this case the top level task node is just buses, however, DUDE does support multiple tasks such as buses and trains.

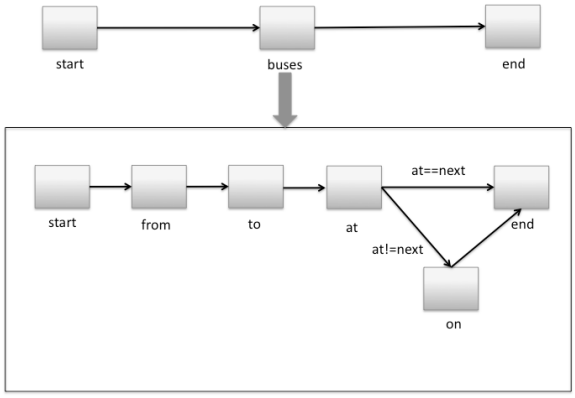

Fig. 2. Task and slot level dialogue flow

Drilling down on the buses task node allows one to create a flow for the slots, see the lower half of Figure 2. In this case the slots are: origin (the "from" node), destination (the "to" node), time (the "at" node) and day of the week (the "on" node). The arrows provide the default call flow, for example the system will prompt the user for the destination after it knows where the user is taking the bus from. However, as the system is mixed initiative, the user can provide information in any order and with multiple slots, for example "I want to go to the airport from Downtown". Conditions can be placed on the transition arrows, for example in Figure 2 if the caller says "next bus" then the dialogue information gathering part of the dialogue ends, otherwise it prompts the user for the day of the week.

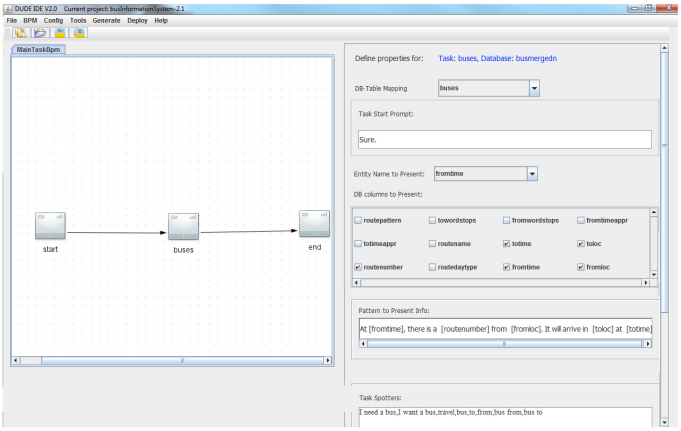

Fig. 3. The DUDE Development Environment to enter task level information

Once the task flow is complete, it is loaded into the main GUI where the student can define system prompts and add task spotter phrases and synonyms to the grammars. Figure 3 shows the GUI with the task flow on the left hand side and the properties pane for the buses task on the right hand side. In this pane the student can define the system prompt, the information to be presented to the user and the spotter phrases. Here the student is associating the phrases "I need a bus, I want to go to." with the buses task. This means that if the user says, for example, "I want to go downtown", the buses part of the task flow will be triggered. This would be particularly important if there were more than one task level node, e.g. buses and trains. Note that multiword phrases may also be defined. The defined spotters are automatically compiled into the grammar for parsing and speech recognition. By default, all the lexical entries for answer-types for the slots are also included as spotter phrases. DUDE checks for possible ambiguities, for example if the user simply said "downtown" without being prompted, the system would be unsure if this

was a destination or origin and would, therefore, trigger a clarification subdialogue to resolve the ambiguity at runtime.

Figure 4 shows the student specifying the required linguistic information to automate the "to" slot of the buses task. Here the student specifies the system prompt "Where do you want to go to?" and a phrase for implicit confirmation of provided values, e.g. "going to $X ", where $X is a variable that will be replaced with the semantics of the speech recognition hypothesis for the user input. The student also specifies here the answer type that will resolve the system prompt. There are predefined answer-types extracted from the SQL database, and the student can select and/or edit these, adding phrases and synonyms. In addition, they have the ability to define their own answer-types.

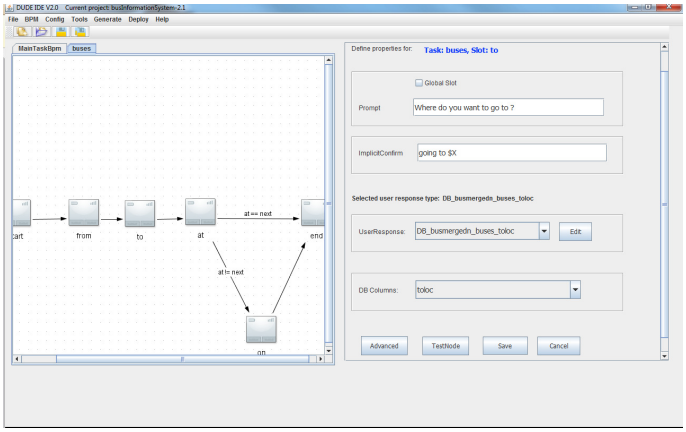

Fig. 4. The DUDE Development Environment to enter information for the destination slot

4.2. Dialogue Manager Generation

Once the necessary fields have been completed by the student, all that is left is to compile the dialogue system. The resulting dialogue system contains an Information State Update dialogue manager [2]. The DUDE environment is used to specify *domain-specific* dialogue strategies, which are combined with *domain-general* strategies. Specifically, the DM consults the task flow to determine what task-based steps to take next, such as asking for destination after establishing origin. XML format is used for the task flow, and is compiled into finite state machines consulted by the Spoken Dialogue System. Values for constraints on transitions and branching in the task flow, for example "ask for week day if user does not say next bus", are compiled into domain-specific parts of the Information State. General aspects of dialogue, such as confirmation and clarification strategies, are handled by the domain-general part of the DM. The domain-general dialogue manager was mostly abstracted from the TALK system [7].

4.3. Grammar Generation

Writing grammars for a domain can be very time consuming and due to the constraints of a student's course, often only small domains can be built. Using DUDE, however, students do not have to write a single line of grammar code. Within DUDE there are three types of grammars: (1) a core grammar, (2) a grammar generated from the database and task flow, and (3) dynamically generated grammars at run-time. The core grammar (1) was developed to cover basic information-seeking interactions. To create grammar (2), the system compiles relevant database entries and their properties into the appropriate "slot-filling" parts of a SRGS GRXML (Speech Recognition Grammar Specification) grammar for each specific task and slot node. Task level grammars are used to allow a level of mixed initiative, for example, if the system asks "Where do you want to go?" the user can reply with a destination slot filler and also any other slot type, such as "I want to go from the airport to downtown". The dynamically generated grammars (3), such as for bus times and route numbers currently in discussion, minimizes grammar size and makes the system more efficient. In addition to the above-mentioned grammars, using the development environment students are able to provide task spotter phrases and synonyms reflecting how users might respond. If these are not already covered by the existing grammar, DUDE automatically generates rules to cover them.

The generated SRGS GRXML grammars are used to populate the VoiceXML pages that are dynamically generated during the conversation. These VoiceXML pages are interpreted by a VoiceXML Platform Speech recogniser. In this case, we deploy our system to the Voxeo Platform (http://www.voxeo.com). The Automatic Speech Recogniser (ASR) and Text To Speech Synthesizer run on this platform with the ASR loading up the specific grammar defined by that VoiceXML page. As well as the W3C standard SRGS GRXML, DUDE is able to generate alternative grammar specifi-

cations such as SRGS ABNF (Augmented Backus-Naur Form), JSGF ABNF (Java Speech Grammar Format) and Nuance's GSL (Grammar Specification Language).

5. EVALUATION

The DUDE system was one of four systems participating in the initial testing phase of the 2010 Spoken Dialogue Challenge. Each caller interacted with each system twice using different scenarios picked from a pool of sixteen. The scenarios consisted of a destination and an origin represented in terms of a map or location name and a time represented as an exact time, a time period or "now". In some scenarios the bus name was also given (e.g. 61C). For the DUDE system described above, 62% of calls reached the stage of presenting results to the user. Further analysis will be required to discover how many of the failed calls were due to telephony or platform errors (unfortunately we experienced such dropped calls quite often). Of these calls, 61% gave fully correct information to the users, and 74% were correct with respect to the route information (though the presented times were not always correct). These results are promising for a system which was developed so rapidly, using general tools. The resulting system could then be further improved in student project work.

6. SUMMARY AND FUTURE WORK

This paper describes a teaching platform for rapidly creating ISU Spoken Dialogue Systems using large databases. Its successful application was illustrated with the bus scheduling system for the 2010 Spoken Dialogue Challenge that was created and deployed to a VoiceXML platform and called successfully by many users. This system was created in less than 3 weeks using DUDE, which is certainly an appropriate timeframe for project work in class, or MSc projects or for initial stages of Ph.D. work. Future work includes enabling DUDE to handle multi-table databases, which would allow more efficient use of large databases.

7. ACKNOWLEDGEMENT

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement number 216594 (CLASSiC project project: www.classic-project.org) and by a Scottish Enterprise Proof of Concept Grant (project number 8-ELM-004). DUDE technology is the property of the University of Edinburgh and was used under licence from Edinburgh.

8. REFERENCES

[1] H. Hastie, X. Liu, and O. Lemon, "Automatic Generation of Information State Update Dialogue Systems that Dynamically Create VoiceXML, as Demonstrated on the iPhone," in *Proceedings of SIGDIAL (demonstrations)*, 2009.

[2] O. Lemon, "Context-sensitive speech recognition in Information-State Update dialogue systems: results for the grammar switching approach," in *Proceedings of the 8th Workshop on the Semantics and Pragmatics of Dialogue, CATALOG'04*, 2004.

[3] B.A. Hockey and G. Christian, "Zero to spoken dialogue system in one quarter: teaching computational linguistics to linguists using Regulus," in *TeachCL '08: Proceedings of the Third Workshop on Issues in Teaching Computational Linguistics*, Morristown, NJ, USA, 2008, pp. 80–86, Association for Computational Linguistics.

[4] D. Bobbert and M. Wolska, "Dialogue OS: an extensible platform for teaching spoken dialogue systems," in *Proceedings of the 11th Workshop on the Semantics and Pragmatics of Dialogue*, 2007.

[5] R. Cole, "Tools for research and education in speech science," in *Proceedings of the International Conference of Phonetic Sciences*, 1999.

[6] A. Raux, B. Langner, A. Black, and M. Eskenazi, "Let's go public! taking a spoken dialog system to the real world," in *Proceedings of Eurospeech*, 2005.

[7] O. Lemon, K. Georgila, J. Henderson, and M. Stuttle, "An ISU dialogue system exhibiting reinforcement learning of dialogue policies: generic slot-filling in the TALK in-car system," in *Proceedings of EACL*, 2006, pp. 119–122.