Building the next generation of personal digital assistants

Pierrick Milhorat, Stephan Schlögl, Gérard Chollet, Jérôme Boudy, Anna Esposito, Gianni Pelosi

BUILDING THE NEXT GENERATION OF PERSONAL DIGITAL ASSISTANTS

P. Milhorat, S. Schlögl*, G. Chollet*, J. Boudy†*

Institut Mines-Télécom
*Télécom ParisTech & †Télécom SudParis
Paris, France

A. Esposito

2nd Universit di Napoli
Department of Psychology
Caserta, Italy

G. Pelosi

Integrationi & Sistemi
Via dei Granai di Nerva
Rome, Italy

ABSTRACT

Voice-based digital Assistants such as Apple's Siri and Google's Now are currently booming. Yet, despite their promise of being context-aware and adapted to a user's preferences and very distinct needs, truly personal assistants are still missing. In this paper we highlight some of the challenges in building personalized speech-operated assistive technology and propose a number of research and development directions we have undertaken in order to solve them. In particular we focus on natural language understanding and dialog management aspects as we believe that these parts of the technology pipeline require the biggest amount of augmentation.

Index Terms— Personal Digital Assistants, Spoken Dialog Systems, Assistive Technology, Language Understanding, Dialog Design

1. INTRODUCTION

Recent years have seen a significant increase in the deployment of voice-controlled Personal Digital Assistants. During the last two decades the necessary technology integration (i.e. the combination of Automatic Speech Recognition, Natural Language Understanding, Dialog Management and Text-to-Speech Synthesis) has been the focus of extensive academic and industrial research, eventually resulting in commercial products such as Apple's Siri[1], Google's Now[2] and Nuance's Nina[3]. These products not only show the advancements of recent technologies but also manage to bring the concept of an artificial personal assistant, i.e. a system that is able to (at least to some extend) understand and respond to spoken inputs, to a somewhat broader end-user level. Continuous progress happened from a technological perspective as well as with respect to supported application domains. While the first systems that used natural language as an interaction modality were predominantly focusing on the travel domain, possible application scenarios have progressively been extended and now include areas such as weather forecast [1], navigation [2], translation [3], infotainment [4], tutoring [5] and even health care [6]. Despite the overall advancements the use of natural language driven assistive technologies is, however, still cautious. One reason for resistance may be found in the fact that Siri & Co., for all their efforts, are not really personal. More focus on personal adaptation is necessary so as to better integrate systems into users' daily routines. While the technology is at a point where it may be called robust, the integration of context and personalized behavior is still at its starting point. Significant improvements are needed in order to move from pure digital assistants to truly personal ones.

The research presented in this paper is conducted as part of the vAssist project (AAL-2010-3-106), which is partially funded by the European Ambient Assisted Living Joint Programme and the National Funding Agencies from Austria, France and Italy.

[1] http://www.apple.com/ios/siri/

[2] http://www.google.com/landing/now/

[3] http://www.nuance.com/landing-pages/products/nina/default.asp

2. PERSONAL DIGITAL ASSISTANTS

Our common understanding of a personal assistant is that of a person (or an agent) who is able to provide distinct help at a given time and in a given activity context. For example, a secretary situated in a general work context provides support for activities such as answering incoming calls, recording meetings and appointments, ordering products, or interacting with clients. An important characteristic of personal assistants is that they adapt to the distinct demands of their 'master' and furthermore (over time) progressively pay attention to her/his personal preferences and routines. Also, as by their definition, personal assistants should each be helping only one person, making this one-to-one relationship between assistant and 'master' a crucial benefit. Given this definition of a personal assistant it is easy to argue that many people would find it convenient to have such a person at their disposal, even though not all of them may have a clear understanding of their exact context of use. Hence, when we try to approach this subject from a more technical perspective, we already find a set of requirements and expectations coming from users. Among the most requested features when thinking of digital assistants, are simplicity, flexibility and easiness of interaction. Voice-based input/output interfaces may be the easiest way to fulfill these requirements. This is be-

cause voice-based interaction is usually simple, flexible and does not require cognitive efforts, attention and/or memory resources on the side of the user. Voice interfaces for example, can flatten option menus and supply rapidly complex verbal responses.

How such a voice-based Personal Digital Assistant could work was first exemplified in 1995 by the futuristic APPLE Knowledge Navigator concept [4]. The video emphasizes the effectiveness and efficacy of having perfect Voice User Interfaces. Watching the clip, any expert in speech processing, synthesis and recognition can clearly see why the proposed application is rather futuristic. The dialog among the digital assistant and the user is free of any constraints. The assistant knows the psychological motivations of the user, it is able to perceive the user's emotions and even able to suggest actions that include the user's social relationships (e.g. recommending a friend's recently published research paper). In addition the assistant speaks perfectly fluent English, employing appropriate intonation and prosodic cues that transmit paralinguistic information, such as for example concealed disapproval of how the user is behaving with his mother.

However, from a technical perspective VUIs represent a complex interface option. In particular, since currently the capabilities of the underlying technologies for Automatic Speech Recognition (ASR), Natural Language Understanding (NLU) and Text-to-Speech (TTS) synthesis are constrained by pre-defined application contexts and not yet ready for supporting a free-form human-machine conversation. This is why current voice enabled systems are developed based on specific use case descriptions. Scenarios range from rather simple applications such as using speech input to surf the Internet, to more complex settings where voice may be used to monitor the well being of elderly people, or to offer them assistance in operating technological services such as sending an SMS, writing an e-mail, searching their weekly agenda, or managing a calendar program. Systems targeted at elderly people face particular challenges, as due to possible age-related fine motor articulatory impairments additional efforts are required in order to adapt and improve the speech recognition algorithms [7].

In summary it can be said, that users' expectations when speaking to systems, exploiting their ordinary verbal interactions with other humans, are much higher than those they have when interacting with graphical user interfaces. Therefore it is necessary to develop VUIs that are able to satisfy these expectations. Given that the current technology is not able to provide algorithms to process and understand free-forms of conversations, the appropriate design of the dialog management, such that the user has the feeling of a naturalistic interaction, is crucial to ensure an effective and efficient use of the system. To our knowledge, there are no standards for the development of more 'satisfying' VUIs. Although there have been efforts in providing suggestions for potential solutions [8], this issue is still at a research stage. In the following we will describe our approach to improve the current state of the art. Based on the considerations discussed above, we particularly focus on natural language understanding and dialog management. In our opinion, these components require the biggest amount of improvement in order to render associated VUIs more naturalistic and satisfying for the end users.

[4]http://www.youtube.com/watch?v=JIE8xk6Rl1w

3. POTENTIAL AREAS OF IMPROVEMENT

Given the system requirements described above and the current state of the art we see four areas where significant improvements are possible:

3.1. Extended Dialog History

Deployed systems, whether they have a commercial or a research purpose, usually utilize the dialog history as an instrument to disambiguate user utterances and to keep track of the dialog state. Yet, the memorized activity log is often focused on a single dialog and does not spread across different conversations. Expanding the history element may allow for building systems that adapt to the user over time. Hence, systems should be able to adapt the speech segment's acoustic models, the language models, the understanding as well as the general way of completing a dialog. For example, the overall interaction could be improved if the system has information about the identity, the gender and the age of a user, and furthermore respects his/her preferences for interacting with technology.

3.2. Improved Context Awareness

In addition to what is directly requested by the user, a vast amount of information is usually available which may be processed by a system and eventually could improve its context awareness. For example, sensors embedded in (future) homes, in the office or simply in every day used appliances such as our beloved smartphones, are precious data sources that could be used to augment human-machine interaction. Also the Internet may be a potential data pool that could be mined in order to provide an enhanced knowledge of the "outside world" and consequently improve a system's reasoning.

3.3. Dynamic System Adaptation

The development and implementation of an SDS usually consists of three stages. The first stage is concerned with the specification of the relevant components, the second with the implementation and third with the actual deployment of the system. From the implementation stage onward, a system's configuration is often static, i.e. it does not change with usage nor according to a given dialog state. We therefore propose to use a multi-agent architecture so that the settings of one agent can be changed based on the input coming from other agents. So if one component detects a change in the dialog context it

can inform all the other components, updating their configuration. This dynamic mechanism would allow for modifying the system even while it is running.

3.4. Supported Task Hierarchy Design

Previous work has highlighted an overall classification for Dialog Management (DM) paradigms [9]. According to this classification a DM is based on either stochastic processes i.e. MDPs [10], POMDPs [11], the information state principles [12], or a hierarchy of tasks [13, 14]. Each of theses categories has advantages and drawbacks. The information state paradigm is considered as an inaccurate theoretical framework and therefore does not meet the requirements of most practical implementations. The same applies to stochastic processes whose main drawback is that training data has to be collected in order to build a system. Despite significant research efforts to overcome this issue (e.g. reinforcement learning [15]) these processes are yet to be streamlined. The task-based paradigm therefore remains the most appropriate technique. Depending on the size of a dialog, the initiative given to the user, and the conditional execution steps, task hierarchies may often be very complex. Better tools and methods should therefore be available so that the design of task hierarchies and their automatic transformations can be shifted to the specification stage of an application.

4. A PROPOSED ARCHITECTURE

In order to approach the above discussed areas of improvement, a Spoken Dialog System (SDS) has been implemented. Its architecture is shown in Fig. 1. The illustrated base components represent specialized agents which communicate with each other using the ActiveMQ messaging server[5]. The data flow between components is restricted so as to control for a near-sequential processing pipeline. The leftmost layer depicts the interface with the user. Currently this is achieved through a single sensor i.e. a microphone that captures the signal, which is subsequently analyzed by the ASR engine [16]. The ASR frames the signal, converts it to a sequence of finite observation vectors and matches those against a set of acoustic and linguistic stochastic models. It produces a ranked list of hypotheses for the text content recognized from the speech signal and associates them with confidence scores related to their acoustic likelihood and linguistic probability. Only transcriptions with scores passing a given threshold are propagated to the core system. In the case that no transcription passes the threshold a generic "not understood" message is passed on. On the other end of the pipeline the TTS engine [17] generates and plays audio responses from text sent by intermediate processing component.

[5]http://activemq.apache.org/

This primary system layer links the signal level with the textual level of the architecture. Additional sensors may be integrated in the future so as to increase the entropy of the captured information and consequently improve the system's overall performance. For the moment, however, the focus lies on improving the middle components dealing with language understanding, dialog management and text generation. Here the Natural Language Generator (NLG) is currently based on a fairly simple template selection process which randomly chooses text utterances according to the current semantic representation of the system. The Natural Language Understanding (NLU) component, however, is a complex association of several integrated components. Section 5 will described these components in more detail as well as the successive transformations that have to be applied in order to process the text transcriptions coming from the speech recognizer.

Finally, the center piece of the proposed architecture is the dialog manager [14]. It is based on a generic inference engine that has to be configured using scenario-based task hierarchies. It executes actions and generates semantic concepts related to user inputs. Also here the reader will find a more detailed description of the component, and the way it is set and used by the system, in Section 5.

5. COMPONENTS AND ALGORITHMS

In this section, we present the components and algorithms we propose in order to tackle the challenges described in Section 3. First, a discussion of the natural language understanding components is provided followed by a clarification of the types of problems we intend to solve with this setup. Then, the implementation of a dialog model design tool is detailed.

Overall the natural language understanding process aims at providing a mapping between the unrestricted infinite word-level semantic space a user may utilize when interacting with a system, and the dynamic set of parameterized dialog acts a system is capable of dealing with at any given point in a dialog. As such natural language understanding is a multi-step dynamic process with clearly separated roles taken on by individual sub-components. The input space size of any sub-component is greater or equal to the size of its output space. In other words, an NLU engine processes user utterances sequentially and, at every stage, the level of the semantics extracted from the text is more machine oriented. This sequential processing of a transcription until it reaches the dialog manager is illustrated in Fig. 2.

5.1. Parsing

A semantic parser associates semantic labels with a text utterances (or parts of it). The most commonly used parsing techniques are based on context-free grammars or probabilistic context-free grammars, which are either hand-coded based on the analysis of collected dialog data, or designed by ex-

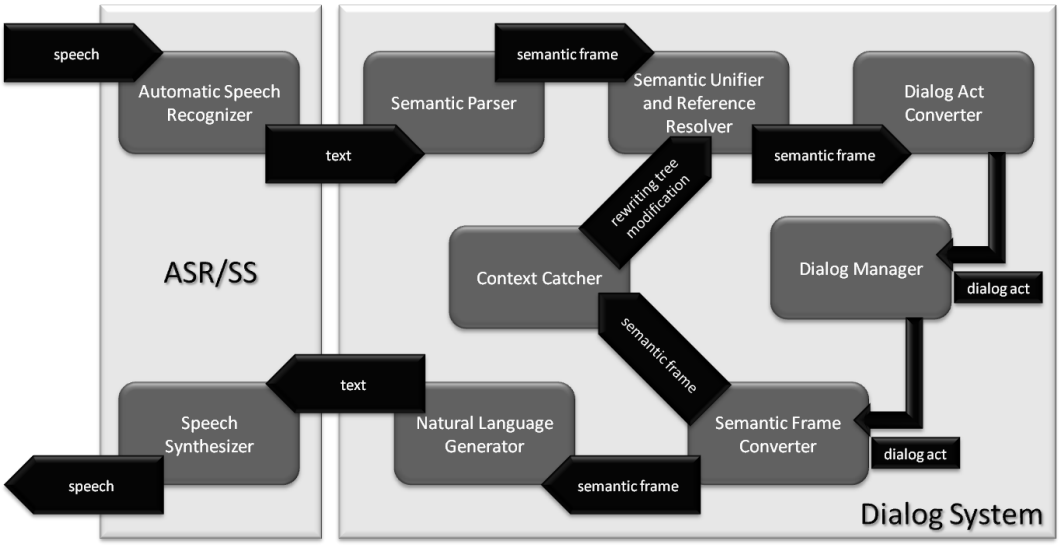

Fig. 1. The Proposed System Architecture

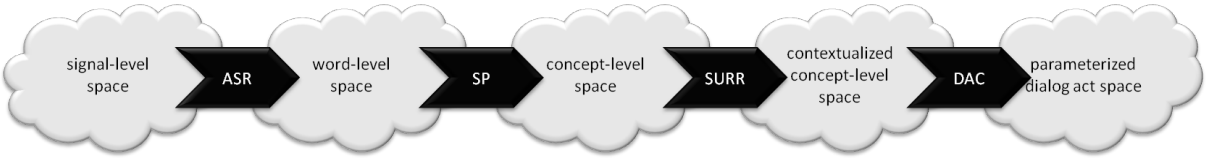

Fig. 2. The One-way Interpretation Process of Spoke User Utterances

perts. For our system we integrated the algorithm proposed by Jurčíček et al. [18]. Instead of matching complete-sentence patterns with text input, Jurčíček et al propose to look for patterns in chunks of the text-level sentence and in the temporary (i.e. already assigned) Semantic Frame (SF). An operation is applied to the current SF when the triggering pattern is detected. The rules consisting of a trigger part (the pattern to look for) and an operation part (the transformation to apply) are automatically learned from an annotated corpus of utterances associated with their final SFs. At every step of the training process, the algorithm tests all the [trigger:operation] pairs available. The one pair that gets the best score, i.e. the one whose application results in the largest Levenshtein distance reduction between the obtained SFs and the reference ones, is added to the decoding piles and applied to the temporary corpus.

5.2. Unification

As already mentioned, the parsing relies solely on rules derived from a priori offline training and a user's text input, without processing any contextual information. Thus parsing results may be shallow. For instance, a user utterance such as "two" would be labeled as "input:number=2", where input is the goal of the semantic frame and "2" is the value of the slot "number". The utterance itself does not bear more information, which shows the need for other sources of contextual information to augment the primary semantic representation. We have therefore defined a module for Semantic Unification and Reference Resolution (SURR). The SURR maintains a dynamic tree collection whose nodes are sets of slots (some of them valued). The edge between a parent node and one or more children node(s) bears a conditioned transformation that applies to the value or the name (or both) of the slots in the children nodes when going up the trees. In order for a semantic frame to pass the test of this filtering component, the algorithm has to find a path in the forest of nodes. The slot part of the input SF is split into subsets that are mapped onto the tree nodes. The algorithm succeeds if it finds a single path leading from the initial split to the root nodes.

Algorithm 1 illustrates the process, given the following set of slots in the input SF: $Slots = \{Slot_1 = Value_1, Slot_2 = Value_2, \ldots, Slot_n = Value_n\}$. The splittings of Set (line 4) are kept in memory so that the algorithm does not test the same partition of a set twice and exits the recursive function when all of them fail. The SURR produces a "no solution" message when it is unable to retrieve a path in the structure. This is to signal to the user that the last utterance was not understood by the system (i.e. not valid in the current configuration).

Algorithm 1 SURR algorithm

```
1: procedure FINDPATH(Slots)
2:     if Slots is made of root sets of slots then return Slots
3:     else
4:         split Slots into two parts, Split and Remaining
5:         if Split can be mapped to a node Node in the trees then
6:             if the transformation Transform from Node to its parent is valid then
7:                 apply Transform
8:                 merge the transformation of Split with Remaining to get a new set TransSlots
9:                 FINDPATH(Slots)
10:            else
11:                FINDPATH(Slots)
12:            end if
13:        else
14:            FINDPATH(Slots)
15:        end if
16:    end if
17: end procedure
```

A final piece of software, the dialog act converter, then provides a mapping between the contextualized meaning representation of the user intent and the set of dialog acts available at the current internal state of the dialog manager. For that, the stack of tasks maintained by the reasoning engine is mined to retrieve the expectations. A matching process is then triggered to convert the frame into a parameterized act. If no suitable matching is available, a "can not map" dialog act is created.

5.3. Data Integration

The SURR tree slots are defined as dynamic predicates, i.e. they do not remain static after being loaded. Some of them obtain a value by calling either a sensor reading or an external software service (local or remote). This is how we propose to resolve all references to the external environment, i.e. the context that is not internally maintained by the DM. For example, information such as the geographic location of the user, the weather, the date and time, or the day of the week. We propose to integrate external data streams as early as possible, supporting the NLU process with better contextual information. Following the integration of external sources of information, the internal context is also used to update the structure of the SURR. Status information is intercepted and processed by the Context Catcher (CC), a module which sends new predicate definitions and remove-commands to the SURR so that branches are adapted to the internal context of the system. Finally, in order to save an extended history across dialogs, it is planned to add an external profile file. Accessible by the SURR , this file will store user-dependent values, which were learned over time and proven to be relevant.

5.4. Dialog Design

As highlighted earlier in the paper, in order to make a system truly versatile and adaptive, the dialog models must be easily and quickly modifiable. However, a task hierarchy grows exponentially into an interleaved net of tasks which is hard to debug and/or to modify. To solve this issue, we propose a new XML-based design language with a set of automatic transformations to convert dialog designs into machine readable code. For example, one wants to add voice interaction to an already existing application. The specifications of the application, i.e. its input and output mechanisms, are known.

Using our proposed design language, we can describe the application in terms of connected forms with associated execution scripts. The XML code for such a description scheme is shown in Fig. 3. Recursive eXtensible Stylesheet Language Transformations are then applied to process the forms and automatically build a task-based dialog model including variable definitions, inputs and outputs, conditions as well as execution scripts.

6. CONCLUSION AND FUTURE WORK

The proposed procedures discussed in this paper aim at making a constrained human-machine dialogue more flexible and adaptable to the user's requirements, bypassing the limitations of the current technological capabilities. We discussed several challenges for building future SDS-driven Personal Digital Assistants, and described a number of components and algorithms with which we aim to tackle them. Our goal for the future is to further improve our solutions and make them available, so that also other systems may implement them. The challenge of meeting sophisticated real-world dialog requirements as well as the complexity of specialized

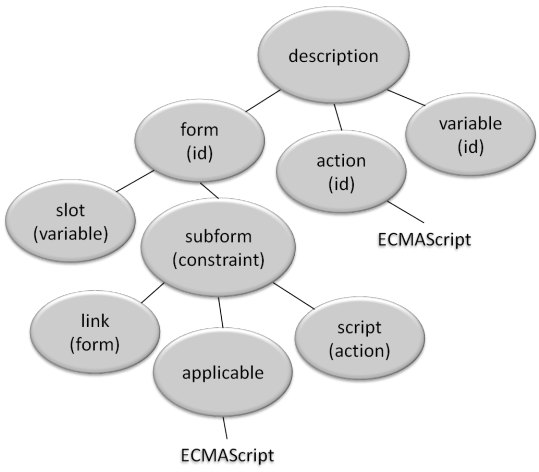

Fig. 3. Application Description Language Scheme

contextual instances suggest that, for some time, it will be improbable to develop any standard for designing effective and efficacy dialogue systems, even in environmental constrained applications. The ideas proposed in this paper should therefore be considered as an attempt to progress towards better, more flexible natural language user interfaces.

7. REFERENCES

[1] V. Zue, S. Seneff, J. R. Glass, J. Polifroni, C. Pao, T. J. Hazen, and L. Hetherington, "J UPITER : A Telephone-Based Conversational Interface for Weather Information," *IEEE Transactions on Speech and Audio Processing*, vol. 8, no. 1, pp. 85–96, 2000.

[2] R. Belvin, R. Burns, and C. Hein, "Development of the HRL route navigation dialogue system," in *Proceedings of ACL-HLT*, 2001, pp. 1–5.

[3] M. Kolss, D. Bernreuther, M. Paulik, S. Stücker, S. Vogel, and A. Waibel, "Open Domain Speech Recognition & Translation: Lectures and Speeches," in *Proceedings of ICASSP*, 2006.

[4] S. Möller, J. Krebber, A. Raake, P. Smeele, M. Rajman, M. Melichar, V. Pallotta, G. Tsakou, B. Kladis, A. Vovos, J. Hoonhout, D. Schuchardt, N. Fakotakis, T. Ganchev, and I. Potamitis, "INSPIRE: Evaluation of a Smart-Home System for Infotainment Management and Device Control," in *Proceedings of LREC*, 2004, pp. 1603–1606.

[5] I. Braun and N. Rummel, "Facilitating Learning From Computer-Supported Collaborative Inquiry: the Challenge of Directing Learners' Interactions To Useful Ends," *Research and Practice in Technology Enhanced Learning*, vol. 05, no. 03, pp. 205, 2010.

[6] D. Coyle, G. Doherty, M. Matthews, and J. Sharry, "Computers in talk-based mental health interventions," *Interacting with Computers*, vol. 19, no. 4, pp. 545–562, 2007.

[7] D. R. S. Caon, T. Simonnet, P. Sendorek, J. Boudy, and G. Chollet, "vAssist: The Virtual Interactive Assistant for Daily Homer-Care," in *Proceedings of pHealth*, 2011.

[8] B. Balentine and D. P. Morgan, *How to Build a Speech Recognition Application: Style Guide for Telephony Dialogues*, Enterprise Integration Group, 2001.

[9] R. Catizone, A. Setzer, and Y. Wilks, "State of the art in dialogue management," *Deliverable D5*, 2002.

[10] E. Levin, R. Pieraccini, and W. Eckert, "Using Markov decision process for learning dialogue strategies," *Proceedings of ICASSP*, 1998.

[11] J. Henderson and O. Lemon, "Mixture model POMDPs for efficient handling of uncertainty in dialogue management," *Proceedings ACL-HLT*, pp. 73–76, 2008.

[12] S. Larsson and D. Traum, "Information state and dialogue management in the TRINDI dialogue move engine toolkit," *Natural language engineering*, 2000.

[13] D. Bohus and A. I. Rudnicky, "RavenClaw: Dialog Management Using Hierarchical Task Decomposition and an Expectation Agenda," in *Proceedings of EUROSPEECH*, 2003.

[14] C. Rich and C. L. Sidner, "Collaborative discourse, engagement and always-on relational agents," in *Proceedings of AAAI*, 2010.

[15] 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.

[16] A. Lee, *The Julius book*, 2010.

[17] M. Schröder and J. Trouvain, "The German text-to-speech synthesis system MARY: A tool for research, development and teaching," *International Journal of Speech Technology*, 2003.

[18] F. Jurčíček, F. Mairesse, M. Gašić, S. Keizer, B. Thomson, K. Yu, and S. Young, "Transformation-based Learning for semantic parsing," in *Proceedings of INTERSPEECH*, 2009, pp. 2719–2722.