Multi-step Natural Language Understanding

Pierrick Milhorat, Stephan Schlögl, Gérard Chollet
Institut Mines-Télécom
Télécom ParisTech, CNRS LTCI
Paris, France
{lastname}@enst.fr

Jérôme Boudy
Institut Mines-Télécom
Télécom SudParis
Paris, France
boudy@telecom-sudparis.eu

Abstract

While natural language as an interaction modality is increasingly being accepted by users, remaining technological challenges still hinder its widespread employment. Tools that better support the design, development and improvement of these types of applications are required. This demo presents a prototyping framework for Spoken Dialog System (SDS) design which combines existing language technology components for Automatic Speech Recognition (ASR), Dialog Management (DM), and Text-to-Speech Synthesis (TTS) with a multi-step component for Natural Language Understanding (NLU).

1 Introduction

Recently speech and other types of natural language are experiencing an increased acceptance when being used for interacting with 'intelligent' computing systems. This trend is particularly reflected by products such as Apple's *Siri*[1], Google's *Now*[2] and Nuance's *Dragon Solutions*[3]. While these applications demonstrate the industry's vision of how we should be interacting with our current and future devices, they also highlight some of the great challenges that still exist. One of these challenges may be seen in the fact that Automatic Speech Recognition (ASR) remains a highly error-prone technology which influences subsequent natural language processing components such as Natural Language Understanding (NLU) and Dialog Management (DM) and leads to often unsatisfying user experiences. Hence we require appropriate tools that better support the testing and studying of language as an interaction modality and consequently allow us to build better, more user-centered applications.

[1]http://www.apple.com/ios/siri/
[2]http://www.google.com/landing/now/
[3]http://www.nuance.com/dragon/

This demo presents our approach of developing a prototyping tool for Spoken Dialog Systems (SDS). Our solution is particularly focusing on the natural language understanding aspect of SDS design. The overall framework is composed of a set of existing open-source technology components (i.e. ASR, DM, TTS) which are expanded by several additional NLP modules responsible for natural language understanding as well as generation. The following sections first provide a general overview of the entire framework and then focus particularly on the NLU part of our solution and the different sub-modules it integrates.

2 Spoken Dialog System Design

A state-of-the-art SDS usually consists of a set of technology components that are integrated to form a consecutive processing chain. Starting on the input side the ASR module produces a hypothesis about the orthographic content of a spoken utterance. The NLU takes this recognized utterance and converts it into a machine readable command or input Dialog Act (DA). The DM processes this input DA and sends the relevant output DA to the Natural Language Generation (NLG) component. The NLG is then responsible for converting the output DA into appropriate natural language text. Finally, the Text-to-Speech (TTS) synthesis component takes the text transmitted by the NLG and speaks it to a user.

According to this general architecture different open-source language components have been integrated to form a loosely coupled SDS framework. The framework includes ASR performed by the Julius Large Vocabulary Continuous Speech Recognition engine[4], dialog management based on the Disco DM library (Rich, 2009; Rich

[4]http://julius.sourceforge.jp/en_index.php

and Sidner, 2012) and TTS achieved through the MARY Text-to-Speech Synthesis Platform[5]. Additionally, we have integrated the WebWOZ Wizard of Oz Prototyping Platform[6] (Schlögl et al., 2010) in order to allow for the simulation of (flawless) natural language understanding. Expanding these existing components we have then developed as a set of modules responsible for actual system-based natural language processing. The following section describes these modules in more detail and highlights the types of challenges they try to overcome.

3 Natural Language Understanding

Within the processing chain of a spoken/text-based dialog system, the NLU component is the link between the wide and informal communication space of a user's input and the formal and rather restrictive semantic space that can be processed by the DM (Mori et al., 2007). Trying to bridge these two spaces we have connected several modules to form an NLU processing segment whose different modules are described below.

3.1 Semantic Parsing

First we use a Semantic Parsing (SP) module to convert the transcribed speech provided by the ASR into so-called Semantic Frames (SFs). To achieve this mapping Jurčíček et al. (2009) designed a Transformation-Based Learning Semantic Parser (Brill, 1995) which we adapted to integrate it with our framework. The algorithm applies an ordered set of rules to hypothetical [*utterance, SF*] pairs in order to find the closest matching SF.

3.2 Semantic Unification

Next we use what we call the Semantic Unifier and Reference Resolver (SURR) module to convert input SFs into SFs that can be processed by the DM input interface. To do this we implemented a bottom-up search algorithm for rewriting trees whose nodes contain lists of valued slots. The algorithm looks for a group of root nodes that can be reached in the forest (i.e. the existing number of trees) by transforming an input SF's set of slots according to the given rewriting rules. It succeeds when all slots can be rewritten into a root list of slots. This module is supported by external knowledge sources such as for example the context in which an utterance has been produced (i.e. it receives input from the Context Catcher module described below). Furthermore it could call operating system functions, sensor readings [7] or other knowledge sources capable of providing relevant data, in order to resolve and disambiguate input. For instance, special-valued slots like 'date=today' are dynamically resolved to the correct data type and value, making the NLU more sensitive to its surrounding environment.

3.3 Context Inclusion

In order to optimize information exchange Human-Human interactions usually build up a common knowledge between dialog participants. This inherent grounding process can be compared to the dialog history recorded in an SDS's DM. Using these recordings we have introduced a so-called Context Catcher (CC) module. The way this module is currently working is as follows: The DM requests information from the user to progress through the task-oriented dialog. The user replies without specifying the type of data he/she is providing, the overall intent of the utterance or the relation to any dialog slot. The CC evaluates the request expressed by the DM and consequently updates various parameters of the SURR component. Consequently the SURR is able to provide a better, more context-specific mapping between raw SFs provided by the SP module and the expected slots to be filled by the DM component.

3.4 Dialog Act Conversion

An SDS's DM expects formal meaning representations to be converted to actual dialog moves or Dialog Acts (DA); similar to parametrized dialog commands. A DA is the smallest unit of deterministic action to support the dialogue flow. The number of DAs that are available at any given point is finite, dynamic and depends on the current state of the dialog (Note: Here a state does not refer to a 'real' state, such as the ones used in Markov Decision Processes or Partially Observable Markov Decision Processes, but rather to a general status of the dialog). In other words, two input utterances carrying the same meaning may lead to different consequences depending on a given dialog state. The right action, i.e. the accurate DA, is to be determined by the NLU component. As there

[5]http://mary.dfki.de/

[6]https://github.com/stephanschloegl/WebWOZ

[7]Note: At the moment sensor readings are not implemented as they are currently not available in the developing environment

is usually a many-to-many matching between SFs and actual DAs we integrated an additional Dialog Act Converter (DAC) module. This module uses the context to generate a list of expected slots for which a user may provide a value (i.e. it converts possible DAs to SFs). Then a matching between the actual inputs and the expectations is applied in order to find the most probable DA.

4 Supporting Mixed Initiatives

SDS dialog designs usually run along an initiative scale that ranges from user-driven to strictly machine-driven interaction. In the case of a machine-driven dialog a user has to follow the requests of the system. Interactions that lie out of the scope of this dialog design are not understood and may either be discarded or, in the worst case, lead to a system failure. Despite this potential for failure, machine-driven designs make the dialog easier to control and thus less prone to errors, yet, due to the lack of adaptability exposed by the system, also less human-like. On the other hand, pure user-driven dialog designs minimize the functional range of a system as they only react to commands without assuring their functional integrity.

The above described modular approach to NLU aims to support a mixed initiative design where a system's integrity and its goals are sufficiently defined; the user, however, is not restricted by the type and amount of spoken input he/she can use to interact. To offer this type of interaction the system needs to handle three kinds of potential mis-usages: (1) out-of-application cases, (2) out-of-dialog cases and (3) out-of-turn cases. To address the first one our training corpus has been augmented so that it includes examples of garbage SFs. As a result an out-of-application utterance triggers a generic reply from the system, notifying the user that he/she is outside the scope of the application. In the case where a user stays within the scope of the application but tries to initiate a new unrelated dialog (i.e. out-of-dialog case), the DM's stack of tasks is incremented with the new dialog. The system will lead the user back to the previous topic once the newly added one is completed. Finally, as for the out-of-turn cases i.e. the cases where a user would answer a system request with a non-expected utterance such as an over-complete one, the NLU process, retrieving the DM's expectations, discards unrelated or over-complete information.

5 Demo Description

Focusing on the NLU aspect of the SDS pipeline this demo will demonstrate how the different modules described above (i.e. SP, SURR, CC, and DAC) work together. An application scenario from the ambient assisted living domain (i.e. the operation of a 'Pillbox' application) will serve as an example use case. It will be shown how the natural language input potentially recognized by an ASR component is further interpreted by our NLU processing segment. All the steps discussed in Section 3 will be visible.

6 Conclusion

In this paper we described a set of NLU components that were integrated as part of a loosely coupled SDS. Separate modules for semantic parsing, semantic unification and reference resolution, context inclusion as well as dialog act conversion have been described. Furthermore we have highlighted how our system offers support for mixed-initiative dialog interactions. A first test of this NLU processing chain showed that the use of our multi-component approach is feasible, and we believe that this solution can be seen as a valuable test and development framework for natural language processing research.

References

E. Brill. 1995. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational linguistics*.

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

R. De Mori, F. Béchet, D. Hakkani-Tur, M. McTear, G. Riccardi, and G. Tur. 2007. Spoken language understanding: A survey. *Proceedings of ASRU*.

C. Rich and C. L. Sidner. 2012. Using collaborative discourse theory to partially automate dialogue tree authoring. *Intelligent Virtual Agents*, pages 327–340.

C. Rich. 2009. Building task-based user interfaces with ANSI/CEA-2018. *Computer*.

S. Schlögl, G. Doherty, S. Luz, and N. Karamanis. 2010. WebWOZ: A Wizard of Oz Prototyping Framework. In *Proceedings of ACM EICS*, pages 109–114.