A Multi-domain Dialog System to integrate heterogeneous Spoken Dialog Systems

Joaquin Planells, Lluís-F. Hurtado, Encarna Segarra, Emilio Sanchis

Departament de Sistemes Informàtics i Computació. Universitat Politècnica de València
Camí de Vera s/n, 46022, València, Spain.
{xplanells, lhurtado, esegarra, esanchis}@dsic.upv.es

Abstract

In this paper, we present an architecture to create a multi-domain spoken dialog system with minimum effort by composing heterogeneous pre-existent spoken dialog systems into a new system able to perform richer interactions. A Task Manager acts as a proxy for the different sub-domains and activates one of the systems each turn. The different sub-systems are not aware that they are used in a multi-domain scenario and believe that they are speaking directly to the user. This allows us to add new domains leaving most of the underlying models unmodified and reducing the amount of time and money needed to deploy the application.

The proposed architecture has been applied to create a multi-domain system combining three heterogeneous spoken dialog systems (sport facilities booking, weather service and personal calendar) in Spanish. The evaluation with naive real users shows that this is an appropriate approach to develop multi-domain spoken dialog systems.

Index Terms: multi-domain dialog systems, spoken dialog systems, dialog management, stochastic finite-state transducers

1. Introduction

A spoken dialog system (SDS) can be seen as a human-machine interface that recognizes and understands the speech input and generates a spoken answer in successive turns in order to achieve a goal, such as obtaining information or carrying out an action. Voice-driven applications such as in-car navigation systems or telephone information services are common examples of spoken dialog systems. Most of the dialog systems are oriented to restricted domain tasks, mixed initiative, and telephone access although several new applications have appeared on portable devices like mobile phones or tablets. The new applications like Apple's Siri or Google Now tend to be multi-domain, user-driven spoken systems where the system answers are based on the content of the last utterance.

Different modules take part in order to carry out the final goal of a SDS: the Speech Recognition/Understanding Module, the Dialog Manager, the Answer Generator, and the Text-to-Speech Synthesizer. Each one has its own characteristics and the selection of the most convenient model varies depending on certain factors: the goal of each module, the possibility of manually defining the behavior of the module, or the capability of automatically obtaining models from training samples. Interest in the use of statistical techniques for the development of the different modules that compose the SDS has been growing over the last few years. These methodologies have been traditionally applied within the fields of Automatic Speech Recognition and Natural Language Understanding [1, 2, 3, 4, 5].

The Dialog Manager (DM) selects the best action at each turn based on the user utterance and the dialog history. Therefore, a dialog can be seen as a multi-stage decision problem. The application of statistical methodologies to model the behavior of the DM has provided compelling results in more recent years [6, 7, 8, 9, 10]. Recently, an approach based on Stochastic Finite-State Transducers (SFST) [11, 12] has been proposed. In this approach, given a system state and a user turn, a system action is selected and a transition to a new state is done. Therefore, dialog management is based on the modelization of the sequences of system actions and user dialog turn pairs. Then, a dialog describes a path in the transducer model from its initial state to a final one.

In a DM based on state models [8, 12], the number of states grows exponentially as attributes or concepts are added to the task. For example, if there are four attributes and the DM just stores whether or not it knows their values there are 2^4 different states. The number of states grows exponentially on the number of attributes and, in some approaches, the growth factor can be larger than 2. The more states a system has, the more expensive it is to estimate a model for it, and therefore, more training samples are needed for this estimation. Other approaches like POMDPs also suffer from the curse of dimensionality, and several approaches have been presented to make these models more tractable [13, 14].

Recently, there has been an increasing interest in multi-domain SDS. The goal is to be able to help the user in different domains using the same interface. Some techniques for multi-domain semantic understanding have recently been presented [15, 16]. In the case of dialog management, the exponential increase of dialog states makes it really difficult to create a DM that can serve several domains using the approaches that are used for single-domain systems. However, such systems can usually be easily divided, and it is possible to estimate separate models for each sub-domain individually. Furthermore, some attributes and concepts are common to several subtasks like *dates* and *times*, while others can be specific. One approach for building multi-domain SDS is to use a module that parses the user turn and redirects it to the appropriate single-domain DM [17, 18, 19]. The different modules or models share certain information about the dialog.

In this paper, we propose a multi-domain dialog system architecture that allows a designer to combine fully-functional SDS to create a richer system, without modifying the systems that are already created. Unlike other multi-domain architectures, this is done by taking the domain-specific SDS as black-boxes and communicating with them using a fake user. This architecture also allows us, among other things, to add a new domain to a multi-domain system by simply combining a dia-

log system for the new domain with the complete system. This is particularly useful because usually the training of models for SDS needs the interaction with hundreds of (paid) real users [20, 21], and it is desirable to reuse previous work.

The next section presents the modules used in the architecture and explains the job of each one. In Section 3, a description of the sport facilities booking, weather service, and personal calendar tasks is presented. In Section 4, an evaluation with real users is presented. In the last section, we present our conclusions and outline future work on this topic.

2. The Multi-domain Architecture

The multi-domain spoken dialog system architecture presented in this paper can be divided into several modules, as Figure 1 shows. It uses other SDS as sub-modules to create a system that is capable of richer interactions with the user. The multi-domain system acts as a proxy for the different single-task sub-systems and, at each turn, selects which one interacts with the user.

The modules that make up this multi-domain system are:

- Speech Recognizer (ASR): Generic automatic speech recognition module that transcribes the user utterance.
- Task Manager: Selects, at each turn, which sub-system is used.
- Context Register: Stores information about the dialog adapted to each sub-system.
- Single-task Sub-systems: Fully functional SDS for each domain.
- Multi-modal Answer Generator: Converts the system action to text and/or visual information.
- Synthesizer (TTS): Converts the system utterance into voice.

The vocabulary of the ASR has to cover all the vocabularies of the different sub-tasks that are included. Each of the sub-systems is a fully functional SDS with the modules described in the introduction. For simplicity, we used sub-systems that operate with text as input and output. In other words, ASR, TTS and Multi-modal Answer Generator are only used by the general multi-domain system. All the sub-systems are frozen during the dialog and at most one is awakened at each turn to interact with the user. No sub-system is aware that it is being used in a bigger system.

The Task Manager uses the word-level features and/or the content of the Context Register to classify the turn among all the possible sub-tasks. Each turn, it selects which task (if any) the user is interested in and awakens the sub-system associated to this task. The possible actions that the Task Manager can perform are simple: either awaken one sub-system or ask for task disambiguation if it is unsure which task the user is interested in.

If the awakened sub-system is the same as in the previous system turn, it receives the user utterance in text form like any standard SDS. However, if the sub-system has been frozen for at least one turn, it may have lost some important information supplied during the dialog while it was not active. The Context Register monitorizes both the user input and the system output to catch data that can be used by more than one task such as *times*, *dates* or *places*. This information is stored in a personalized buffer for each sub-system and injected into the user turn when the sub-system is finally awakened.

This architecture provides a way to combine existing SDS into a multi-domain system without having to modify the underlying models of each task. The effort needed is very small compared with the amount of work needed to build a multi-domain system from scratch and learn its models. When a new SDS sub-task is added, the designer should update the Context Register so that it can extract information related to the new task; for example, writing regular expressions to extract dates or adding a table of sport names. Not doing so prevents the sub-system from catching information from the dialog when it is frozen, but it does not invalidate the model. It just forces the user to repeat information more often. Information about the new task needs to be provided to update the Task Manager model.

2.1. Task Manager

When the user starts a dialog, the system starts one dialog with each sub-system and requests a turn from each of them (this is the welcome or *opening* turn of each task). These turns are ignored and a generic *opening* utterance is used instead. Then the system waits for the user to speak.

The Task Manager uses a Logistic Regression (LR) model [22] to classify the current user turn into a sub-task. The features used for the vectorization of the sentence are: bag of words, bag of bigrams, and co-ocurrence of two words in the same sentence. A one-vs-all classifier was created for each sub-task that assigned a score to the input sentence. The total number of different features was about 10^5, but, unfortunately, only very few samples were available for training. A first pass of the LR training algorithm was performed and then the features that had an absolute value weight below a certain threshold (experimentally defined) were discarded, leaving a final set of 2043 features, which was used to train the final LR model. We trained the model using a total of 2000 sample sentences extracted from the dialogs and the training samples of the semantic parsers of each task. The accuracy of the task classifier was 90.2% on a test set of 200 sentences. The errors encountered most often corresponded to turns where there was no proper task information or to turns that were valid for more than one task (for example, *yes* or *on Monday*).

The task with highest confidence is usually selected; however, if the confidence of the second-ranked task is close, the Task Manager outputs a disambiguation turn asking for the user's goal. The Task Manager usually makes the user add some task-related expressions like "book" or "weather" that can be used by the task classifier to make more accurate predictions. We try to mitigate this problem by providing more context to the utterance, and by extracting features from the current user turn and the previous user turn. In practice, we have found that this is the best strategy. Using more previous turns makes it more difficult to switch among tasks.

When one sub-system has finished a dialog by generating a closing turn, a fake "good bye" turn is sent to every sub-task to force the end of all the sub-dialogs.

2.2. Context Register

In the Context Register, we have a set of attribute-value pairs associated to each task. Both user and system turns are parsed to collect information from them. We use the semantic parser of each sub-system to perform the collection although a generic collector may also be used. The collector of each task extracts information from the turns that might be relevant to the task, for example *dates*, *times* or *place names*.

Later, when a sub-system is awakened, if there is some in-

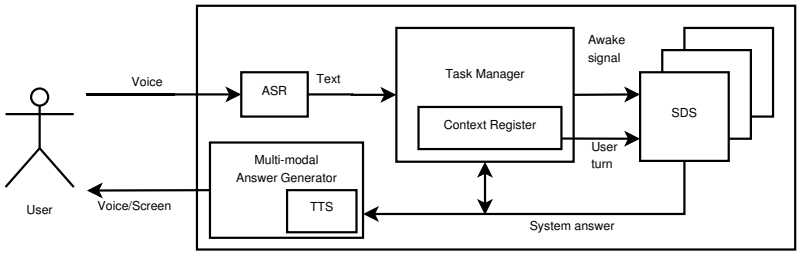

Figure 1: Architecture diagram

formation for it waiting for delivery, it is added as part of the user turn creating a fake turn. For example, if the user utterance is "*I want to book a basketball court*" and the content of the Context Register is *date=monday*, when the booking subsystem is awakened, it will receive the fake user turn "*I want to book a basketball court. The date is Monday*". Providing the context information, that is injected in the user turn, to the sub-systems allows us to treat these sub-systems as black boxes. Note that the semantic parsers of the sub-tasks are not trained specifically with these injected sentences, so the format of the fake turns needs to be carefully chosen to prevent misunderstandings or the introduction of errors. Testing different formats for injections will be needed to improve the performance of the system.

3. Sub-tasks

The sub-tasks included in the multi-domain SDS are presented in this section. The language for all the tasks is Spanish.

3.1. The SPORT task

The *sport* booking system can be used to make or cancel reservations for the sport facilities of our university. The SDS used for this task is the EDECAN-SPORT system presented in [11] and [12]. It is a mixed-initiative system which uses a Stochastic Finite-State Transducer approach for the dialog management. The Dialog Manager model was learned from dialogs that were created automatically by our dialog generator.

The semantic representation of the user turns is a frame structure that includes the different functionalities required for the task: a set of 4 task-dependent concepts representing user intentions (*Booking*, *Cancellation*, *Availability*, *Booked*), and 4 task-independent concepts (*Acceptance*, *Rejection*, *Not-Understood*, *Bye*). Up to 6 attributes can be attached to each concept (*Sport*, *Hour*, *Date*, *Court-Type*, *Court-Number*, *Order-Number*).

Dialog Manager answers are represented using a set of 21 actions. There are actions for opening and closing the dialog, confirming user supplied attributes, asking for more information, or showing information to the user. It also attaches a piece of HTML to some of the answers which is displayed on the screen.

In the following example, you can see a user turn, the system answer, and their corresponding semantic representations:

User: I want to book a basketball court for tomorrow.
`Booking`
`Sport:basketball Date:tomorrow`
System: You can see the available courts on the screen.
`Show-availability`

3.2. The WEATHER task

This system is a rule-based user-initiative SDS that answers questions about the weather forecast. Users can ask for the forecast for any day or ask for certain information (like temperature or wind strength). The semantics of this task is similar to the CUED dialog acts [23]. Users can ask for information (*request*) or confirm if something is going to happen (*confirm*). There are 10 attributes that can be used which include: date, time, weather state, wind speed, temperature, etc.

The following example shows two user turns, the system answers, and their corresponding semantic representations:

User: Tell me the temperature for tomorrow morning.
`request(temperature, date=tomorrow, time=morning)`
System: The temperature for tomorrow morning is expected to be high. Around 28 degrees.
`inform(temperature=28, date=tomorrow, time=morning)`
User: Is it going to be sunny tomorrow?
`confirm(state=sunny, date=tomorrow)`
System: Yes, tomorrow is expected to be sunny.
`affirm(state=sunny, date=tomorrow)`

The user has to provide all the information of the query in the same turn. The semantic decoder parses the sentence into a dialog act and then the Dialog Manager asks a database for the information and returns the answer either by providing information or by confirming whether or not the user-provided information is correct according to the forecast.

3.3. The CALENDAR task

This service allows the user to create or delete events and also search by name. This SDS is implemented as a rule-based system using a frame-based semantics. The Dialog Manager tries to fill all the values of the event in a fixed order: *title*, *date*, *time-begin*, *time-end*, etc.

The understanding module for this task uses pattern matching with regular expressions to find slot values in the sentence. The only exception is the *title*, which is an open-vocabulary slot and needs to be marked by the expression "*the title is*". Here you can see an example with a user turn, the system answer, and their corresponding semantic representations:

User: Show me my calendar for tomorrow.
`show-events date=tomorrow`
System: Here are your appointments for tomorrow.
`show-events date=tomorrow`

Note that the same semantics can encode different things depending on whether it represents a user turn or a system turn.

Figure 2 shows an example of dialog where the user exploits the three sub-systems. Figures 2b, 2c, and 2d show the same dialog but as seen by each sub-system. The turns where the Context Register has injected information into the user turn are labeled as *Fake user*. Sometime turns are requested from the sub-systems but are *ignored* and the result is never shown to the user. In the example, the user turn $U2$ causes a fake user turn for the WEATHER system ($U0$), and the user turn $U4$ causes a fake user turn for the CALENDAR system ($U0$).

4. Evaluation

In order to assess the quality of the multi-domain SDS, we defined 16 scenarios and asked users to interact with the system. Seven of them were single-task scenarios that cover some of the most common use cases of the pre-existent systems. Seven scenarios involved the use of two tasks in the same dialog, and the remaining two scenarios used all three tasks.

For the evaluation with real users, two user interfaces were developed. The first one is a web-based interface where users can write the utterance or use the WebKit speech recognizer integrated in Chrome. The other implementation is an Android application which uses the standard speech recognizer and the text-to-speech API. Both versions share the same underlying code for dialog management. At the end of the dialog, the user can label the dialog as correct if the goal was accomplished or wrong if some error occurred.

A total of 28 naive users were recruited. They performed 7.4 different scenarios on average, resulting in a set of 207 dialogs. Table 1 summarizes the results according to user evaluation. The success ratio decreases for the scenarios involving more than one task. Fifty dialog logs were manually reviewed, and we found that the accuracy of the task identification was 91.7%, which is similar to the accuracy achieved during training. Most of the errors corresponded to user turns that were valid for more than one tasks.

Distinct users	28
Total dialogs	207
Correct dialogs with 1 task	94.3%
Correct dialogs with 2 tasks	85.1%
Correct dialogs with 3 tasks	76.0%
Overall correct dialogs	84.5%

Table 1: Evaluation results.

5. Conclusions and Future Work

The architecture presented in this paper allowed the composition of three heterogeneous SDS with different semantic parsers and dialog management models into a multi-domain system. The sub-systems are not aware that they are collaborating in a multi-domain scenario allowing us to reuse any existing SDS. The effort needed is very small compared with the amount of work needed to build a multi-domain system from scratch and learn its models. A more accurate evaluation of the system should be done to include not only the user evaluation but also some objective measures.

As future work, we are interested in the use of this architecture to differentiate between different types of users. The system may have two different sub-systems for the same task: one system-oriented for naive users and another, with mixed-initiative, for expert.

S0: Hi! How may I help you?
U0: I want to book a tennis court.
S1: On the screen you can see the available courts. (A)
U1: Next Sunday from ten to eleven in the morning.
S2: Do you want to book the selected tennis court? (B)
U2: Is it going to rain?
S3: No, next Sunday is expected to be sunny. (C)
U3: Ok, book this court.
S4: Tennis court booked. Do you need anything else? (D)
U4: Yes, create an appointment on my calendar.
S5: What is the title for the event? (E)
U5: The title is tennis match.
S6: Create an event with the title "Tennis match"? (F)
U6: Yes.
S7: Event created. Do you need anything else? (G)
U7: No, thank you. Good bye.
S8: Thank you for using this system.

(a) Original dialog

S0: How may i help you? (ignored)
U0: I want to book a tennis court.
S1: On the screen you can see the available courts. (A)
U1: Next Sunday from ten to eleven in the morning.
S2: Do you want to book the selected tennis court? (B)
U2: Ok, book this court.
S3: Tennis court booked. Do you need anything else? (D)
U3: *Good bye*
S4: Good bye. Thank you for using this system. (ignored)

(b) Dialog as seen by the SPORT sub-system.

S0: How may I help you? (ignored)
U0: Yes, create an appointment on my calendar. *Date is next Sunday. Time is from ten to eleven.*
S1: What is the title for the event? (E)
U1: The title is tennis match.
S2: Create an event with the title "Tennis match"? (F)
U2: Yes.
S3: Event created. Do you need anything else? (G)
U3: No, thank you. Good bye.
S4: Thank you for using this system. (ignored)

(c) Dialog as seen by the CALENDAR sub-system.

S0: How may I help you? (ignored)
U0: Is it going to rain? *Date is next Sunday. Time is morning.*
S1: No, next Sunday is expected to be sunny. (C)

(d) Dialog as seen by the WEATHER sub-system.

Figure 2: Example dialog that activates the three tasks. All utterances have been translated from Spanish. SubFigure 2a shows the original dialog with a real user. Subfigures 2b, 2c, and 2d show the same dialog from the perspective of each sub-system. Turns where sub-systems are being used are labelled with the letters A-F. The text in *italics* has been injected by the Context Register.

6. Acknowledgements

This work is partially supported by the Spanish MICINN under contract TIN2011-28169-C05-01.

7. References

[1] E. Levin and R. Pieraccini, "Concept-Based Spontaneous Speech Understanding System," in *Proc. 4th Eurospeech'95)*, 1995, pp. 555–558.

[2] W. Minker, "Stocastically-based semantic analysis," in *Kluwer Academic Publishers*, Boston, USA, 1999.

[3] E. Segarra, E. Sanchis, M. Galiano, F. García, and L. Hurtado, "Extracting Semantic Information Through Automatic Learning Techniques," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 16, no. 3, pp. 301–307, 2002.

[4] Y. He and S. Young, "A data-driven spoken language understanding system," in *Proc. of ASRU'03*, 2003, pp. 583–588.

[5] Y. Esteve, C. Raymond, F. Bechet, and R. D. Mori, "Conceptual Decoding for Spoken Dialog systems," in *Proc. of EuroSpeech'03*, 2003, pp. 617–620.

[6] F. Jurcicek, B. Thomson, S. Keizer, F. Mairesse, M. Gasic, K. Yu, and S. Young, "Natural belief-critic: A reinforcement algorithm for parameter estimation in statistical spoken dialogue systems," in *Proc. of InterSpeech'10*, Makuhari, Japan, 2010, pp. 90–93.

[7] D. Griol, L. F. Hurtado, E. Segarra, and E. Sanchis, "A statistical approach to spoken dialog systems design and evaluation," *Speech Communication*, vol. 50, no. 7-9, pp. 666–682, 2008.

[8] J. Williams and S. Young, "Partially Observable Markov Decision Processes for Spoken Dialog Systems," in *Computer Speech and Language 21(2)*, 2007, pp. 393–422.

[9] O. Lemon, K. Georgila, and J. Henderson, "Evaluating Effectiveness and Portability of Reinforcement Learned Dialogue Strategies with real users: the TALK TownInfo Evaluation," in *Proc. of SLT'06*, Aruba, 2006.

[10] M. Gasic, F. Jurcicek, B. Thomson, K. Yu, and S. Young, "On-line policy optimisation of spoken dialogue systems via live interaction with human subjects," in *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*. IEEE, 2011, pp. 312–317.

[11] L.-F. Hurtado, J. Planells, E. Segarra, E. Sanchis, and D. Griol, "A stochastic finite-state transducer approach to spoken dialog management," in *Proc. of InterSpeech'10*, Makuhari, Japan, 2010, pp. 3002–3005.

[12] J. Planells, L.-F. Hurtado, E. Sanchis, and E. Segarra, "An online generated transducer to increase dialog manager coverage," in *Proc. of InterSpeech'12*, Portland , USA, 2012.

[13] M. Gašic, S. Keizer, F. Mairesse, J. Schatzmann, B. Thomson, K. Yu, and S. Young, "Training and evaluation of the his-pomdp dialogue system in noise," *Proc. Ninth SIGdial, Columbus, OH*, 2008.

[14] B. Thomson and S. Young, "Bayesian update of dialogue state: A pomdp framework for spoken dialogue systems," *Computer Speech & Language*, vol. 24, no. 4, pp. 562–588, 2010.

[15] A. Celikyilmaz, D. Hakkani-Tür, and G. Tur, "Multi-domain spoken language understanding with approximate inference," in *Proceedings of Interspeech*, 2011.

[16] M. Jeong and G. G. Lee, "Multi-domain spoken language understanding with transfer learning," *Speech Communication*, vol. 51, no. 5, pp. 412–424, 2009.

[17] K. Komatani, N. Kanda, M. Nakano, K. Nakadai, H. Tsujino, T. Ogata, and H. G. Okuno, "Multi-domain spoken dialogue system with extensibility and robustness against speech recognition errors," in *Proc. 7th SIGdial Workshop on Discourse and Dialogue*, 2006, pp. 9–17.

[18] B.-s. Lin, H.-m. Wang, and L.-s. Lee, "A distributed architecture for cooperative spoken dialogue agents with coherent dialogue state and history," in *Asru*, vol. 99. Citeseer, 1999, p. 4.

[19] O. Lemon, A. Gruenstein, and S. Peters, "Collaborative activities and multi-tasking in dialogue systems: Towards natural dialogue with robots," *TAL. Traitement automatique des langues*, vol. 43, no. 2, pp. 131–154, 2002.

[20] F. Jurcicek, S. Keizer, M. Gasic, F. Mairesse, B. Thomson, K. Yu, and S. Young, "Real user evaluation of spoken dialogue systems using amazon mechanical turk," *Proc, Interspeech2011, Florence*, 2011.

[21] M. Rayner, I. Frank, C. Chua, N. Tsourakis, and P. Bouillon, "For a fistful of dollars: Using crowd-sourcing to evaluate a spoken language call application," in *Proc. of Speech and Language Technology in Education Workshop (SLaTE)*, 2011.

[22] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: A library for large linear classification," *The Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.

[23] S. J. Young, "Cued standard dialogue acts," Cambridge University Engineering Department, Tech. Rep., 2007.