NAACL-HLT 2007

Bridging the Gap: Academic and Industrial Research in Dialog Technologies

Proceedings of the Workshop

April 26, 2007
Rochester, New York

Production and Manufacturing by
Omnipress Inc.
Post Office Box 7214
Madison, WI 53707-7214

Order copies of this and other ACL proceedings from:

Association for Computational Linguistics (ACL)
75 Paterson Street, Suite 9
New Brunswick, NJ 08901
USA
Tel: +1-732-342-9100
Fax: +1-732-342-9339
`acl@aclweb.org`

Preface

In the recent years, we have seen rapid adoption of dialog systems in commercial applications. They range from telephone-based services, in-car interactive systems, to online conversational service agents and talking characters in computer games. Open-standard platforms such as VoiceXML have been adopted by the industry, and become the driving force for the faster adoption of dialog applications.

The widespread dialog applications in industry setting pose challenge for researchers in both industrial and academic worlds. Progress from academic world has not benefited the real world applications to a satisfactory extent. The purpose of this one-day workshop is to provide a forum to bring industrial and academic researchers together to share their experiences and visions in the dialog technology development, and to identify topics that are of interest to both camps.

There are total 13 papers accepted for presentation at this workshop, with 8 papers for long presentation and 5 for short presentation. These papers are almost evenly divided between the industry and academic communities. In addition, two panels on the related dialog topics have been arranged during the workshop, with distinguished panelists of various backgrounds from academic, industrial, and standardization communities.

We are pleased to the see some real convergence from both industry and academic side.
While academic researchers are proposing and building practical dialog systems, industrial researchers are starting to implement sophisticated learning and uncertainty modeling into their system. The scope of this workshop papers ranges from advanced dialog systems for technical support, multi-modal methods, to POMDP modeling, reinforcement learning and adaptable dialog architecture.

Finally, we would like to thank our program committee members for their work, and thank the NAACL-HLT conference organizers for their timely support. Together, we hope to foster and advance the state of art of dialog technologies.

Fuliang Weng -- Bosch Research
Ye-Yi Wang -- Microsoft Research
Gokan Tur -- SRI International
Junling Hu -- Bosch Research
Program Co-Chairs

ORGANIZERS

Fuliang Weng, Bosch Research
Ye-Yi Wang, Microsoft Research
Gokhan Tur, SRI International
Junling Hu, Bosch Research

PROGRAM COMMITTEE

James Allen, University of Rochester
Mark Fanty, Nuance Communications
Sadaoki Furui, Tokyo Institute of Technology
Dilek Hakkani-Tür, ICSI
Juan Huerta, IBM T.J. Watson Research Center
Michael Johnston, AT&T Labs
Yun-Cheng Ju, Microsoft Research, Microsoft
Dekang Lin, Google Labs, Google
Helen Meng, CUHK
Tim Paek, Microsoft Research
Stanley Peters, Stanford University
Roberto Pieracini, SpeechCycle
Alex Rudnicky, CMU
Stephanie Seneff, MIT
Lenhart Schubert, University of Rochester
Steve Young, Cambridge University

Table of Contents

Conference Program

April 26, 2007

8:30–8:40	Opening
8:40–9:00	*Applying POMDPs to Dialog Systems in the Troubleshooting Domain*
	Jason Williams
9:00–9:20	*Training a real-world POMDP-based Dialog System*
	Blaise Thomson, Jost Schatzmann, Karl Weilhammer, Hui Ye and Steve Young
9:20–9:40	*The Multimodal Presentation Dashboard*
	Michael Johnston, Patrick Ehlen, David Gibbon and Zhu Liu
9:20–10:00	*Technical Support Dialog Systems:Issues, Problems, and Solutions*
	Kate Acomb, Jonathan Bloom, Krishna Dayanidhi, Phillip Hunter, Peter Krogh, Esther Levin and Roberto Pieraccini
10:00–10:30	Break
10:30–10:50	*Olympus: an open-source framework for conversational spoken language interface research*
	Dan Bohus, Antoine Raux, Thomas Harris, Maxine Eskenazi and Alexander Rudnicky
10:50–11:10	*Toward Evaluation that Leads to Best Practices: Reconciling Dialog Evaluation in Research and Industry*
	Tim Paek
11:10–11:30	*Experiments on the France Telecom 3000 Voice Agency corpus: academic research on an industrial spoken dialog system*
	Graldine Damnati, Frdric Bchet and Renato De Mori
11:30–11:50	*Experiences of an In-Service Wizard-of-Oz Data Collection for the Deployment of a Call-Routing Application*
	Mats Wirn and Robert Eklund
11:50-1:00	Lunch
1:00–2:30	Panel Discussion
	Bridging the Gap: Academic and Industrial Research in Dialog Technologies
Panelists	Mazin Gilbert, AT&T Labs - Research
	Michael McTear, University of Ulster
	Stanley Peters, Stanford University, CSLI
	Roberto Pieraccini, SpeechCycle
	Alex Rudnicky, CMU

April 26, 2007 (continued)

Applying POMDPs to Dialog Systems in the Troubleshooting Domain

Jason D. Williams
AT&T Labs – Research
180 Park Ave, Building 103
Florham Park, NJ 07932
jdw@research.att.com

Abstract

This paper reports on progress applying partially observable Markov decision processes (POMDPs) to a commercial dialog domain: troubleshooting. In the troubleshooting domain, a spoken dialog system helps a user to fix a product such as a failed DSL connection. Past work has argued that a POMDP is a principled approach to building spoken dialog systems in the simpler slot-filling domain; this paper explains how the POMDPs formulation can be extended to the more complex troubleshooting domain. Results from dialog simulation verify that a POMDP outperforms a handcrafted baseline.

1 Introduction

In the *troubleshooting domain*, a spoken dialog system (SDS) helps a user to restore a malfunctioning product such as a DSL connection to a working state. Building dialog systems for this domain presents several new challenges. First, the user may make mistakes such as misinterpreting the meaning of a status light or pressing the wrong button, so even if no speech recognition errors are made, the user's response may be misleading. Next, in addition to the speech recognizer, input is also received from running network tests such as pinging the user's DSL modem. Input from both sources may contain errors, and a dialog system must cope with conflicting information from two channels. In sum, the dialog system never knows the true state of the product nor the user's true actions, yet must still instruct the user to successfully restore the product to a working state.

Dialog models which explicitly model uncertainty have been shown to significantly outperform baseline models which do not, primarily because they cope better with conflicting evidence introduced by speech recognition errors (Roy et al., 2000; Zhang et al., 2001; Williams and Young, 2007). However, past work has been confined to slot-filling tasks and has not tackled the troubleshooting domain. Conversely, dialog systems for troubleshooting in the literature have not attempted to model uncertainty directly (Grosz and Sidner, 1986; Lochbaum, 1998).

The contribution of this paper is to show how to model a troubleshooting spoken dialog system as a partially observable Markov decision process (POMDP). We argue that past work in the general troubleshooting literature represents simplifications or special cases of a POMDP, then we show how a troubleshooting POMDP can be combined with a dialog system POMDP to create a unified framework that admits global optimization. Experiments with simulated users show how the POMDP formulation effectively balances diagnostic actions (such as a network test) with communicative actions (such as giving the user instructions), and how the POMDP formulation outperforms a hand-crafted baseline both in terms of efficiency and task completion.

This paper is organized as follows. Section 2 reviews POMDPs, the general troubleshooting problem, and POMDP-based spoken dialog systems; section 3 explains how these two POMDPs can be combined to model a troubleshooting spoken dialog system; sections 4-5 present results from simulation; and section 6 concludes.

2 Background

A POMDP is a model for control when there is uncertainty in the effects of actions and in the state of the environment. Formally, a POMDP $\mathfrak{P}$ is defined as a tuple $\mathfrak{P} = (\mathbb{S}, \mathbb{A}, \mathbb{T}, \mathbb{R}, \mathbb{O}, \mathbb{Z}, \gamma, b_0)$ where $\mathbb{S}$ is a set of states s describing the environment with $s \in \mathbb{S}$; $\mathbb{A}$ is a set of actions $a \in \mathbb{A}$ which operate on the environment; $\mathbb{T}$ defines a transition probability $P(s'|s,a)$; $\mathbb{R}$ defines the expected (immediate, real-valued) reward $r(s,a) \in \Re$; $\mathbb{O}$ is a set of observations $o \in \mathbb{O}$ which describe the state of the environment; $\mathbb{Z}$ defines an observation probability $P(o'|s',a)$; γ is a geometric discount factor $0 \leq \gamma \leq 1$; and b_0 is an initial belief state, defined below.

The POMDP operates as follows. At each time-step, the environment is in some unobserved state s. Since s is not known exactly, a *distribution* over possible states called a *belief state* b is maintained where $b(s)$ indicates the probability of being in a particular state s, with b_0 denoting the initial belief state. Based on b, a control algorithm (also called a *policy*) selects an action a, receives a reward r, and the environment transitions to (unobserved) state s', where s' depends only on s and a. The environment then generates an observation o' which is dependent on s' and a. At each time-step, b is updated as

$$b'(s') = \eta \cdot P(o'|s',a) \sum_s P(s'|s,a) b(s) \qquad (1)$$

where η is a normalization constant (Kaelbling et al., 1998). The process of maintaining b at each time step is called *belief monitoring*. The cumulative, infinite-horizon, discounted reward is called the *return* and written $V = \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$, where s_t and a_t indicate the state of the environment and the action taken at time t, respectively. The goal of the control algorithm is to choose actions that maximize the expected return $E[V]$ given b and the POMDP parameters $\mathfrak{P}$, and the process of searching for such a control algorithm is called *optimization*.

2.1 Troubleshooting as a POMDP

The goal of the general (non-dialog) problem of automated troubleshooting is for a control algorithm to fix a product by taking a sequence of diagnosis and repair actions. Different actions have different reliabilities and different costs, and the aim is to find the sequence that minimizes the total cost. Since the actions are not completely reliable, the true state of the product can't be known with certainty: for example, an instrument may provide a faulty reading.

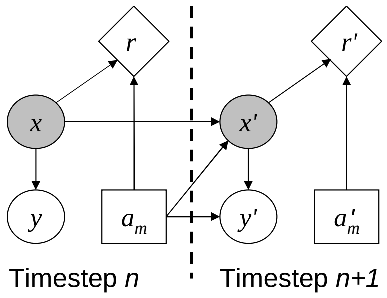


Figure 1: Influence diagram depiction of automated troubleshooting. Round nodes represent random variables, shaded nodes are unobservable and clear nodes are observable. Arcs show conditional dependence. Squares indicate actions, selected by the policy. Diamonds indicate real-valued rewards.

Formalizing this, a product has some hidden state x, which is usually decomposed into components $x = (x_1, x_2, \ldots, x_n)$. A control algorithm takes action a_m, which changes the state of x according to $P(x'|x, a_m)$. The product then produces an observation y according to $P(y'|x', a_m)$. Replacing cost with reward, the control algorithm receives reward $r(x, a_m)$ and the goal is to find the sequence of actions which maximizes the cumulative sum of reward. When viewed in this way, automated troubleshooting can be readily viewed as a POMDP (Shakeri et al., 1997). Figure 1 shows the automated troubleshooting task as an influence diagram.

Although POMDPs are an elegant model for troubleshooting, they are also notoriously difficult to optimize and much of the troubleshooting literature seeks appropriate constraints which render the optimization tractable, such as assuming that each action affects at most one product state component, that actions have deterministic effects, and that there is only fault present (Heckerman et al., 1995). More recently, advances in the POMDP literature have radically increased the scalability of optimization algorithms: for example, Poupart optimizes a substantial network troubleshooting problem cast as a generic POMDP (Poupart and Boutilier, 2004). Viewing troubleshooting as a generic POMDP increases the scope of admissible troubleshooting tasks, and as will be discussed in section 3, this view also allows the uncertainty in the product state to be explicitly modelled in a spoken dialog system.

2.2 Spoken dialog as a POMDP

Past work has argued that POMDPs represent a principled approach to modelling (non-troubleshooting) spoken dialog systems (Roy et al., 2000; Zhang et al., 2001; Williams and Young, 2007). The intuition is that a user's goals and actions form the unobserved state and the (possibly erroneous) ASR result forms the observation. The SDS-POMDP model (Williams and Young, 2007) formalizes this by decomposing the POMDP state variable s into three components, $s = (s_u, a_u, d)$. The component s_u gives the *user's goal*, such as a complete travel itinerary in a travel reservation task. The component a_u gives the most recent *user action* (communicative intent), such as stating a place the user would like to travel to. Finally the component d records relevant *dialog history*, such as the grounding status of a slot. None of these components is observable directly by the dialog system and the SDS-POMDP belief state is formed of a distribution over these components $b(s_u, a_u, d)$. The POMDP action a corresponds to the dialog system action a_m, such as asking the user where they want to go to. Finally, the POMDP observation o is set to $(\tilde{a}_u, c)$, where $\tilde{a}_u$ is the hypothesis of the user's action (communicative intent) provided by the speech recognition and understanding process, and c is the confidence score. Figure 2 shows the SDS-POMDP model as an influence diagram, and also shows the conditional dependencies assumed in the SDS-POMDP model.

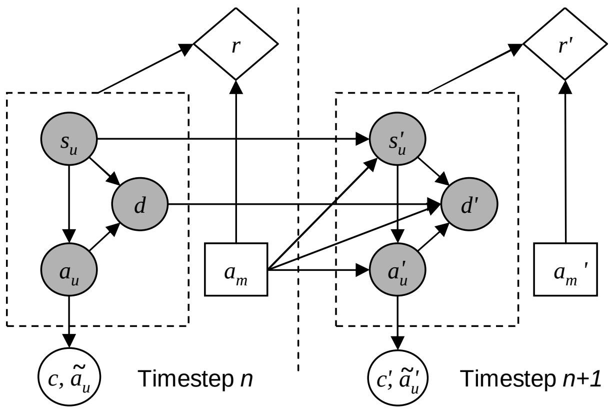


Figure 2: SDS-POMDP model shown as an influence diagram. The dotted box refers to all of the (hidden) POMDP state components.

3 Troubleshooting SDS-POMDP model

In this section, we develop a statistical model of a troubleshooting dialog system. The formulation begins by taking the union of the state spaces of the dialog POMDP and the troubleshooting POMDP, (s_u, a_u, d, x), and making two modifications. First, it is assumed that the user's goal s_u is known and constant (i.e., to fix the product), and as such does not need to be included. Second, the user's action a_u is decomposed into two components: a_u^{ts} denotes *troubleshooting* actions that are directed toward the product, such as turning a modem on or off, entering a user name or just observing the status lights; and a_u^{com} denotes *communicative* actions to the dialog system such as saying "green" or "yes". Reordering, the combined POMDP state has components:

$$s = (a_u^{ts}, x, a_u^{com}, d). \tag{2}$$

Next, the combined observation is formed of the union of the observations from the dialog and troubleshooting POMDPs:

$$o = (\tilde{a}_u^{com}, c, y). \tag{3}$$

Finally, since the POMDP may choose only one action at each time-step, the POMDP action is simply a_m.

Substituting eq. 2 into the POMDP transition function $P(s'|s, a)$ yields $P(a_u^{ts\prime}, x', a_u^{com\prime}, d'|a_u^{ts}, x, a_u^{com}, d, a_m)$ and is decomposed as follows. First, it is assumed that the user's troubleshooting action $a_u^{ts\prime}$ depends only on the system's action a_m, the previous product state x and the dialog history d. Next, it is assumed that the product state x' depends only on the previous product state x, and the most recent user's and dialog system's troubleshooting actions $a_u^{ts\prime}$ and a_m. Further, the user's communicative action $a_u^{com\prime}$ depends only on the most recent user's troubleshooting action $a_u^{ts\prime}$, product state x', dialog history d and system action a_m. Finally, the dialog history component d' is a function of the previous dialog history d and the most recent user and dialog system actions $a_u^{ts\prime}$, $a_u^{com\prime}$, and a_m. With these assumptions, the combined transition function is:

$$\begin{aligned} &P(a_u^{ts\prime}, x', a_u^{com\prime}, d'|a_u^{ts}, x, a_u^{com}, d, a_m) \approx \\ &\quad P(a_u^{ts\prime}|x, d, a_m) \cdot P(x'|x, a_m, a_u^{ts\prime}) \cdot \\ &\quad P(a_u^{com\prime}|d, a_m, a_u^{ts\prime}, x') \cdot \\ &\quad P(d'|d, a_m, a_u^{ts\prime}, x', a_u^{com\prime}) \end{aligned} \tag{4}$$

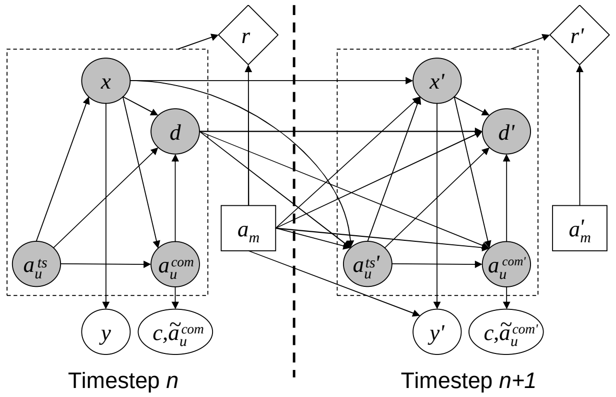


Figure 3: Influence diagram of a troubleshooting spoken dialog system.

Substituting eq. 3 into the POMDP observation function $P(o'|s', a)$ yields $P(\tilde{a}_u^{com\prime}, c', y'|a_u^{ts\prime}, x', a_u^{com\prime}, d', a_m)$. It is assumed that the ASR hypothesis $\tilde{a}_u^{com\prime}$ and confidence score c' depend only on the user's speech in $a_u^{com\prime}$ and that the result of the troubleshooting test (conducted by the dialog system) y' depends only on the state of the product x' and the dialog system's action a_m:

$$P(\tilde{a}_u^{com\prime}, c', y'|a_u^{ts\prime}, x', a_u^{com\prime}, d', a_m) \approx \\ P(\tilde{a}_u^{com\prime}, c'|a_u^{com\prime}) \cdot P(y|a_m, x') \tag{5}$$

An influence diagram of the model is shown in Figure 3.

At runtime, a belief state (i.e., distribution) is maintained over the POMDP state variables, $b(a_u^{ts}, x, a_u^{com}, d)$. Based on this belief state the policy chooses an action a_m and receives observation $(\tilde{a}_u^{com\prime}, c', y')$. The belief state is updated by applying Eq 1, and the cycle repeats.

The user action models $P(a_u^{ts\prime}|x, d, a_m)$ and $P(a_u^{com\prime}|d, a_m, a_u^{ts\prime}, x')$ indicate how users are likely to respond in troubleshooting dialogs and can be estimated from annotated dialog data. The product models $P(x'|x, a_m, a_u^{ts\prime})$ and $P(y'|a_m, x')$ indicate how user and dialog system actions change the state of the product and the reliability of tests, and these can be estimated by interviewing domain experts or by examining logs of product performance. As in the SDS-POMDP model, the dialog history model $P(d'|d, a_m, a_u^{com\prime}, x', a_u^{ts\prime})$ can be handcrafted so as to incorporate features from the dialog history which the dialog designer believes are important, such as appropriateness or notions of grounding. The ASR confusion model $P(\tilde{a}_u^{com\prime}, c'|a_u^{com\prime})$ can be estimated from speech recognition data or derived analytically. The reward function can include distinct costs for different diagnostic tests, dialog actions, and for successful/unsuccessful task completion. It is not specified explicitly here since it depends on the application.

4 Illustration: DSL-1

To illustrate the general framework, we first created a very simple troubleshooting spoken dialog system called DSL-1. Table 1 shows the values for all of the variables. In DSL-1, the are just 2 possible problems: *no-power* and *no-network*.

The conditional probability tables composing the model were handcrafted based on conversations with troubleshooting experts and past experience with spoken dialog systems. For example, the model of user's troubleshooting action assumes that the user performs the correct action with $p = 0.9$, doesn't understand with $p = 0.05$, and performs an incorrect action with $p = 0.05$. The model of the user's communicative action assumes that the user provides correct (but possibly incomplete) information with $p = 0.9$, and remains silent with $p = 0.1$.

The model of the product was designed such that the user's *check-power* and *check-network* actions are always effective, but if power is restored there may still be *no-network* with $p = 0.2$.

The model of the speech recognition and understanding process uses a concept error rate of 30%, where errors are uniformly distributed, and no confidence scores are used. For example, when the user expresses the concept *all-ok*, it will be recognized correctly 70% of the time, and will be misrecognized as *no-power* 5% of the time, as *no-network* 5% of the time, etc. The model for y indicates how reliable the *ping* action is, set with a parameter p_{err}: for example if $p_{err} = 0.1$, the result of a ping test will be incorrect 10% of the time. In the experiments below, the value of p_{err} is varied to explore how the POMDP policy trades off between the *ping* action and communicative actions.

The reward function provides $+100$ for taking the *end-call* action when the connection is working, -100 for taking the *done* action when the connection isn't working, and -1 for any communicative or test action. The dialog continues until the dialog

	Variable	Values
State Components	a_u^{ts}	$\{$check-power, check-network, observe, do-nothing, dont-understand$\}$
	x	$\{$all-ok, no-power, no-network$\}$
	d	$\{$start, not-done, done$\}$
	a_u^{com}	$\{$no-power, no-network, power-ok, all-ok, silent, didnt-understand$\}$
Observation Components	$\tilde{a}_u^{com}$	(same set as a_u^{com})
	y	$\{$ping-ok, no-response$\}$
Action	a_m	$\{$ping, ask-working-ok, req-check-power, req-check-network, end-call$\}$

Table 1: Variable values in the DSL-1 simple troubleshooting example.

system takes the *done* action, at which point the dialog is over.

4.1 Results

The POMDP was optimized using a standard algorithm from the literature (Spaan and Vlassis, 2005). This algorithm optimizes the policy at a discrete set of belief points; as more points are added, the quality of the resulting policy improves at the expense of more computation. We found that 300 belief points achieved asymptotic performance. A model was constructed for values of p_{err} ranging from 0.0 to 0.5; each model was optimized and then evaluated using 5000 simulated dialogs.

Results are shown in Figures 4 and 5. In each figure the x-axis is the accuracy of the *ping* action: $p_{err} = 0\%$ indicates that the *ping* action is entirely reliable and $p_{err} = 50\%$ indicates that the *ping* action returns useless noise. In Figure 4, the y-axis shows average return, and in Figure 5, the solid line shows the task completion rate and the dotted line shows the average dialog length. The error bars indicate the 95% confidence interval.

As the error rate for the *ping* action increases from 0% to 20%, the average dialog length increases from 5.1 turns to 6.5 turns, and the successful task completion rate falls from 100.0% to 98.9%. These figures then remain broadly constant from 20% to 50%. In other words, as errors in the ping action increase, dialogs become longer and occasionally the system fails to fix the connection. Inspecting the dialog transcripts showed that at $p_{err} = 0\%$, the policy relies on the *ping* action to judge whether the connection is working. As p_{err} increases, the policy decreasingly employs the *ping* diagnostic action in favor of the *ask-working-ok* communicative action until $p_{err} = 20\%$, at which point the ping action is

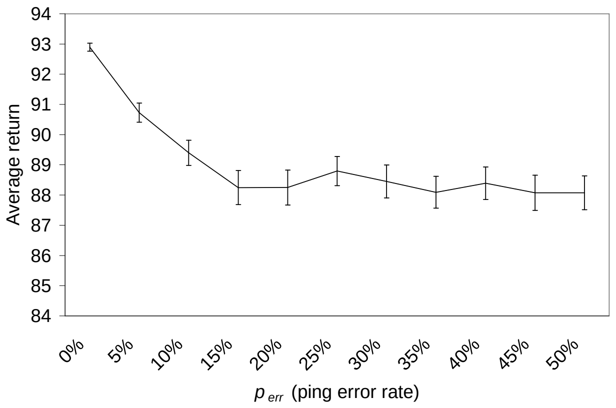


Figure 4: Error rate of the *ping* action vs. reward gained per dialog. As the error rate of the *ping* action is increased, performance declines until the error rate reaches 20%, at which point the system no longer uses the *ping* action.

not used at all. At this point the planning process has determined that the ping action doesn't help produce better dialogs than just interacting with the caller, and the performance from 20% to 50% is constant.[1]

These experiments confirm that, for a very simple troubleshooting dialog system in simulation, the POMDP approach is able to synthesize noisy information gained from communicative and test actions into one unified belief while the underlying, hidden product state is changing. This is an important result because past work that has applied POMDPs to dialog systems has employed a single modality (communicative actions), and have largely had fixed persistent state. Even so, this illustration is much too small to be of practical use, and relies entirely on hand-crafted models of the dynamics. In the next section a model of realistic scale is presented with transition dynamics estimated from real conversa-

[1]The variations in performance between 20% and 50% are due to sampling in the optimization algorithm.

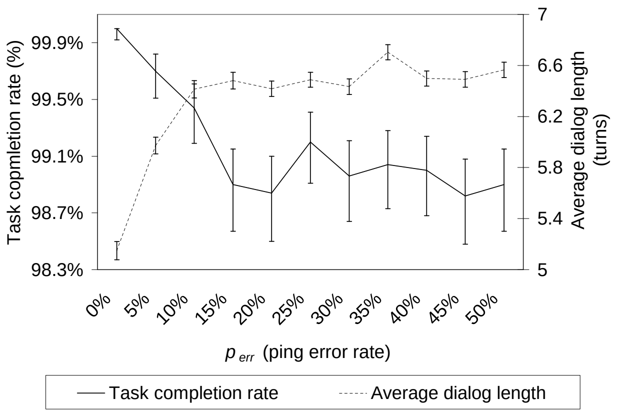


Figure 5: Error rate of the *ping* action vs. successful task completion rate and average dialog length. The left y axis and the solid line show the task completion rate, and the right y axis and the dotted line show the average dialog length in number of turns.

tional data.

5 Illustration: DSL-2

In this section we present a second POMDP-based troubleshooting dialog system called DSL-2 which captures many of the properties of a real-world DSL troubleshooting task. Approximately 100 telephone calls between (human) DSL support agents and customers were monitored, and the observations of these conversations guided creation of the dialog system, including typical problems, agent instructions, and user responses. The product state X was decomposed into 19 components which track, for example, whether there are any outages reported, whether the DSL modem is switched on, and whether the username has been entered correctly in the DSL configuration. Seven of these components can cause the connection to fail: (1) router powered off or crashed, (2) an upstream network crash, (3) a service outage, (4-6) a wrong username, password, or connection type entered in the DSL modem configuration, and (7) an unknown root cause which can't be fixed by the dialog system. Some of the problems can only be identified or fixed by the dialog system (such as a service outage or an upstream network crash), and the rest only by the user (such as a router being off or wrong username entered). The problems may occur in any combination: for example, there may be a service outage while the user's password is entered incorrectly. The system action set (A_m) consisted of 18 actions such as asking the user to turn the modem on, providing the correct username, checking whether any outages have been reported, and rebooting the upstream network interface. The user's troubleshooting action set A_u^{ts} consisted of 12 actions such as turning the modem on or off, opening the DSL configuration screen, entering a password, and attempting to surf to a website. The user's communicative action set A_u^{com} consisted of 11 actions such as saying the color of a light (e.g., "red" or "green"), yes and no, back-channel, silence, and an "out-of-grammar" action which accounts for user speech which cannot be recognized.

The conditional probability tables for each of the product components were handcrafted based on interviews with DSL technicians and are almost all deterministic. For example, if the DSL modem is powered on, the power light will always be on. Next a subset of the agent/user telephone calls were transcribed and annotated with simple dialog acts, and from these the two user models were estimated. Smoothing was applied so that the models allow for the user to take any action at any point in the dialog. Concept recognition errors were generated with $p = 0.30$, and confidence scores were drawn from an exponential distribution such that (at an equal error rate confidence threshold) about half of the concept errors could be identified. The reward function provides $+100$ for ending the dialog having correctly identified (and if possible resolved) the root causes, -100 for ending the dialog with unidentified or unresolved root causes, and -1 for any other action. If a dialog ran for more than 100 turns, it was considered a failure and terminated.

We created a state-based dialog manager by hand (called HC) which broadly reflects the agents' troubleshooting practices and which serves as our baseline. HC consisted of 19 dialog states, where each state specified an action a_m to take (for example to ask the user to turn the modem on), and observations from the speech recognizer $\tilde{a}_u^{com}$ or troubleshooting tests y may cause transitions between dialog states. HC first asks the user to power cycle the modem, then checks for outages and "resets" the upstream network interface, then verifies that the username, password, and network type are configured correctly on the router. After each step HC checks if the connection is working by asking if the network light is green, pinging the modem, then asking the user

	POMDP	HC	HC(0)
CER	30%	30%	0%
N	500	500	500
TCR	96.1%	78.0%	88.6%
Length	19.9	76.5	48.5
Return	73.3	8.13	48.8

Table 2: Results for the POMDP and hand-crafted dialog managers. CER is concept error rate; TCR is task completion rate; Length is measured in turns.

to open a web browser; if any one of these tests fails, troubleshooting resumes, and if they all succeed then HC ends the dialog. If an outage is detected, HC says this and exits, and if the connection still isn't working at the end of the dialog then HC escalates the call to a (human) technician. In general when HC receives an unexpected answer or confidence score below the equal-error rate threshold, it treats this as a likely speech recognition error and remains in the same dialog state.

Next, optimization was performed as described in (Williams et al., 2005). This technique takes as input a POMDP model and a state-based dialog controller, and produces an improved dialog controller. Space limitations prevent a full description here; the intuition is that the algorithm uses the POMDP belief state at runtime to "rewire" the dialog controller to achieve an improvement in reward. Because this optimization algorithm improves a standard state-based dialog controller (in this case the HC baseline), it provides an indication of the value of adding the POMDP machinery.

5.1 Results and discussion

First, 500 simulated dialogs were run with the POMDP, and then 500 simulated dialogs were run with the HC baseline controller. Finally, as a further comparison, the ASR simulation was changed so that no ASR errors were made, and HC was run for 500 dialogs in this configuration, which we call HC(0). Results are shown in Table 2. All of the observed differences are statistically significant ($p \ll 0.01$).

In the presence of speech recognition errors, the POMDP produces dialogs which are significantly shorter and more successful than HC. Moreover, the POMDP, which faced ASR errors, also outperforms HC(0), which did not. Examination of the dialog transcripts found that the main source of failure for HC(0) was exceeding 100 turns. In other words, quantitatively, the POMDP is both more robust to ASR errors and (independent of ASR errors) more efficient.

The dialog transcripts were inspected to determine qualitatively how the POMDP attained better performance. An example is shown in Table 3. At the start of the conversation, the belief (probability) that the connection is working $p(\text{allOk})$ is 56% and the belief that the power to the DSL modem is on $p(\text{pwrOn})$ is 98.0% (these are 2 of the 19 components in the product state x). As the dialog progresses, belief monitoring updates these to account for the evidence received. For example, the unsuccessful *ping* in S1 causes $p(\text{allOk})$ to drop from 56% to 14%. The belief monitoring process also naturally makes use of indirect evidence – for example, in U14 the user indicates the network light is "red": since the network light will only be on if the power to the DSL modem is on, this causes an increase in the belief that the power is on, from 99.1% to 99.8%.

The key benefit of the POMDP approach is that the dialog manager can exploit the belief state to make better progress in the face of low-confidence or even nonsensical replies, without sacrificing overall task completion. For example, in S1 through S9 the POMDP policy differs from the baseline controller: the baseline controller would have ignored the lower-confidence recognitions in U4 and U8, but the POMDP policy moves ahead. When the policy receives a nonsensical reply, for example in U6, it reverts back to an earlier stage of the troubleshooting procedure it had previously skipped. This latter behavior ensures that omitting steps to move faster through the procedure doesn't ultimately sacrifice task completion.

6 Conclusions

This paper has shown how a spoken dialog system for troubleshooting can be cast as a POMDP. The troubleshooting domain has important differences to past applications of the POMDP approach and the two illustrations provided in this paper support our claim that, at least in dialog simulation, the advantages of POMDPs apply to this domain.

After finishing simulation experiments, we installed DSL-2 into a real dialog system, and found that belief monitoring runs slower than real-time. We subsequently developed a method to address this, which we will report on separately in the future, and are now preparing for a pilot study with real users.

References

BJ Grosz and CL Sidner. 1986. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12(3):175–204.

D Heckerman, JS Breese, and K Rommelse. 1995. Decision-theoretic troubleshooting. *Communications of the ACM*, 38(3):49–57.

L Kaelbling, ML Littman, and AR Cassandra. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101.

KE Lochbaum. 1998. A collaborative planning model of intentional structure. *Computational Linguistics*, 24(4):525–572.

P Poupart and C Boutilier. 2004. VDCBPI: an approximate scalable algorithm for large scale POMDPs. In *Proc Advances in Neural Information Processing Systems 17 (NIPS), Vancouver, Canada*, pages 1081–1088.

N Roy, J Pineau, and S Thrun. 2000. Spoken dialog management for robots. In *Proc Association for Computational Linguistics (ACL), Hong Kong*.

M Shakeri, KR Pattipati, V Raghavan, A Patterson-Hine, and DL Iverson. 1997. Multiple fault isolation in redundant systems. Technical Report NCC2-5123, NASA Ames.

MTJ Spaan and N Vlassis. 2005. Perseus: randomized point-based value iteration for POMDPs. *Journal of Artificial Intelligence Research*, 24:195–220.

JD Williams and SJ Young. 2007. Partially observable markov decision processes for spoken dialog systems. *Computer Speech and Language*, 21(2):393–422.

JD Williams, P Poupart, and SJ Young. 2005. Partially observable Markov decision processes with continuous observations for dialogue management. In *Proc SIGdial Workshop on Discourse and Dialogue, Lisbon*.

B Zhang, Q Cai, J Mao, and B Guo. 2001. Planning and acting under uncertainty: A new model for spoken dialogue system. In *Proc Conf on Uncertainty in Artificial Intelligence (UAI), Seattle, Washington*, pages 572–579.

ID	p(allOk)	p(pwrOn)	Transcript
*S1	56%	98.0%	I'm going to try pinging your DSL modem from here.
S2	-	-	There was no response so it seems your connection isn't working yet.
*S3	14%	96.1%	Please turn the DSL modem on.
U4	-	-	*ok* [ok, 0.004]
*S5	30%	96.4%	What color is the network light on the DSL modem?
U6	-	-	*dontunderstand* [no, 0.052]
*S7	30%	93.8%	Please turn the DSL modem on.
U8	-	-	*dontunderstand* [ok, 0.454]
*S9	31%	99.1%	I'm going to check if there are any outages reported, just a moment.
S10	-	-	There are no outages reported.
S11	39%	99.1%	I'm going to try resetting your connection from here, just a moment.
S12	-	-	Ok, that has been done.
S13	48%	99.1%	What color is the network light on the DSL modem?
U14	-	-	*red* [red, 0.678]
S15	3%	99.8%	Please open the DSL configuration screen.
...	...	...	...

Table 3: Fragment of a conversation with the POMDP dialog manager. Asterisks (*) indicate transitions not in the baseline dialog manager. p(allOk) shows the probability that DSL connectivity is working and p(pwrOn) shows the probability that the power to the DSL modem is on, according to the POMDP belief state. The simulated user's communicative actions are shown in *italics*, followed by the [concept, confidence score] produced by the ASR simulation.

Training a real-world POMDP-based Dialogue System

Blaise Thomson, Jost Schatzmann, Karl Weilhammer, Hui Ye and Steve Young
Cambridge University Engineering Department
Trumpington Street, Cambridge, CB21PZ, United Kingdom
{brmt2, js532, kw278, hy216, sjy}@eng.cam.ac.uk

Abstract

Partially Observable Markov Decision Processes provide a principled way to model uncertainty in dialogues. However, traditional algorithms for optimising policies are intractable except for cases with very few states. This paper discusses a new approach to policy optimisation based on grid-based Q-learning with a summary of belief space. We also present a technique for bootstrapping the system using a novel agenda-based user model. An implementation of a policy trained using this system was tested with human subjects in an extensive trial. The policy gave highly competitive results, with a 90.6% task completion rate.

1 Introduction

Recent work on statistical models for dialogue systems has argued that Partially Observable Markov Decision Processes (POMDPs) provide a principled mathematical framework for modeling the uncertainty inherent in human-machine dialogue (Young, 2006). Briefly speaking, POMDPs extend the traditional (fully-observable) Markov Decision Process (MDP) framework by maintaining a *belief state*, ie. a probability distribution over dialogue states. This enables the dialogue manager to avoid and recover from recognition errors by sharing and shifting probability mass between multiple hypotheses of the current dialogue state. The framework also naturally incorporates n-best lists of multiple recognition hypotheses coming from the speech recogniser.

Due to the vast space of possible belief states, however, the use of POMDPs for any practical system is far from straightforward. Exact algorithms for solving POMDPs do exist, but have been shown to be intractable except for domains limited to a few states (Kaelbling et al., 1998). In a practical dialogue system the minimum number of dialogue states is typically determined by the number of possible user goals, and this number usually far exceeds the limits of exact solution algorithms.

Approximate algorithms have been developed to overcome the intractibility of exact algorithms but even the most efficient of these techniques such as Point Based Value Iteration (PBVI) cannot scale to the many thousand states required by a statistical dialogue manager (Williams, 2006; Pineau et al., 2003). Previous work by Williams and Young (2006) on Composite Summary Point Based Value Iteration (CSPBVI) has suggested the use of a small *summary space* for each slot where PBVI policy optimisation can be applied. This has shown to give good results on synthetic data but remains untested within real dialogue systems.

One potential problem with the CSPBVI technique is that policy learning can only be performed *offline*, ie. at design time, because policy training requires an existing accurate model of user behaviour. In this paper, an alternative technique for *online* training based on Q-learning is presented. Online training allows the system to adapt to real users as new dialogues are recorded.

The learning algorithm presented here does not require any model of user behaviour so initial dialogues may well be incoherent. In fact, the system requires several thousand dialogues before convergence to a suitable policy begins. This means that in practice the model needs to be bootstrapped via a user simulator. Further adapatation can then be done with real users.

The paper is organised as follows. Section 2 provides an introduction to the POMDP model and explains the Summary POMDP framework that is used in the remainder of the paper. A new online method for policy optimisation is presented in Section 3 and a novel agenda-based user model for bootstrapping the system is introduced in Section 4. Section 5 discusses an evaluation of a sample implementation built for a Tourist Information System and tested with human subjects. The system performed competitively with 90.6% of tasks successfully completed despite a mix of native and non-native speakers. The paper concludes with a summary and some directions for future work.

2 Background

2.1 POMDP Basics

A POMDP is defined in much the same way as an MDP, except that the states are not observable and instead have to be estimated from observations. Formally, a POMDP is a tuple $\{S_m, A_m, T, R, O, Z, \lambda, \boldsymbol{b}_0\}$ where:

- S_m is a set of machine states
- A_m is a set of actions that the machine may take
- O is a set of possible observations
- T defines the transition probability such that $T(s_m, a_m, s'_m) = P(s'_m|s_m, a_m)$
- R defines the immediate reward obtained from taking a particular action in a particular state to be $r(s_m, a_m)$
- Z defines the probability of a particular observation given the state and machine action $P(o'|s'_m, a_m)$
- λ is a geometric discount factor $0 \leq \lambda \leq 1$
- $\boldsymbol{b}_0$ is an initial belief state.

When the POMDP operates, it also makes use of a policy $\pi : \Pi(S) \longrightarrow A_m$ that chooses an action given a point in belief space. Here $\Pi(S)$ is the set of all possible probability distributions over S_m (an $|S_m| - 1$ dimensional simplex). $\pi(\boldsymbol{b})$ gives the action to take when the POMDP is in belief state $\boldsymbol{b}$.

The sequence of events in the POMDP follows a cycle. At each time step, the machine is in some unobserved state $s_m \in S_m$. Since the true state is unknown, the machine maintains a probability distribution over the states $\boldsymbol{b}$, which is called the belief state. Based on this belief state and the policy π being followed, the system takes an action $a_m = \pi(\boldsymbol{b})$. The machine is rewarded with $r(s_m, a_m)$ and the state transtitions to a new unobserved state s'_m with probability $T(s_m, a_m, s'_m)$. The machine then receives an observation $o' \in O$, with probability dependent only on the new state s'_m and the machine action a_m. The belief state is updated based on the events of the turn and the cycle repeats. The belief state update is computed as

$$b'(s'_m) = k \cdot P(o'|s'_m, a_m) \sum_{s_m \in S_m} P(s'_m|a_m, s_m) b(s_m) \tag{1}$$

where k is a normalisation constant(Kaelbling et al., 1998). Maintaining this belief state as the dialog evolves is called *belief monitoring*.

Figure 1 shows a graphical representation of a POMDP based dialogue system. When the user utters a user act a_u it is transmitted via speech to the dialogue system. From the machine's point of view, the speech acts as a noisy channel so that the observation received, o, is not necessarily the true dialogue act. Instead it typically describes an n-best list of hypothesised user acts. Based on this, the POMDP belief state is updated and a machine dialogue act, a_m, is selected. As can be seen from the diagram, it is quite trivial to replace a real user with a simulated one so that training can be performed less laboriously.

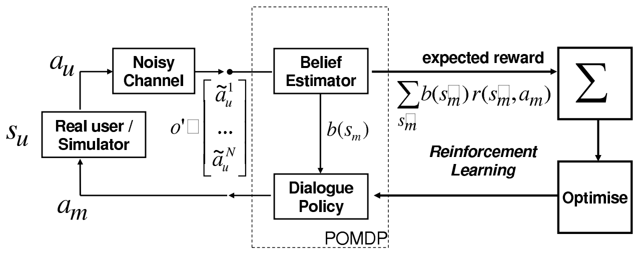


Figure 1: Training a POMDP with a simulated user

Given a particular policy, the infinite horizon expected reward as a function of belief state is called the value function. It is calculated as:

$$\begin{aligned} V^\pi(\boldsymbol{b}) &= \sum_{t=0}^{\infty} \lambda^t r(\boldsymbol{b}_t, a_{m,t}) && (2) \\ &= \sum_{t=0}^{\infty} \lambda^t \sum_{s_m \in S_m} b_t(s_m) r(s_m, a_{m,t}) && (3) \\ &= \sum_{t=0}^{\infty} \lambda^t \sum_{s_m \in S_m} b_t(s_m) r(s_m, \pi(\boldsymbol{b}_t)) && (4) \end{aligned}$$

The goal of POMDP policy optimisation is to find the policy that maximises the value function at every point $\boldsymbol{b}$. It can be shown that such a function always exists and is both continuous and convex.

In the context of policy optimisation it is also useful to define the concept of a Q function(Sutton and Barto, 1998). This is a function of both belief state $\boldsymbol{b}$ and action a_m and is simply the expected reward obtained by first taking action a_m and then following the policy π.

The Bellman Optimality Equation states that a policy is optimal if and only if

$$\pi(\boldsymbol{b}) = \arg \max_{a \in S_m} \; Q^\pi(a, \boldsymbol{b}) \tag{5}$$

2.2 The Summary POMDP

As discussed in the introduction, directly optimising POMDPs for dialogue systems is completely impractical. Instead, the belief state and actions are mapped down to a summarised form where optimisation becomes tractable. In this context, the original belief space and actions are

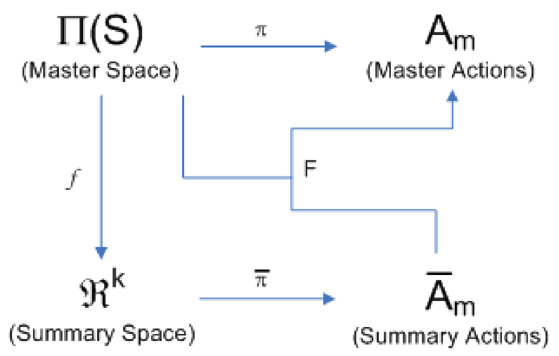


Figure 2: The Summary POMDP framework

called *master* space and *master* actions, while the summarised versions are called *summary* space and *summary* actions.

Action selection in the full model would be a mapping from a belief state $\boldsymbol{b} \in \Pi(S_m)$ to an action $a_m \in A_m$. The summary POMDP splits this up as follows. The model initially extends the standard POMDP with a set of summary actions $\bar{A}_m$ and a mapping from summary actions to master actions F. This function should be allowed access to the master belief space so that the summary can be as brief as possible (formally, $F : \bar{A}_m \times \Pi(S_m) \longrightarrow A_m$). Next, a summarising function f is defined from master belief space $\Pi(S_m)$ to summary belief space $\mathbb{R}^k$. Finally, a summary policy $\bar{\pi}$ is defined as a mapping from summary space $\mathbb{R}^k$ to summary actions $\bar{A}_m$. A policy in master space is composed from the above three functions by first mapping to summary space via f, using policy $\bar{\pi}$ to find an appropriate summary action $\bar{a}_m$ and then obtaining a master action with F. The full process is shown graphically in Figure 2. Algebraically the master policy is defined by:

$$\pi(\boldsymbol{b}) = F(\bar{\pi}(f(\boldsymbol{b})), \boldsymbol{b}) \tag{6}$$

Further explanation of the Summary POMDP method can be found in (Williams and Young, 2005). Note that the formalism introduced above may be used for both the summary methods previously used for dialogue systems as well as belief compression techniques used in a more general setting (Roy et al., 2005)

At the summary level, policies may not have enough information to act truly optimally. Hence defining an optimal summary policy is not so obvious. If f is chosen well, however, then one could hope that the optimal action is dependent only on $f(\boldsymbol{b})$. If this is true then a summary policy is called optimal when the following equation holds for every $\boldsymbol{b}$ such that $f(\boldsymbol{b}) = (\boldsymbol{x})$:

$$\bar{\pi}(\boldsymbol{x}) = \arg \max_{\bar{a} \in \bar{A}_m} \quad Q(F(\bar{a}, \boldsymbol{b}), \boldsymbol{b}) \tag{7}$$

3 Summarised Q-learning

Q-learning is a technique for online learning traditionally used in an MDP framework. It is an iterative Monte-Carlo style algorithm where a sequence of sample dialogues are used to estimate the Q functions for each state and action. Inspired by grid-based methods, the summarised Q-learning algorithm discretises *summary space* and uses Q-learning on the resulting MDP-like grid.

Operation of the algorithm proceeds by simply engaging the dialogue manager with either a real user or a user model. At each point where the system must choose an action, the master belief space is mapped down to the summary level as described in Section 2.2. The nearest summary point in the grid is found and the optimal summary action given by that point is chosen.

At the end of the dialogue, the discounted future reward is known for each stage where a choice was taken. This value is recorded along with the grid point where the decision was made, and the action chosen. This is a sample of the discounted future reward obtained by taking the particular action and then following the current policy - i.e. the Q-function evaluated at this grid point. If sufficient dialogues are done the mean of these values will give a good estimate of the true Q-value.

In order to enable learning, an exploration paramater ϵ is selected so that a random summary act will be chosen with probability ϵ. After a batch of dialogues have been completed, the estimates of the Q-functions are updated with the new dialogue scores. The optimal action is then chosen for each point p by

$$\bar{a}_p = \underset{\bar{a}}{\operatorname{argmax}}\, \hat{Q}(a, p) \tag{8}$$

The selection of which points to put into the grid is a crucial part of the algorithm as one would like the most accuracy at points that will be visited often. As a result, this algorithm uses a variable grid method (Brafman, 1997; Bonet, 2002). During operation, when a point is reached that is further away from any other point than some threshold paramater, the point is added to the grid. This ensures that points are only included in the grid if needed.

Grid-based methods are often criticised because they do not scale well to large state spaces (Pineau et al., 2003). However, when using the Summary POMDP method the state space is reduced significantly before the grid is applied. Although there are no convergence guarantees for this method in the context of Summary POMDPs, Q-learning does guarantee convergence to the optimal policy for standard MDPs. As can be seen from Figure 4, in practice the method does converge to a high performing policy after several thousand dialogues.

4 Agenda-Based Simulation

4.1 User Simulation-Based Training

As described in the introduction to this paper, *online* methods for training statistical dialogue managers allow the dialogue policy to be adapted and improved at runtime, ie. through interaction with real users. During the initial development phase however, many thousand training dialogues are needed to bootstrap the dialogue policy, and this is generally too time-consuming and expensive to be done with real users.

A number of research groups (Levin et al., 2000; Scheffler and Young, 2002; Pietquin and Dutoit, 2005; Georgila et al., 2005; Rieser and Lemon, 2006) have thus investigated the use of user simulation tools for training the dialogue manager (DM). The simulation-based approach typically involves two steps. Firstly, a statistical user model (such as an n-gram or a graphical model) is trained on a limited amount of dialogue data. The model is then used to simulate dialogues with the interactively learning DM (see Schatzmann et al. (2006) for a literature review). Simulation is usually done at a semantic dialogue act level to avoid having to reproduce the variety of user utterances at the word- or acoustic level.

The simulation-based approach assumes the presence of a small corpus of suitably annotated in-domain dialogues (Lemon et al., 2006). For the experiments presented in this paper, no such data was available for training the user model. Hence, it was necessary to develop a model which was simple enough for the model parameters to be handcrafted and yet capable of producing user behaviour realistic enough for training a prototype system. A similar approach has been previously taken by (Levin et al., 2000; Pietquin and Dutoit, 2005) but the performance of the learned dialogue policies was not evaluated using real users.

4.2 User Simulation at a Semantic Level

Human-machine dialogue can be formalised on a semantic level as a sequence of state transitions and dialogue acts[1]. At any time t, the user is in a state s_u, takes action a_u, transitions into the intermediate state s'_u, receives machine action a_m, and transitions into the next state s''_u where the cycle restarts.

$$s_u \rightarrow a_u \rightarrow s'_u \rightarrow a_m \rightarrow s''_u \rightarrow \cdots \qquad (9)$$

Assuming a Markovian state representation, user behaviour can be decomposed into three models: $P(a_u|s_u)$ for action selection, $P(s'_u|a_u, s_u)$ for the state transition into s'_u, and $P(s''_u|a_m, s'_u)$ for the transition into s''_u.

[1]In this paper, the terms *dialogue act* and *dialogue action* are used interchangeably. The notation *act(a=x, b=y,...)* is used to represent a dialogue act of a given type *act* (such as *inform* or *request* with items $a = x$, $b = y$, etc.

4.3 Goal- and Agenda-Based State Representation

Inspired by agenda-based methods to dialogue management (Wei and Rudnicky, 1999) the approach described here factors the user state into an agenda A and a goal G.

$$s_u = (A, G) \qquad \text{and} \qquad G = (C, R) \qquad (10)$$

During the course of the dialogue, the goal G ensures that the user behaves in a consistent, goal-directed manner. G consists of constraints C which specify the required venue, eg. a centrally located bar serving beer, and requests R which specify the desired pieces of information, eg. the name, address and phone number (cf. Fig. 3).

The user agenda A is a stack-like structure containing the pending user dialogue acts that are needed to elicit the information specified in the goal. At the start of the dialogue a new goal is randomly generated using the system database and the agenda is populated by converting all goal constraints into *inform* acts and all goal requests into *request* acts. A *bye* act is added at the bottom of the agenda to close the dialogue.

As the dialogue progresses the agenda is dynamically updated and acts are selected from the top of the agenda to form user acts a_u. In response to incoming machine acts a_m, new user acts are pushed onto the agenda and no longer relevant ones are removed. The agenda thus serves as a convenient way of tracking the progress of the dialogue as well as encoding the relevant dialogue history. As can be seen in Fig. 3 (turns 1-3), user acts can also be temporarily stored when actions of higher priority need to be issued first, hence providing the simulator with a simple model of user memory.

4.4 Action Selection

At any time during the dialogue, the updated agenda of length N contains all dialogue acts the user intends to convey to the system. Since the agenda is ordered according to priority, with $A[N]$ denoting the top and $A[1]$ denoting the bottom item, selecting the next user act simplifies to popping n items off the top of the stack. Hence, letting $a_u[i]$ denote the ith item in the user act a_u

$$a_u[i] := A[N - n + i] \quad \forall i \in [1..n], 1 \leq n \leq N. \quad (11)$$

and the action selection model becomes a Dirac delta function

$$P(a_u|s_u) = P(a_u|A, G) = \delta(a_u, A[N-n+1..N]) \quad (12)$$

where $A[N - n + 1..N]$ is a Matlab-like shorthand notation for the top n items on A and the random variable n corresponds to the level of initiative taken by the simulated user. In a statistical model the probability distribution over integer values for n should be conditioned on A and learned from dialogue data. For the purposes of

$$C_0 = \begin{bmatrix} \mathit{type} = \mathit{bar} \\ \mathit{drinks} = \mathit{beer} \\ \mathit{area} = \mathit{central} \end{bmatrix}$$

$$R_0 = \begin{bmatrix} \mathit{name} = \\ \mathit{addr} = \\ \mathit{phone} = \end{bmatrix}$$

Sys 0 Hello, how may I help you?

$$A_1 = \begin{bmatrix} inform(\mathit{type} = \mathit{bar}) \\ inform(\mathit{drinks} = \mathit{beer}) \\ inform(\mathit{area} = \mathit{central}) \\ request(\mathit{name}) \\ request(\mathit{addr}) \\ request(\mathit{phone}) \\ \mathit{bye}() \end{bmatrix}$$

Usr 1 I'm looking for a fine beer bar.

Sys 1 Ok, a wine bar. What pricerange?

$$A_2 = \begin{bmatrix} negate(\mathit{drinks} = \mathit{beer}) \\ inform(\mathit{price} = \mathit{cheap}) \\ inform(\mathit{area} = \mathit{central}) \\ request(\mathit{name}) \\ request(\mathit{addr}) \\ request(\mathit{phone}) \\ \mathit{bye}() \end{bmatrix}$$

Usr 2 No, beer please!

Sys 2 You are looking for a beer bar, correct?

$$A_3 = \begin{bmatrix} affirm() \\ inform(\mathit{price} = \mathit{cheap}) \\ inform(\mathit{area} = \mathit{central}) \\ request(\mathit{name}) \\ request(\mathit{addr}) \\ request(\mathit{phone}) \\ \mathit{bye}() \end{bmatrix}$$

Usr 3 Yeah something cheap in the town centre.

Sys 3 Murphy's on Main Square serves cheap beer.

$$A_4 = \begin{bmatrix} request(\mathit{phone}) \\ \mathit{bye}() \end{bmatrix}$$

Usr 4 Ok, and what's the phone number?

Sys 4 The number is 796 69 94.

$$A_5 = \begin{bmatrix} \mathit{bye}() \end{bmatrix}$$

Usr 5 Thanks, goodbye!

Figure 3: Sample dialogue and agenda sequence

bootstrapping the system, n can be assumed independent of A and any distribution $P(n)$ that places the majority of its probability mass on small values of n can be used.

4.5 State Transition Model

The factorisation of s_u into A and G can now be applied to the state transition models $P(s'_u|a_u, s_u)$ and $P(s''_u|a_m, s'_u)$. Letting A' denote the agenda after selecting a_u (as explained in the previous subsection) and using $N' = N - n$ to denote the size of A', we have

$$A'[i] \quad := \quad A[i] \quad \forall i \in [1..N']. \tag{13}$$

Using this definition of A' and assuming that the goal remains constant when the user executes a_u, the first state transition depending on a_u simplifies to

$$\begin{aligned} P(s'_u|a_u, s_u) &= P(A', G'|a_u, A, G) \\ &= \delta(A', A[1..N'])\delta(G', G). \end{aligned} \tag{14}$$

Using $s_u = (A, G)$, the chain rule of probability, and reasonable conditional independence assumptions, the second state transition based on a_m can be decomposed into *goal update* and *agenda update* modules:

$$\begin{aligned} &P(s''_u|a_m, s'_u) \\ &\quad = \underbrace{P(A''|a_m, A', G'')}_{\text{agenda update}} \underbrace{P(G''|a_m, G')}_{\text{goal update}}. \end{aligned} \tag{15}$$

When no restrictions are placed on A'' and G'', the space of possible state transitions is vast. The model parameter set is too large to be handcrafted and even substantial amounts of training data would be insufficient to obtain reliable estimates. It can however be assumed that A'' is derived from A' and that G'' is derived from G' and that in each case the transition entails only a limited number of well-defined atomic operations.

4.6 Agenda Update Model

The agenda transition from A' to A'' can be viewed as a sequence of push-operations in which dialogue acts are added to the top of the agenda. In a second "clean-up" step, duplicate dialogue acts, *null()* acts, and unnecessary *request()* acts for already filled goal request slots must be removed but this is a deterministic procedure so that it can be excluded in the following derivation for simplicity. Considering only the push-operations, the items 1 to N' at the bottom of the agenda remain fixed and the update model can be rewritten as follows:

$$\begin{aligned} &P(A''|a_m, A', G'') \\ &\quad = P(A''[1..N'']|a_m, A'[1..N'], G'') \qquad (16) \\ &\quad = P(A''[N'{+}1..N'']|a_m, G'') \\ &\qquad \cdot \ \delta(A''[1..N'], A'[1..N']). \qquad (17) \end{aligned}$$

The first term on the RHS of Eq. 17 can now be further simplified by assuming that every dialogue act item in a_m triggers one push-operation. This assumption can be made without loss of generality, because it is possible to push a *null()* act (which is later removed) or to push an act with more than one item. The advantage of this assumption is that the known number M of items in a_m

now determines the number of push-operations. Hence $N'' = N' + M$ and

$$\begin{aligned} &P(A''[N'{+}1..N'']|a_m, G'') \\ &\quad = \; P(A''[N'{+}1..N'{+}M]|a_m[1..M], G'') \quad (18) \\ &\quad = \; \prod_{i=1}^{M} P(A''[N'{+}i]|a_m[i], G'') \quad (19) \end{aligned}$$

The expression in Eq. 19 shows that each item $a_m[i]$ in the system act triggers one push operation, and that this operation is conditioned on the goal. This model is now simple enough to be handcrafted using heuristics. For example, the model parameters can be set so that when the item *x=y* in $a_m[i]$ violates the constraints in G'', one of the following is pushed onto A'': *negate()*, *inform(x=z)*, *deny(x=y, x=z)*, etc.

4.7 Goal Update Model

The goal update model $P(G''|a_m, G')$ describes how the user constraints C' and requests R' change with a given machine action a_m. Assuming that R'' is conditionally independent of C' given C'' it can be shown that

$$\begin{aligned} &P(G''|a_m, G') \\ &\quad = \; P(R''|a_m, R', C'')P(C''|a_m, R', C'). \quad (20) \end{aligned}$$

To restrict the space of transitions from R' to R'' it can be assumed that each request slot (ag. *addr,phone,etc.*) is either filled using information in a_m or left unchanged. One can further assume that the value of any slot depends on its value at the previous time step, the value provided by a_m and that the transition needs to be conditioned on whether the information given in a_m matches the goal constraints. Using $R[k]$ to denote the k'th request slot we can approximate

$$\begin{aligned} &P(R''|a_m, R', C'') \\ &\quad = \textstyle\prod_k P(R''[k]|a_m, R'[k], \mathcal{M}(a_m, C'')). \quad (21) \end{aligned}$$

To simplify $P(C''|a_m, R', C')$ we assume that C'' is derived from C' by either adding a new constraint, setting an existing constraint slot to a different value (eg. *drinks=dontcare*), or by simply changing nothing. The choice of transition does not need to be conditioned on the full space of possible a_m, R' and C'. Instead it can be conditioned on simple boolean flags such as "Does a_m ask for a slot in the constraint set?", "Does a_m signal that no item in the database matches the given constraints?", etc. The model parameter set is then sufficiently small for handcrafted values to be assigned to the probabilities.

5 Evaluation

5.1 A scalable POMDP-based system

The summary Q-learning algorithm and agenda-based user model were tested by implementing a POMDP-based dialogue system for a Tourist Information Domain. Users are assumed to have arrived in a town unknown to them and must find a bar, a hotel or a restaurant in the town subject to some constraints (eg. a cheap, Chinese restaurant in the centre of town). The town used was fictitious so that users could not know any of the venues.

The speech recognition was implemented using the Application Toolkit for HTK (ATK) with a vocabulary of about 2000 words. A simple keyword-spotting semantic decoder was used to extract meaning representations (dialogue acts) from the output of the recogniser. The dialogue manager is based on the Hidden Information State (HIS) model (Young et al., 2007), which gives an efficient way of implementing the belief state update in a POMDP-based dialogue system.

In the implementation used for testing, the town included approximately 40 possible venues. Eight different variables are used by the system in deciding which venue to recommend: type of venue; pricerange; area; proximity to a particular place; stars; drinks; food and music. Additionally, the user could ask for the average price, the phone number, the address or a comment on a particular venue.

The model allows for a rich structure in possible user goals via simple ontology rules. For example, venues can only have a *food* concept if their type is *restaurant*. Hence, one would expect that most information retrieval type dialogues could be modeled in a similar manner.

5.2 System Training

The HIS manager factors the machine state of the POMDP into three parts: the user's goal, the dialogue state and the last machine act. An important feature of the system is that indistinguishable user goals are grouped together into *partitions*. For example, if the user is trying to obtain information about restaurants and has not mentioned what type of food they would like, then restaurants will be grouped together regardless of the type of food they serve. In the HIS model, a *hypothesis* refers to the grouping of a partition with a dialog state.

The splitting of the machine state into separate hypotheses provides for a simple mapping to summary state for the Q-learning algorithm, where only information from the top two hypotheses is included. The summary state used has five components: the probabilities of the two most probable hypotheses along with three summary features. These enumerate the possibilities for how many database items fall into the partition, a summary of the dialog state and the type of the last dialog act.

Rewards were given based on task completion and the number of turns in the dialogue. The system was given 20 points for a successful dialogue and 0 for an unsuccessful one. One point was subtracted for each dialogue turn. This encourages the system to be sure of the user's goal, while penalising inefficient system behaviour.

Training was done using the agenda-based user model described in Section 4. Initially, batches of 1000 dialogues were performed with no error-modeling, updating Q-values and optimal actions at the end of each batch. Figure 4 shows the average reward obtained from each of these policies over 1000 sample dialogues. The policy score converged after approximately 25 000 dialogues and reached an average score of around 14. This illustrates that the Q-learning approach described above does in fact converge in practice. Further training was then done including simulated errors by changing random concept words with a probability of 0.05.

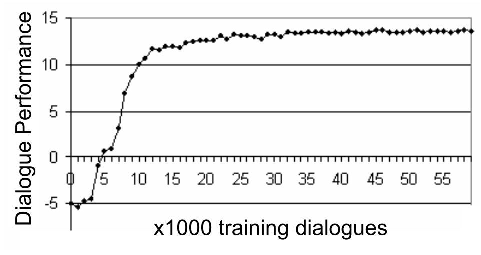


Figure 4: Training a POMDP system

5.3 User testing

The prototype dialogue system described above was evaluated as part of the EU FP6 Talk Project. No online learning was performed during the evaluation. 40 human subjects took part, each of whom completed 4 dialogues. In total, 160 dialogues were recorded, with 6452 turns and 21667 words. Subjects were selected to include a mix of native and non-native speakers and the average Word-Error-Rate was 29.8%.

The performance of the system was measured based on the recommendation of a correct venue, ie. a venue matching all constraints specified in the given task (all tasks were designed to have exactly one solution). Based on this definition, 145 out of 160 dialogues (90.6%) were completed successfully, and the average number of turns to completion was 5.59 (if no correct venue was offered the full number of turns was counted). An extract from an example dialogue is given as an illustration in Fig. 5.

6 Summary and Future Work

This paper has investigated a new online method for training a real-world POMDP-based dialogue system. A novel agenda-based user simulation technique was also presented for bootstrapping the dialogue manager without access to training data. Although there is no guarantee of convergence, evaluation results suggest that in practice the training algorithm does converge to a suitable policy. The results also show that the learned policy works well

Turn 1:	
System:	Hello and welcome to our dialogue system! Are you looking for a hotel, a restaurant or a bar?
User:	I'm looking for a five star hotel, preferably in the East side of town, near the park
ASR:	I'm looking for a five star hotel preferably use side of town near the park
DA:	request(hotel,stars=5,near=Park)
Turn 2:	
System:	What part of town do you have in mind? For example something in the centre of town or in the north or on the riverside
User:	In the East side of town
ASR:	In the East side of town
DA:	inform(area=east)
Turn 3:	
System:	Did you say you are looking for a hotel in the east part of town?
User:	Yes
ASR:	Yes
DA:	affirm()
Turn 4:	
System:	The Royal Hotel is a great hotel in the east part of town
User:	I was wondering whether I could get the telephone number for that town... for that hotel?
ASR:	was wondering about could get the telephone number of the town for a hotel
DA:	request(phone,hotel)
Turn 5:	
System:	The Royal Hotel has 5 stars and is near the Park
User:	Could I have the phone number?
ASR:	I have the phone number
DA:	request (phone)
Turn 6:	
System:	No problem the phone number of the Royal Hotel is 7027003

Figure 5: Sample test dialogue along with most likely output from the automatic speech recognition (ASR) and decoded dialogue act (DA). The dialogue act format is described in (Young et al., 2005).

for human users. Future work will focus on finding better mappings to summary space, on training the simulated user with recorded data and on investigating the effectiveness of online training with real users.

Acknowledgements

This research was partly funded by the EU FP6 TALK Project (http://www.talk-project.org). The system evaluation was conducted in collaboration with Oliver Lemon, Kalliroi Georgila and Jamie Henderson at Edinburgh University and their work is gratefully acknowledged. We would also like to thank the anonymous referees for their helpful comments.

References

B Bonet. 2002. An e-Optimal Grid-based Algorithm for Partially Observable Markov Decision Processes. In *Proceedings of the Nineteenth International Conference on Machine Learning(ICML 2002),*, Sydney, Australia.

RI Brafman. 1997. A Heuristic Variable Grid Solution Method for POMDPs. In *AAAI*, Cambridge, MA.

K. Georgila, J. Henderson, and O. Lemon. 2005. Learning user simulations for information state update dialogue systems. *Proc. of Eurospeech.* Lisbon, Portugal.

LP Kaelbling, ML Littman, and AR Cassandra. 1998. Planning and Acting in Partially Observable Stochastic Domains. *Artificial Intelligence*, 101:99–134.

O. Lemon, K. Georgila, and J. Henderson. 2006. Evaluating Effectiveness and Portability of Reinforcement Learned Dialogue Strategies with real users: the TALK TownInfo Evaluation. In *Proc. of IEEE/ACL SLT*, Palm Beach, Aruba.

E. Levin, R. Pieraccini, and W. Eckert. 2000. A Stochastic Model of Human-Machine Interaction for Learning Dialog Strategies. *IEEE Trans Speech and Audio Processing*, 8(1):11–23.

O. Pietquin and T. Dutoit. 2005. A probabilistic framework for dialog simulation and optimal strategy learning. *IEEE Transactions on Speech and Audio Processing, Special Issue on Data Mining of Speech, Audio and Dialog*.

Joelle Pineau, Geoffrey Gordon, and Sebastian Thrun. 2003. Point-based value iteration: An anytime algorithm for pomdps. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1025 – 1032, August.

V. Rieser and O. Lemon. 2006. Cluster-based User Simulations for Learning Dialogue Strategies. In *Proc. of ICSLP*, Pittsburgh, PA.

Nicholas Roy, Geoffrey Gordon, and Sebastian Thrun. 2005. Finding approximate pomdp solutions through belief compression. *Journal of Artificial Intelligence Research*, 23:1–40.

J. Schatzmann, K. Weilhammer, M.N. Stuttle, and S. Young. 2006. A Survey of Statistical User Simulation Techniques for Reinforcement-Learning of Dialogue Management Strategies. *Knowledge Engineering Review, Cambridge University Press*, 21(2):97–126.

K. Scheffler and S. J. Young. 2002. Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning. In *Proc. of NAACL/HLT.* San Diego, CA.

RS Sutton and AG Barto. 1998. *Reinforcement Learning: An Introduction.* Adaptive Computation and Machine Learning. MIT Press, Cambridge, Mass.

X Wei and AI Rudnicky. 1999. An agenda-based dialog management architecture for spoken language systems. In *Proc. of IEEE ASRU.* Seattle, WA.

JD Williams and SJ Young. 2005. Scaling up POMDPs for Dialogue Management: the Summary POMDP Method. In *IEEE workshop on Automatic Speech Recognition and Understanding (ASRU2005)*, Puerto Rico.

J. Williams and S. Young. 2006. Scaling pomdps for dialog management with composite summary point-based value iteration (cspbvi). In *AAAI Workshop on Statistical and Empirical Approaches for Spoken Dialogue Systems*, Boston.

Jason D. Williams. 2006. *Partially Observable Markov Decision Processes for Spoken Dialogue Management.* Ph.D. thesis, University of Cambridge, April.

SJ Young, JD Williams, J Schatzmann, MN Stuttle, and K Weilhammer. 2005. The hidden information state approach to dialogue management. Technical report, Cambridge Univ. Engineering Dept.

S. Young, J. Schatzmann, K. Weilhammer, and H. Ye. 2007. The Hidden Information State Approach to Dialog Management. In *Proc. of ICASSP*, Honolulu, Hawaii.

S. Young. 2006. Using POMDPs for Dialog Management. In *Proc. of IEEE/ACL SLT*, Palm Beach, Aruba.

The Multimodal Presentation Dashboard

Michael Johnston
AT&T Labs Research
180 Park Ave
Florham Park, NJ
johnston@research.att.com

Patrick Ehlen
CSLI
Stanford University
Palo Alto, CA
ehlen@csli.stanford.edu

David Gibbon
AT&T Labs Research
180 Park Ave
Florham Park, NJ
dcg@research.att.com

Zhu Liu
AT&T Labs Research
180 Park Ave
Florham Park, NJ
zliu@research.att.com

Abstract

The multimodal presentation dashboard allows users to control and browse presentation content such as slides and diagrams through a multimodal interface that supports speech and pen input. In addition to control commands (e.g. "take me to slide 10"), the system allows multimodal search over content collections. For example, if the user says "get me a slide about internet telephony," the system will present a ranked series of candidate slides that they can then select among using voice, pen, or a wireless remote. As presentations are loaded, their content is analyzed and language and understanding models are built dynamically. This approach frees the user from the constraints of linear order allowing for a more dynamic and responsive presentation style.

1 Introduction

Anthropologists have long informed us that the way we work—whether reading, writing, or giving a presentation—is tightly bound to the tools we use. Web browsers and word processors changed the way we read and write from linear to nonlinear activities, though the linear approach to giving a presentation to a roomful of people has evolved little since the days of Mylar sheets and notecards, thanks to presentation software that reinforces—or even further entrenches—a linear bias in our notion of what "giving a presentation" means to us. While today's presentations may be prettier and flashier, the spontaneity once afforded by holding a stack of easily re-arrangeable sheets has been lost.

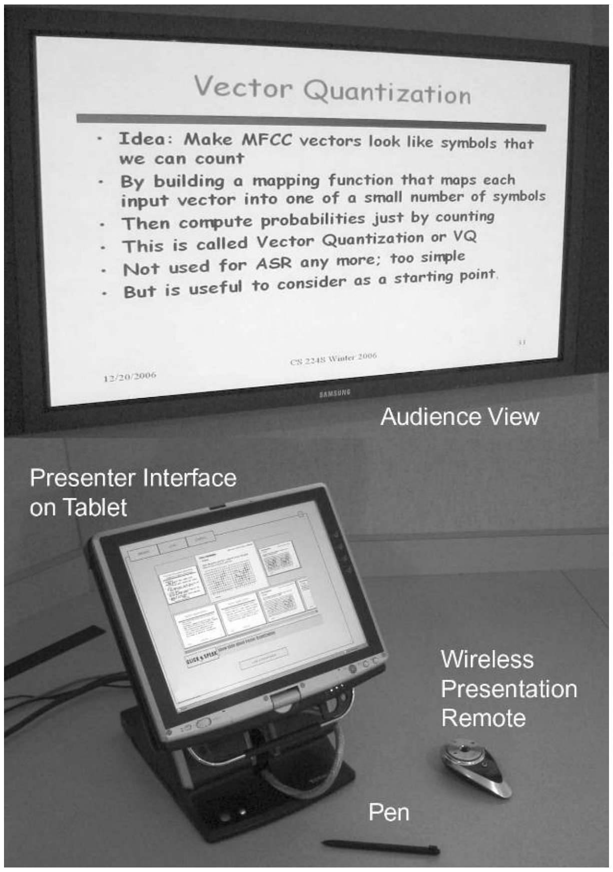

Vector Quantization
- Idea: Make MFCC vectors look like symbols that we can count
- By building a mapping function that maps each input vector into one of a small number of symbols
- Then compute probabilities just by counting
- This is called Vector Quantization or VQ
- Not used for ASR any more; too simple
- But is useful to consider as a starting point.

Figure 1 Presentation dashboard in action

Instead, a question from the audience or a change in plan at the podium results in a whizzing-by of all the wrong slides as the presenter sweats through an awkward silence while hammering an arrow key to track down the right one. In theory there are "search" functions that presenters could use to find another slide in the same presentation, or even in another presentation on the same machine, though none of the authors of this paper has ever seen a presenter do this. A likely reason is that these search functions are designed for desktop ergonomics rather than for standing at a podium or walking around the room, making them even more disruptive to the flow of a presentation than frantic arrow key hammering.

In some utopian future, we envision presenters who are unhindered by limitations imposed by their presentation tools, and who again possess, as Aristotle counseled, “all available means of persuasion” at the tips of their fingers—or their tongues. They enjoy freeform interactions with their audiences, and benefit from random access to their own content with no arrow hammering and no disruption in flow. Their tools help to expand their possible actions rather than limiting them. We are hardly alone in this vision.

In that spirit, many tools have been developed of late—both within and outside of research labs—with the aim of helping people work more effectively when they are involved in those assemblies of minds of mutual interest we often call “meetings.” Tools that capture the content of meetings, perform semantic understanding, and provide a browsable summary promise to free meeting participants from the cognitive constraints of worrying about trying to record and recall what happened when a meeting takes place (e.g., Ehlen, Purver & Niekrasz, 2007; Tucker & Whittaker, 2005).

Presentations are a kind of meeting, and several presentation tools have also sought to free presenters from similar constraints. For example, many off-the-shelf products provide speech interfaces to presentation software. These often replace the linear arrow key with the voice, offering command-based navigation along a one-dimensional vector of slides by allowing a presenter to say “next slide please” or “go to the last slide.”

A notable exception is the Jabberwocky interface to PowerPoint (Franklin, Bradshaw & Hammond, 1999; 2000), which aims to follow along with a presenter’s talk—like a human assistant might do—and switch to the appropriate slide when the presenter seems to be talking about it. Using a method similar to topic modeling, words spoken by the presenter are compared to a probability distribution of words across slides. Jabberwocky changes to a different slide when a sufficient probability mass has been reached to justify the assumption that the speaker is now talking about a different slide from the one that’s already showing.

A similar effort (Rogina & Schaaf, 2002) uses words extracted from a presentation to augment a class-based language model and attempt automatic tracking of a presentation as it takes place. This intelligent meeting room system then aligns the presenter’s spoken words with parts of a presentation, hoping to determine when a presenter has moved on to a new slide.

A major drawback of this “machine-initiative” approach to presentation assistance is that a presenter must speak enough words associated with a new slide for a sufficient probability mass to be reached before the slide is changed. The resulting delay is likely to make an audience feel like the presentation assistant is rather dim-witted. And any errors that change slides before the presenter is ready can be embarrassing and disruptive in front of potentially important audiences.

So, in fashioning our own presentation control interface, we chose to allow the presenter to retain full initiative in changing slides, while offering a smarter and more flexible way to navigate through a presentation than the single degree of freedom afforded by arrow keys that simply traverse a predetermined order. The result is the Multimodal Presentation Dashboard, a presentation interface that integrates command-based control with probabilistic, content-based search. Our method starts with a context-free grammar of speech commands, but embeds a stochastic language model generated from the presenter’s slide deck content so a presenter can request any slide from the deck—or even a large set of decks—just by asking for its contents. Potentially ambiguous results are resolved multimodally, as we will explain.

2 Multimodal interface for interactive presentations

The presentation dashboard provides presenters with the ability to control and adapt their presentations on the fly in the meeting room. In addition to the traditional next/previous approach to navigating a deck of slides, they can access slides by position in the active deck (e.g., “show slide 10” or “last slide please”) or they can multimodally combine voice commands with pen or remote control to browse for slides by content, saying, for instance, “show the slide on internet telephony,” and then using the pen to select among a ranked list of alternatives.

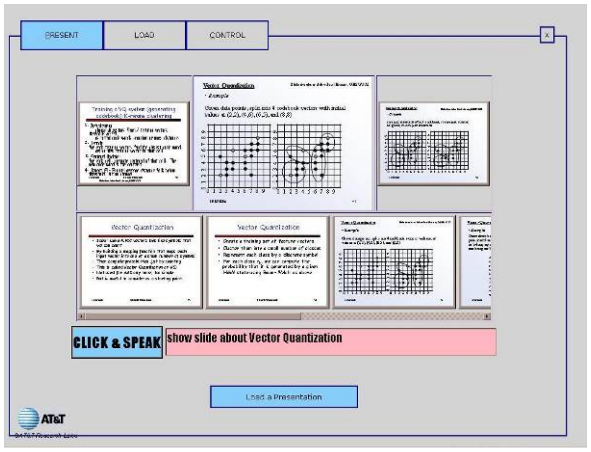
2.1 Setup configuration

Though the dashboard offers many setup configurations, the preferred arrangement uses a single PC with two displays (Figure 1). Here, the dashboard is running on a tablet PC with a large monitor as a second external display. On the tablet, the dashboard UI is visible only to the presenter. On the external display, the audience sees the current slide, as they would with a normal presentation.

The presenter can interact with the dashboard using either the microphone onboard the tablet PC, or, preferably, a wireless microphone. A wireless remote functions as a presentation control, which can be used to manually change slides in the traditional manner, and also provides a "push to talk" button to tell the dashboard when to listen. A wireless microphone combined with the wireless presentation control and voice selection mode (see Section 2.3) allows a presenter to stroll around the room or stage completely untethered.

2.2 Presenter UI

The presenter's primary control of the system is through the presenter UI, a graphical user interface augmented with speech and pen input. The interface has three main screens: a presentation panel for controlling an ongoing presentation (Figure 2), a loader panel for selecting a set of presentations to load (Figure 4), and a control panel for adjusting system settings and bundling shareable index and grammar models. The user can select among the panels using the tabs at the top left.



show slide about Vector Quantization

Figure 2 The presentation panel

The presentation panel has three distinct functional areas from top to bottom. The first row shows the current slide, along with thumbnails of the previous and next slides to provide context. The user can navigate to the next or previous slide by clicking on these thumbnails. The next row shows a scrolling list of search results from content-based queries. The last row contains interaction information. There is a *click & speak* button for activating the speech recognizer and a feedback window that displays recognized speech.

Some user commands are independent of the content of slide decks, as with basic commands for slide navigation:

- "next slide please"
- "go back"
- "last slide"

In practice, however, navigation to next and previous slides is much easier using buttons on the wireless control. The presenter can also ask for slides by position number, allowing random access:

- "take me to slide 10"
- "slide 4 please"

But not many presenters can remember the position numbers of some 40 or 50 slides, we'd guess, so we added content-based search, a better method of random access slide retrieval by simply saying key words or phrases from the desired slide, e.g.:

- "slides about internet telephony"
- "get me the slide with the system architecture"
- "2006 highlights"
- "budget plan, please"

When the presenter gives this kind of request, the system identifies any slides that match the query and displays them in a rank ordered list in the middle row of the presenter's panel. The presenter can then scroll through the list of thumbnails and click one to display it to the audience.

This method of ambiguity resolution offers the presenter some discretion in selecting the correct slide to display from multiple search results, since search results appear first on the presenter's private interface rather than being displayed to the audience. However, it requires the presenter to return to the podium (or wherever the tablet is located) to select the correct slide.

2.3 Voice selection mode

Alternatively, the presenter may sacrifice discretion for mobility and use a “voice selection mode,” which lets the presenter roam freely throughout the auditorium while making and resolving content-based queries in plain view of the audience. In this mode, if a presenter issues a content-based query (e.g., “shows slides about multimodal access”), thumbnails of the slides returned by the query appear as a dynamically-generated interactive “chooser” slide (Figure 3) in the main presentation viewed by the audience. The presenter can then select the desired slide by voice (e.g., “slide three”) or by using the previous, next, and select controls on the wireless remote. If more than six slides are returned by the query, multiple chooser slides are generated with six thumbnails to each slide, which can be navigated with the remote.

While voice selection mode allows the presenter greater mobility, it has the drawback of allowing the audience to see thumbnails of every slide returned by a content-based query, regardless of whether the presenter intended for them to be seen. Hence this mode is more risky, but also more impressive!

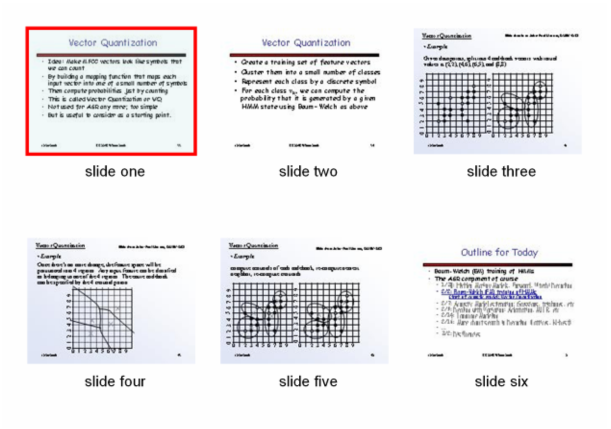


Figure 3 Chooser slide for voice selection mode

2.4 Compiling deck sets

Sometimes a presenter wishes to have access to more than one presentation deck at a time, in order to respond to unexpected questions or comments, or to indulge in a whimsical tangent. We respond to this wish by allowing the presenter to compile a *deck set*, which is, quite simply, a user-defined bundle of multiple presentations that can all be searched at once, with their slides available for display when the user issues a query. In fact, this option makes it easy for a presenter to follow spontaneous tangents by switching from one presentation to another, navigating through the alternate deck for a while, and then returning to the original presentation, all without ever walking to the podium or disrupting the flow of a presentation by stopping and searching through files.

Deck sets are compiled in the loader panel (Figure 4), which provides a graphical browser for selecting a set of active decks from the file system. When a deck set is chosen, the system builds ASR and language understanding models and a retrieval index for all the slides in the deck set. A compiled deck set is also portable, with all of the grammar and understanding model files stored in a single archive that can be transferred via e-mail or thumb drive and speedily loaded on another machine.

A common use of deck sets is to combine a main presentation with a series of other slide decks that provide background information and detail for answering questions and expanding points, so the presenter can adapt to the interests of the audience.

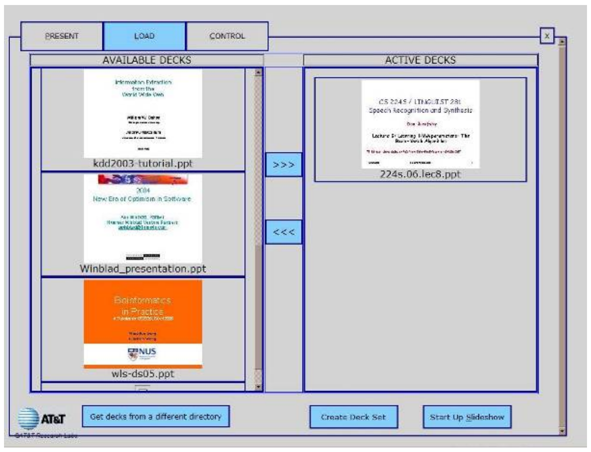


Figure 4 The loader panel

3 Multimodal architecture

The Multimodal Presentation Dashboard uses an underlying multimodal architecture that inherits core components from the MATCH architecture (Johnston et al 2002). The components communicate through a central messaging facilitator and include a speech recognition client, speech recognition server (Goffin et al 2005), a natural language understanding component (Johnston & Bangalore 2005), an information retrieval engine, and a graphical user interface client. The graphical UI runs in a web browser and controls PowerPoint via its COM interface.

We first describe the compilation architecture, which builds models and performs indexing when the user selects a series of decks to activate. We then describe the runtime architecture that operates when the user gives a presentation using the system. In Section 3.3, we provide more detail on the slide indexing mechanism and in Section 3.4 we describe a mechanism used to determine keyphrases from the slide deck that are used on a drop down menu and for determining relevancy.

3.1 Compilation architecture

In a sense, the presentation dashboard uses neither static nor dynamic grammars; the grammars compiled with each deck set lie somewhere in-between those two concepts. Command-based speech interfaces often fare best when they rely on the predictability of a fixed, context-free grammar, while interfaces that require broader vocabulary coverage and a wider range of syntax are better off leveraging the flexibility of stochastic language models. To get the best of both worlds for our ASR model, we use a context-free command "wrapper" to a stochastic language model (c.f. Wang & Acero 2003). This is coupled to the understanding mechanism using a transducer with a loop over the content words extracted from the slides.

This *combined grammar* is best thought of as a fixed, context-free template which contains an embedded SLM of dynamic slide contents. Our method allows a static background grammar and understanding model to happily co-exist with a dynamic grammar component which is compiled on the fly when presentations are loaded, enabling custom, content-based queries.

When a user designates a presentation deck set and compiles it, the slides in the set are processed to create the combined grammar by composing an SLM training corpus based on the slide content.

First, a slide preprocessor extracts sentences, titles, and captions from each slide of each deck, and normalizes the text by converting numerals and symbols to strings, Unicode to ASCII, etc. These *content phrases* are then used to compose (1) a combined corpus to use for training an SLM for speech recognition, and (2) a finite-state transducer to use for multimodal natural language understanding (Johnston & Bangalore 2005).

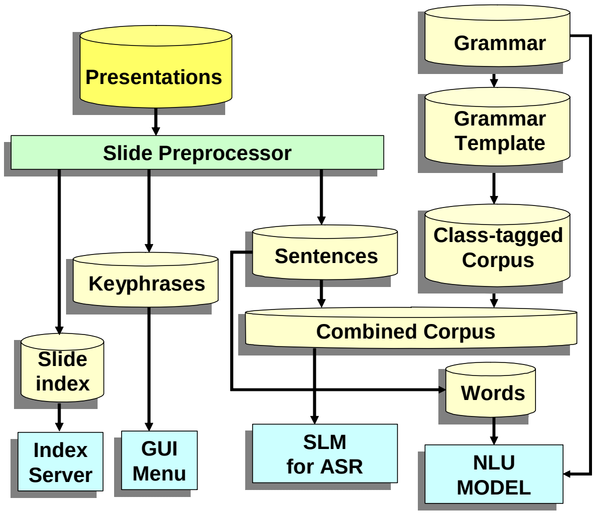


Figure 5 Compilation architecture

To create a combined corpus for the SLM, the content phrases extracted from slides are iterated over and folded into a static template of corpus classes. For instance, the template entry,

```
<POLITE> <SHOWCON> <CONTENT_PHRASE>
```

could generate the phrase "*please show the slide about* `<CONTENT_PHRASE>`" for each content phrase—as well as many others. These templates are currently manually written but could potentially be induced from data as it becomes available.

The content corpus is appended to a command corpus of static command classes that generate phrases like "next slide please" or "go back to the last one." Since the number of these command phrases remains constant for every grammar while the number of content phrases depends on how many phrases are extracted from the deck set, a weighting factor is needed to ensure the number of examples of both content and command phrases is balanced in the SLM training data. The resulting combined corpus is used to build a stochastic language model that can handle variations on commands and slide content.

In parallel to the combined corpus, a stack of slide *content words* is compiled for the finite state understanding machine. Phrases extracted for the combined corpus are represented as a terminal `_CWORD` class. (Terminals for tapes in each grammar class are separated by colons, in the format `speech:meaning`, with empty transitions represented as ε) For example, the phrase “internet telephony” on a slide would appear in the understanding grammar like so:

```
_CWORD internet:internet
_CWORD telephony:telephony
```

These content word classes are then “looped” in the FSM (Figure 6) into a flexible understanding model of potential slide content results using only a few grammar rules, like:

```
_CONTENT _CWORD _CONTENT
_CONTENT _CWORD
```

The SLM and the finite-state understanding machine now work together to extract plausible meanings from dynamic and inexact speech queries.

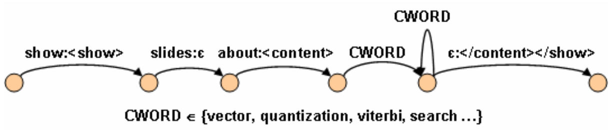


Figure 6 Understanding FSM

To provide an example of how this combined approach to understanding comes together in the running system, let’s say a presenter’s slide contains the title “Report for Third Quarter” and she asks for it by saying, “put up the third quarter report slide.” Though she asks for the slide with language that doesn’t match the phrase on the slide, our forgiving stochastic model might return a speech result like, “put up third quarter report mine.” The speech result is then mapped to the finite-state grammar, which catches “third quarter report mine” as a possible content phrase, and returns, “`third,quarter,report,mine`” as a content-based meaning result. That result is then used for information retrieval and ranking to determine which slides best match the query (Section 3.3).

3.2 Runtime architecture

A primary goal of the presentation dashboard was that it should run standalone on a single laptop. A tablet PC works best for selecting slides with a pen, though a mouse or touch screen can also be used for input. We also developed a networked version of the dashboard system where indexing, compilation, speech recognition, and understanding are all network services accessed over HTTP and SIP, so any web browser-based client can log in, upload a presentation, and present without installing software aside from PowerPoint and a SIP plug-in. However, our focus in this paper is on the tablet PC standalone version.

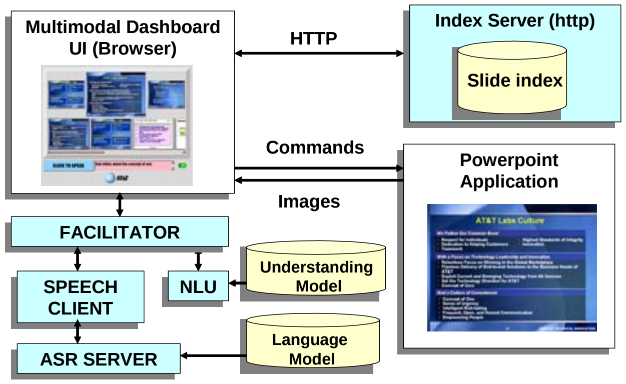


Figure 7 Multimodal architecture

The multimodal user interface client is browser-based, using dynamic HTML and Javascript. Internet Explorer provides COM access to the PowerPoint object model, which reveals slide content and controls the presentation. Speech recognition, understanding, and compilation components are accessed through a java-based facilitator via a socket connection provided by an ActiveX control on the client page (Figure 7). When the user presses or taps the *click & speak* button, a message is sent to the Speech client, which sends audio to the ASR Server. The recognizer’s speech result is processed by the NLU component using a finite-state transducer to translate from the input string to an XML meaning representation. When the multimodal UI receives XML for simple commands like “first slide” or “take me to slide ten,” it calls the appropriate function through the PowerPoint API. For content-based search commands, an SQL query is constructed and issued to the index server as an HTTP query. When the results are returned, multimodal thumbnail images of each slide appear in the middle row of the UI presenter panel. The user can then review the choices and switch to the appropriate slide by clicking on it—or, in voice selection mode, by announcing or selecting a slide shown in the dynamically-generated chooser slide.

The system uses a three stage strategy in searching for slides. First it attempts an exact match by looking for slides which have the words of the query in the same order on the same slide in a single phrase. If no exact matches are found, the system backs off to an AND query and shows slides which contain all of the words, in any order. If that fails, the system resorts to an OR query and shows slides which have any of the query terms.

3.3 Information retrieval

When the slide preprocessor extracts text from a presentation, it retains the document structure as much as possible and stores this in a set of hierarchal XML documents. The structure includes global document metadata such as creation date and title, as well as more detailed data such as slide titles. It also includes information about whether the text was part of a bullet list or text box. With this structure, queries can be executed against the entire text or against specified textual attributes (e.g. “show me the chart titled ‘project budget’”).

For small document collections, XPath queries can search the entire collection with good response time, providing a stateless search method. But as the collection of presentation decks to be searched grows, a traditional inverted index information retrieval system achieves better response times. We use a full text retrieval system that employs stemming, proximity search, and term weighting, and supports either a simplified query syntax or SQL. Global metadata can also constrain queries. Incremental indexing ensures that new presentation decks cause the index to update automatically without being rebuilt from scratch.

3.4 Key phrase extraction

Key phrases and keywords are widely used for indexing and retrieving documents in large databases. For presentation slides, they can also help rank a slide’s relevance to a query. We extract a list of key phrases with importance scores for each slide deck, and phrases from a set of decks are merged and ranked based on their scores.

A popular approach to selecting keywords from a document within a corpus is to find keywords that frequently occur in one document but seldom occur in others, based on term frequency-inverse document frequency (TF-IDF). Our task is slightly different, since we wish to choose key phrases for a single document (the slide deck), independent of other documents. So our approach uses term frequency-inverse term probability (TF-ITP), which expresses the probability of a term calculated over a general language rather than a set of documents.

Assuming a term T_k occurs tf_k times in a document, and its term probability is tp_k, the TF-ITP of T_k is defined as, $w_{Tk} = tf_k / tp_k$. This method can be extended to assign an importance score to each phrase. For a phrase $F_k = \{T_1\ T_2\ T_3 \ldots T_N\}$, which contains a sequence of N terms, assuming it appears ff_k times in a document, its importance score, IS_k, is defined as,

$$IS_k = \sum_{i=1}^{N} \frac{ff_k}{T_i}.$$

To extract a set of key phrases, we first segment the document into sentences based on punctuation and some heuristics. A Porter stemming algorithm (Porter 1980) eliminates word variations, and phrases up to N=4 terms long are extracted, removing any that start or end with noise words. An importance score ranks each phrase, where term probabilities are estimated from transcripts of 600 hours of broadcast news data. A term that is out of the vocabulary with a term frequency of more than 2 is given a default term probability value, defined as the minimum term probability in the vocabulary. Phrases with high scores are chosen as key phrases, eliminating any phrases that are contained in other phrases with higher scores. For an overall list of key phrases in a set of documents, we merge individual key phrase lists and sum the importance scores for key phrases that recur in different lists, keeping the top 10 phrases.

4 Performance and future work

The dashboard is fully implemented, and has been used by staff and management in our lab for internal presentations and talks. It can handle large decks and collections (100s to 1000s of slides). A tablet PC with a Pentium M 1.6Ghz processor and 1GB of RAM will compile a presentation of 50 slides—with ASR, understanding models, and slide index—in under 30 seconds.

In ongoing work, we are conducting a usability test of the system with users in the lab. Effective evaluation of a tool of this kind is difficult without fielding the system to a large number of users. An ideal evaluation would measure how users fare when giving their own presentations, responding to natural changes in narrative flow and audience questions. Such interaction is difficult to simulate in a lab, and remains an active area of research.

We also hope to extend current retrieval methods to operate at the level of concepts, rather than words and phrases, so a request to show "slides about mortgages" might return a slide titled "home loans." Thesauri, gazetteers, and lexicons like WordNet will help achieve this. Analyzing non-textual elements like tables and charts could also allow a user to say, "get the slide with the network architecture diagram." And, while we now use a fixed lexicon of common abbreviations, an automated analysis based on web search and other techniques could identify likely expansions.

5 Conclusion

Our goal with the multimodal presentation dashboard was to create a meeting/presentation assistance tool that would change how people behave, inspiring presenters to expand the methods they use to interact with audiences and with their own material. To this end, our dashboard runs on a single laptop, leaves the initiative in the hands of the presenter, and allows slides from multiple presentations to be dynamically retrieved from anywhere in the room. Our assistant requires no "intelligent room"; only an intelligent presenter, who may now offer the audience a presentation that is as dynamic or as dull as imagination allows.

As Tufte (2006) reminds us in his analysis of how PowerPoint presentations may have precipitated the Columbia shuttle tragedy, the way information is presented can have a profound—even life-threatening—impact on the decisions we make. With the multimodal presentation dashboard, we hope to free future presenters from that single, arrow-key dimension, offering access to presentation slides and diagrams in any order, using a diverse combination of modes. Presenters can now pay more attention to the needs of their audiences than to the rigid determinism of a fixed presentation. Whether they will break free of the linear presentation style imposed by current technology if given a chance remains to be seen.

References

Patrick Ehlen, Matthew Purver, and John Niekrasz. 2007. A meeting browser that learns. In *Proceedings of the AAAI Spring Symposium on Interaction Challenges for Intelligent Assistants*.

David Franklin, Shannon Bradshaw, and Kristian Hammond. 1999. Beyond "Next slide, please": The use of content and speech in multi-modal control. In *Working Notes of the AAAI-99 Workshop on Intelligent Information Systems*.

David Franklin, Shannon Bradshaw, and Kristian Hammond. 2000. Jabberwocky: You don't have to be a rocket scientist to change slides for a hydrogen combustion lecture. In *Proceedings of Intelligent User Interfaces 2000 (IUI-2000)*.

Vincent Goffin, Cyril Allauzen, Enrico Bocchieri, Dilek Hakkani-Tür, Andrej Ljolje, Sarangarajan Parthasarathy, Mazin Rahim, Giuseppe Riccardi, and Murat Saraclar. 2005. The AT&T WATSON speech recognizer. In *Proceedings of ICASSP*.

Michael Johnston, Srinivas Bangalore, Guna Vasireddy, Amanda Stent, Patrick Ehlen, Marilyn Walker, Steve Whittaker, Preetam Maloor. 2002. MATCH: An Architecture for Multimodal Dialogue Systems. In *Proceedings of the 40th ACL*. 376-383.

Michael Johnston and Srinivas Bangalore. 2005. Finite-state multimodal integration and understanding. *Journal of Natural Language Engineering*. 11.2, pp. 159-187, Cambridge University Press.

Martin F. Porter. 1980. An algorithm for suffix stripping, *Program, 14*, 130-137.

Ivica Rogina and Thomas Schaaf. 2002. Lecture and presentation tracking in an intelligent meeting room. In *Proceedings of the 4th IEEE International Conference on Multimodal Interfaces*. 47-52.

Simon Tucker and Steve Whittaker. 2005. Accessing multimodal meeting data: Systems, problems and possibilities. In Samy Bengio and Hervé Bourlard (Eds.) *Lecture Notes in Computer Science*, 3361, 1-11

Edward Tufte. 2006. *The Cognitive Style of PowerPoint*. Graphics Press, Cheshire, CT.

Ye-Yi Wang and Alex Acero. 2003. Combination of CFG and N-gram Modeling in Semantic Grammar Learning. *Proceedings of Eurospeech conference*, Geneva, Switzerland.

Acknowledgements We would like to thank Srinivas Bangalore, Rich Cox, Mazin Gilbert, Vincent Goffin, and Behzad Shahraray for their help and support.

Technical Support Dialog Systems: Issues, Problems, and Solutions

Kate Acomb, Jonathan Bloom, Krishna Dayanidhi, Phillip Hunter, Peter Krogh, Esther Levin, Roberto Pieraccini
SpeechCycle
535 W 34th Street
New York, NY 10001
{kate,jonathanb,krishna,phillip,peter,roberto}@speechcycle.com
esther@spacegate.com

Abstract

The goal of this paper is to give a description of the state of the art, the issues, the problems, and the solutions related to industrial dialog systems for the automation of technical support. After a general description of the evolution of the spoken dialog industry, and the challenges in the development of technical support applications, we will discuss two specific problems through a series of experimental results. The first problem is the identification of the call reason, or *symptom*, from loosely constrained user utterances. The second is the use of data for the experimental optimization of the Voice User Interface (VUI).

1 Introduction

Since the beginning of the telephony spoken dialog industry, in the mid 1990, we have been witnessing the evolution of at least three generations of systems. What differentiates each generation is not only the increase of complexity, but also the different architectures used. Table 1 provides a summary of the features that distinguish each generation. The early first generation systems were mostly informational, in that they would require some information from the user, and would provide information in return. Examples of those systems, mostly developed during the mid and late 1990s, are package tracking, simple financial applications, and flight status information. At the time there were no standards for developing dialog systems, (VoiceXML 1.0 was published as a recommendation in year 2000) and thus the first generation dialog applications were implemented on proprietary platforms, typically evolutions of existing touch-tone IVR (Interactive Voice Response) architectures.

Since the early developments, spoken dialog systems were implemented as a graph, called *call-flow*. The nodes of the call-flow typically represent actions performed by the system and the arcs represent an enumeration of the possible outcomes. Playing a prompt and interpreting the user response through a speech recognition grammar is a typical action. Dialog modules (Barnard et al., 1999) were introduced in order to reduce the complexity and increase reusability of call-flows. A Dialog Module (or DM) is defined as a call-flow object that encapsulates many of the interactions needed for getting one piece of information from the user, including retries, timeout handling, disambiguation, etc. Modern commercial dialog systems use DMs as their active call-flow nodes.

The number of DMs in a call-flow is generally an indication of the application complexity. First generation applications showed a range of complexity of a few to tens of DMs, typically spanning a few turns of interaction.

The dialog modality is another characterization of applications. Early applications supported strict directed dialog interaction, meaning that at each

	GENERATION		
	FIRST	**SECOND**	**THIRD**
Time Period	1994-2001	2000-2005	2004-today
Type of Application	Informational	Transactional	Problem Solving
Examples	Package Tracking, Flight Status	Banking, Stock Trading, Train Reservation	Customer Care, Technical Support, Help Desk.
Architecture	Proprietary	Static VoiceXML	Dynamic VoiceXML
Complexity (Number of DMs)	10	100	1000
Interaction Turns	few	10	10-100
Interaction Modality	directed	directed + natural language (SSLU)	directed + natural language (SSLU) + limited mixed initiative

Table 1: Evolution of spoken dialog systems.

turn the system would *direct* the user by proposing a finite—and typically small—number of choices. That would also result in a limited grammar or vocabulary at each turn.

The applications of the second generation were typically transactional, in the sense that they could perform a transaction on behalf of the user, like moving funds between bank accounts, trading stocks, or buying tickets. Most of those applications were developed using the new standards, typically as collections of VoiceXML documents. The complexity moved to the range of dozens of dialog modules, spanning a number of turns of interactions of the order of ten or more. At the same time, some of the applications started using a technology known as Statistical Spoken Language Understanding, or SSLU (Gorin et al., 1997, Chu-Carroll et al., 1999, Goel et al, 2005), for mapping loosely constrained user utterances to a finite number of pre-defined semantic categories. The *natural language* modality—as opposed to directed dialog—was initially used mostly for call-routing, i.e. to route calls to the appropriate call center based on a more or less lengthy description of the reason for the call by the user.

While the model behind the first and second generations of dialog applications can be described by the form-filling paradigm, and the interaction follows a pre-determined simple script, the systems of the third generation have raised to a qualitatively different level of complexity. Problem solving applications, like customer care, help desk, and technical support, are characterized by a level of complexity ranging in the thousands of DMs, for a number of turns of dynamic interaction that can reach into the dozens. As the sophistication of the applications evolved, so did the system architecture by moving the logic from the client (VoiceXML browser, or voice-browser) to the server (Pieraccini and Huerta, 2005). More and more system are today based on generic dialog application server which interprets a dialog specification described by a—typically proprietary—markup language and serve the voice-browser with dynamically generated VoiceXML documents. Finally, the interaction modality of the third generation systems is moving from the strictly directed dialog application, to directed dialog, with some natural language (SSLU) turns, and some limited mixed-initiative (i.e. the possibility for the user to change the course of the dialog by making an unsolicited request).

2 Technical Support Applications

Today, automated technical support systems are among the most complex types of dialog applications. The advantage of automation is clear, especially for high-volume services like broadband-internet, entertainment (cable or satellite TV), and telephony. When something goes wrong with the service, the only choice for subscribers is to call a technical support center. Unfortunately, staffing a call center with enough agents trained to help solve even the most common problems results in prohibitive costs for the provider, even when outsourcing to less costly locations End users often experience long waiting times and poor service from untrained agents. With the magnitude of the daily increase in the number of subscribers of those services, the situation with human agents is bound to worsen. Automation and self-service can, and does, help reduce the burden constituted by the most frequent call reasons, and resort to human agents only for the most difficult and less common problems.

However, automating technical support is particularly challenging for several reasons. Among them:

- Troubleshooting knowledge is not readily available in a form that can be used for automation. Most often it is based on the idiosyncratic experience of the individual agents.
- End users are typically in a somewhat emotionally altered state—something for which they paid and that is supposed to work is broken. They want it repaired quickly by an expert human agent; they don't trust a machine can help them.
- The description of the problem provided by the user can be imprecise, vague, or based on a model of the world that may be incorrect (e.g. some users of internet cannot tell their modem from their router).
- It may be difficult to instruct non-technically savvy users on how to perform a troubleshooting step (e.g. *Now renew your IP address*.) or request technical information (e.g. *Are you using a Voice over IP phone service?*)
- Certain events cannot be controlled. For instance, the time it would take for a user to complete a troubleshooting step, like rebooting a PC, is often unpredictable.
- The acoustic environment may be challenging. Users may be asked to switch their TV on, reboot their PC, or check the cable connections. All these operations can cause noise that can trigger the speech recognizer and affect the course of the interaction.

On the other hand, one can leverage the automated diagnosis or troubleshooting tools that are currently used by human agent and improve the efficiency of the interaction. For instance, if the IP address of the digital devices at the user premises is available, one can ping them, verify their connectivity, download new firmware, and perform automated troubleshooting steps in the background without the intervention of the user. However, the interplay between automated and interactive operations can raise the complexity of the applications such as to require higher level development abstractions and authoring tools.

3 High Resolution SSLU

The identification of the call reason—i.e. the problem or the symptoms of the problem experienced by the caller—is one of the first phases of the interaction in a technical support application. There are two possible design choices with today's spoken language technology:

- **Directed dialog**. A specific prompt enumerates all the possible reasons for a call, and the user would choose one of them.
- **Natural Language:** An open prompt asks the user to describe the reason for the call. The utterance will be automatically mapped to one of a number of possible call reasons using SSLU technology.

Directed dialog would be the preferred choice in terms of accuracy and cost of development. Unfortunately, in most technical support applications, the number of call-reasons can be very large, and thus prompting the caller through a directed dialog menu would be impractical. Besides, even though a long menu can be structured hierarchically as a cascade of several shorter menus, the terms used for indicating the different choices may be misleading or meaningless for some of the users (e.g. *do you have a problem with hardware, software, or networking?*). Natural language with SSLU is generally the best choice for problem identification.

In practice, users mostly don't know what the actual problem with their service is (e.g. *modem is wrongly configured*), but typically they describe their observations—or *symptoms*—which are observable manifestations of the problem. and not the problem itself (e.g. symptom: *I can't connect to the Web*, problem: modem wrongly configured). Correctly identifying the symptom expressed in natural language by users is the goal of the SSLU module.

SSLU provides a mapping between input utterances and a set of pre-determined categories. SSLU has been effectively used in the past to enable automatic call-routing. Typically call-routing applications have a number of categories, of the order of a dozen or so, which are designed based on the different *routes* to which the IVR is supposed to dispatch the callers. So, generally, in call-routing applications, the categories are known and determined prior to any data collection.

One could follow the same approach for the problem identification SSLU, i.e. determine a number of a-priori *problem* categories and then map a collection of training *symptom* utterances to each one of them. There are several issues with this approach.

First, a complete set of categories—the problems—may not be known prior to the acquisition and analysis of a significant number of utterances. Often the introduction of new home devices or services (such as DVR, or HDTV) creates new problems and new symptoms that can be discovered only by analyzing large amounts of utterance data.

Then, as we noted above, the relationship between the problems—or broad categories of problems—and the manifestations (i.e. the symptoms) may not be obvious to the caller. Thus, confirming a broad category in response to a detailed symptom utterance may induce the user to deny it or to give a verbose response (e.g. Caller: *I cannot get to the Web.* System: *I understand you have a problem with your modem configuration, is that right?* Caller: *Hmm…no. I said I cannot get to the Web.*).

Finally, caller descriptions have different degrees of specificity (e.g. *I have a problem with my cable service* vs. *The picture on my TV is pixilated on all channels*). Thus, the categories should reflect a hierarchy of symptoms, from vague to specific, that need to be taken into proper account in the design of the interaction.

As a result from the above considerations, SSLU for symptom identification needs to be designed in order to reflect the *high-resolution* multitude and specificity hierarchy of symptoms that emerge from the analysis of a large quantity of utterances. Figure 1 shows an excerpt from the hierarchy of symptoms for a cable TV troubleshooting application derived from the analysis of almost 100,000 utterance transcriptions.

Each node of the tree partially represented by Figure 1 is associated with a number of training utterances from users describing that particular symptom in their own words. For instance the topmost node of the hierarchy, "TV Problem", corresponds to vague utterances such as *I have a problem with my TV* or *My cable TV does not work*. The" Ordering" node represents requests of the type *I have a problem with ordering a show*, which is still a somewhat vague request, since one can order "Pay-per-view" or "On-demand" events, and they correspond to different processes and troubleshooting steps. Finally, at the most detailed level of the hierarchy, for instance for the node "TV Problem-Ordering-On Demand-Error", one finds utterances such as *I tried to order a movie on demand, but all I get is an error code on the TV.*

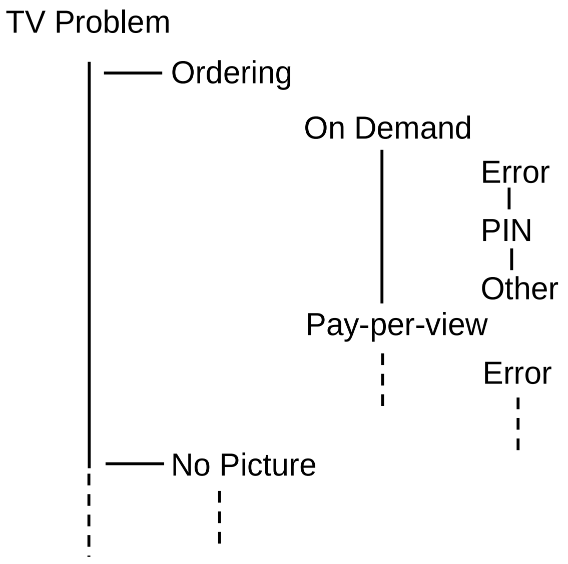


Figure 1: Excerpt from the hierarchical symptom description in a cable TV technical support application

In the experimental results reported below, we trained and tested a hierarchically structured SSLU for a cable TV troubleshooting application. A corpus of 97,236 utterances was collected from a deployed application which used a simpler, non hierarchical, version of the SSLU. The utterances were transcribed and initially annotated based on an initial set of symptoms. The annotation was carried out by creating an *annotation guide* document which includes, for each symptom, a detailed verbal description, a few utterance examples, and relevant keywords. Human annotators were instructed to label each utterance with the correct category based on the annotation guide and their work was monitored systematically by the system designer.

After a first initial annotation of the whole corpus, the annotation consistency was measured by computing a cluster similarity distance between the utterances corresponding to all possible pairs of symptoms. When the consistency between a pair of symptoms was below a given threshold, the clusters were analyzed, and actions taken by the designer in order to improve the consistency, including reassign utterances and, if necessary, modifying the annotation guide. The whole process was repeated a few times until a satisfactory global inter-cluster distance was attained.

Eventually we trained the SSLU on 79 symptoms arranged on a hierarchy with a maximum depth of 3. Table 2 summarizes the results on an independent test set of 10,332 utterances. The result shows that at the end of the process, a satisfactory batch accuracy of 81.43% correct label assignment what attained for the utterances which were deemed to be in-domain, which constituted 90.22% of the test corpus. Also, the system was able to correctly reject 24.56% of out-of-domain utterances. The overall accuracy of the system was considered reasonable for the state of the art of commercial SSLUs based on current statistical classification algorithms. Improvement in the classification performance can result by better language models (i.e. some of the errors are due to incorrect word recognition by the ASR) and better classifiers, which need to take into account more features of the incoming utterances, such as word order[1] and contextual information.

Utterances	10332	100.00%
In domain	9322	90.22%
Correct in-domain	7591	81.43%
Out of domain	1010	9.78%
Correct rejection out-of-domain	249	24.65%

Table 2: Accuracy results for Hierarchical SSLU with 79 symptoms.

[1] Current commercial SSLU modules, as the one used in the work described here, use statistical classifiers based only on bags of words. Thus the order of the words in the incoming utterance is not taken into consideration.

3.1 Confirmation Effectiveness

Accuracy is not the only measure to provide an assessment of how the symptom described by the caller is effectively captured. Since the user response needs to be confirmed based on the interpretation returned by the SSLU, the caller always has the choice of accepting or denying the hypothesis. If the confirmation prompts are not properly designed, the user can erroneously deny correctly detected symptoms, or erroneously accept wrong ones.

The analysis reported below was carried out for a deployed system for technical support of Internet service. The full symptom identification interactions following the initial open prompt was transcribed and annotated for 895 calls. The SSLU used in this application consisted of 36 symptoms structured in a hierarchy with a maximum depth of 3. For each interaction we tracked the following events:

- the first user response to the open question
- successive responses in case of re-prompting because of speech recognition rejection or timeout
- response to the yes/no confirmation question)
- successive responses to the confirmation question in case the recognizer rejected it or timed out.
- Successive responses to the confirmation question in case the user denied, and a second best hypothesis was offered.

Table 3 summarizes the results of this analysis.

The first row reports the number of calls for which the identified symptom was correct (as compared with human annotation) and confirmed by the caller. The following rows are the number of calls where the identified symptom was wrong and the caller still accepted it during confirmation, the symptom was correct and the caller denied it, and the symptom was wrong and denied, respectively. Finally there were 57 calls where the caller did not provide any confirmation (e.g. hung up, timed out, ASR rejected the confirmation utterance even after re-prompting, etc.), and 100 calls in which it was not possible to collect the symptom (e.g. rejections

Accepted correct	535	59.8%
Accepted wrong	118	13.2%
Denied correct	22	2.5%
Denied wrong	63	7.0%
Unconfirmed	57	6.4%
No result	100	11.2%
TOTAL	**895**	**100.0%**

Table 3: Result of the confirmation analysis based on the results of 895 calls

of first and second re-prompts, timeouts, etc.) In both cases—i.e. no confirmation or no symptom collection at all—the call continued with a different strategy (e.g. moved to a directed dialog, or escalated the call to a human agent). The interesting result from this experiment is that the SSLU returned a correct symptom 59.8 + 2.5 = 62.3% of the times (considering both in-domain and out-of-domain utterances), but the actual "perceived" accuracy (i.e. when the user accepted the result) was higher, and precisely 59.8 + 13.2 = 73%. A deeper analysis shows that for most of the wrongly accepted utterances the wrong symptom identified by the SSLU was still in the same hierarchical category, but with different degree of specificity (e.g. Internet-Slow vs. vague Internet)

The difference between the actual and perceived accuracy of SSLU has implications for the overall performance of the application. One could build a high performance SSLU, but a wrongly confirmed symptom may put the dialog off course and result in reduced automation, even though the perceived accuracy is higher. Confirmation of SSLU results is definitely an area where new research can potentially impact the performance of the whole system.

4 Experimental VUI

Voice User Interface (VUI) is typically considered an art. VUI designers acquire their experience by analyzing the effect of different prompts on the behavior of users, and can often predict whether a new prompt can help, confuse, or expedite the interaction. Unfortunately, like all technologies relying on the anecdotal experience of the designer, in VUI it is difficult to make fine adjustments to an interface and predict the effect of competing similar designs before the application is actually deployed. However, in large volume applications, and when a global measure of performance is available, one can test different non-disruptive design hypotheses on the field, while the application is running. We call this process *experimental VUI*.

There have been, in the past, several studies aimed at using machine learning for the design of dialog systems (Levin et al., 2000, Young 2002, Pietquin et al, 2006). Unfortunately, the problem of full design of a system based uniquely on machine learning is a very difficult one, and cannot be fully utilized yet for commercial systems. A simpler and less ambitious goal is that of finding the optimal dialog strategy among a small number of competing designs, where all the initial designs are working reasonably well (Walker 2000, Paek et al 2004, Lewis 2006). Comparing competing designs requires carrying on an exploration based on random selection of each design at crucial points of the dialog. Once a reward schema is defined, one can use it for changing the exploration probability so as to maximize a function of the accumulated reward using, for instance, one of the algorithms described in (Sutton 1998).

Defining many different competing designs at several points of the interaction is often impractical and costly. Moreover, in a deployed commercial application, one needs to be careful about maintaining a reasonable user experience during exploration. Thus, the competing designs have to be chosen carefully and applied to portions of the dialog where the choice of the optimal design can make a significant difference for the reward measure in use.

In the experiments described below we selected the symptom identification as a point worth exploring. in an internet technical support application We then defined three prompting schemas

- Schema A: the system plays an open prompt
- Schema B: the system plays an open prompt, and then provides some examples of requests
- Schema C: The system plays an open prompt, and then suggests a command that provides a list of choices.

The three schemas were implemented on a deployed system for limited time. There was 1/3 probability for each individual call to go through one of the above schemas. The target function chosen for optimization was the average automation rate.

Figure 2 shows the effect on the cumulated average automation rate for each one of the competing design. The exploration was carried out until the difference in the automation rate among the three designs reached statistical significance, which was after 13 days with a total number of 21,491 calls. At that point in time we established that design B had superior performance, as compared to A and C, with a difference of 0.68 percent points.
Event though the gain in total automation rate (i.e. 0.68 percent points) seems to be modest, one has to consider that this increase is simply caused only by the selection of the best wording of a single prompt in an application with thousands of prompts. One can expect to obtain more important improvements by at looking to other areas of the dialog where experimental VUI can be applied and selecting the optimal prompt can have an impact on the overall automation rate.

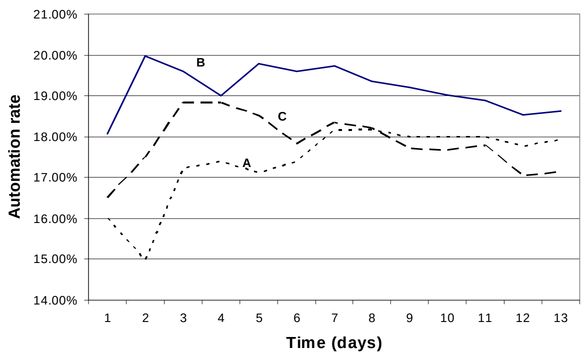


Figure 2: Daily average automation rate for competing designs.

5 Conclusions

We started this paper by describing the advances achieved in dialog system technology for commercial applications during the past decade. The industry moved from the first generation of systems able to handle very structured and simple interactions, to a current third generation where the interaction is less structured and the goal is to automate complex tasks such as problem solving and technical support.We then discussed general issues regarding the effective development of a technical support application. In particular we focused on two areas: the collection of the symptom from natural language expressions, and the experimental optimization of the VUI strategy. In both cases we described how a detailed analysis of live data can greatly help optimize the overall performance.

6 References

Barnard, E., Halberstadt, A., Kotelly, C., Phillips, M., 1999 "A Consistent Approach To Designing Spoken-dialog Systems," *Proc. of ASRU99 – IEEE Workshop*, Keystone, Colorado, Dec. 1999.

Gorin, A. L., Riccardi, G.,Wright, J. H., 1997 Speech Communication, vol. 23, pp. 113-127, 1997.

Chu-Carroll, J., Carpenter B., 1999. "Vector-based natural language call routing," *Computational Linguistics*, v.25, n.3, p.361-388, September 1999

Goel, V., Kuo, H.-K., Deligne, S., Wu S., 2005 "Language Model Estimation for Optimizing End-to-end Performance of a Natural Language Call Routing System," ICASSP 2005

Pieraccini, R., Huerta, J., Where do we go from here? Research and Commercial Spoken Dialog Systems, Proc. of 6th SIGdial Workshop on Discourse and Dialog, Lisbon, Portugal, 2-3 September, 2005. pp. 1-10

Levin, E., Pieraccini, R., Eckert, W., A Stochastic Model of Human-Machine Interaction for Learning Dialog Strategies, IEEE Trans. on Speech and Audio Processing, Vol. 8, No. 1, pp. 11-23, January 2000.

Pietquin, O., Dutoit, T., A Probabilistic Framework for Dialog Simulation and Optimal Strategy Learning, In IEEE Transactions on Audio, Speech and Language Processing, 14(2):589-599, 2006

Young, S., Talking to Machines (Statistically Speaking), Int Conf Spoken Language Processing, Denver, Colorado. (2002).

Walker, M., An Application of Reinforcement Learning to Dialogue Strategy Selection in a Spoken Dialogue System for Email . *Journal of Artificial Intelligence Research, JAIR*, Vol 12., pp. 387-416, 2000

Paek T., Horvitz E.,. Optimizing automated call routing by integrating spoken dialog models with queuing models. Proceedings of HLT-NAACL, 2004, pp. 41-48.

Lewis, C., Di Fabbrizio, G., Prompt Selection with Reinforcement Learning in an AT&T Call Routing Application, Proc. of Interspeech 2006, Pittsburgh, PA. pp. 1770-1773, (2006)

Sutton, R.S., Barto, A.G. (1998). *Reinforcement Learning: An Introduction*. MIT Press.

Olympus: an open-source framework for conversational spoken language interface research

Dan Bohus, Antoine Raux, Thomas K. Harris,
Maxine Eskenazi, Alexander I. Rudnicky
School of Computer Science
Carnegie Mellon University
{dbohus, antoine, tkharris, max, air}@cs.cmu.edu

Abstract

We introduce Olympus, a freely available framework for research in conversational interfaces. Olympus' open, transparent, flexible, modular and scalable nature facilitates the development of large-scale, real-world systems, and enables research leading to technological and scientific advances in conversational spoken language interfaces. In this paper, we describe the overall architecture, several systems spanning different domains, and a number of current research efforts supported by Olympus.

1 Introduction

Spoken language interfaces developed in industrial and academic settings differ in terms of goals, the types of tasks and research questions addressed, and the kinds of resources available.

In order to be economically viable, most industry groups need to develop real-world applications that serve large and varied customer populations. As a result, they gain insight into the research questions that are truly significant for current-generation technologies. When needed, they are able to focus large resources (typically unavailable in academia) on addressing these questions. To protect their investments, companies do not generally disseminate new technologies and results.

In contrast, academia pursues long-term scientific research goals, which are not tied to immediate economic returns or customer populations. As a result, academic groups are free to explore a larger variety of research questions, even with a high risk of failure or a lack of immediate payoff. Academic groups also engage in a more open exchange of ideas and results. However, building spoken language interfaces requires significant investments that are sometimes beyond the reach of academic researchers. As a consequence, research in academia is oftentimes conducted with toy systems and skewed user populations. In turn, this raises questions about the validity of the results and hinders the research impact.

In an effort to address this problem and facilitate research on relevant, real-world questions, we have developed Olympus, a freely available framework for building and studying conversational spoken language interfaces. The Olympus architecture, described in Section 3, has its roots in the CMU Communicator project (Rudnicky et al., 1999). Based on that experience and subsequent projects, we have engineered Olympus into an open, transparent, flexible, modular, and scalable architecture.

To date, Olympus has been used to develop and deploy a number of spoken language interfaces spanning different domains and interaction types; these systems are presented in Section 4. They are currently supporting research on diverse aspects of spoken language interaction. Section 5 discusses three such efforts: error handling, multi-participant conversation, and turn-taking.

We believe that Olympus and other similar toolkits, discussed in Section 6, are essential in order to bridge the gap between industry and academia. Such frameworks lower the cost of entry for research on practical conversational interfaces. They also promote technology transfer through the reuse of components, and support direct comparisons between systems and technologies.

2 Desired characteristics

While developing Olympus, we identified a number of characteristics that in our opinion are necessary to effectively support and foster research. The framework should be open, transparent, flexible, modular, and scalable.

Open. Complete source code should be available for all the components so that researchers and engineers can inspect and modify it towards their ends. Ideally, source code should be free for both research and commercial purposes and grow through contributions from the user community.

Transparent / Analytic. Open source code promotes transparency, but beyond that researchers must be able to analyze the system's behavior. To this end, every component should provide detailed accounts of their internal state. Furthermore, tools for data visualization and analysis should be an integral part of the framework.

Flexible. The framework should be able to accommodate a wide range of applications and research interests, and allow easy integration of new technologies.

Modular / Reusable. Specific functions (e.g. speech recognition, parsing) should be encapsulated in components with rich and well-defined interfaces, and an application-independent design. This will promote reusability, and will lessen the system development effort.

Scalable. While frameworks that rely on simple, well established approaches (e.g. finite-state dialogs in VoiceXML) allow the development of large-scale systems, this is usually not the case for frameworks that provide the flexibility and transparency needed for research. However, some research questions are not apparent until one moves from toy systems into large-scale applications. The framework should strive to not compromise scalability for the sake of flexibility or transparency.

3 Architecture

At the high level, a typical Olympus application consists of a series of components connected in a classical, pipeline architecture, as illustrated by the bold components in Figure 1. The audio signal for the user utterance is captured and passed through a speech recognition module that produces a recognition hypothesis (e.g., `two p.m.`). The recognition hypothesis is then forwarded to a language understanding component that extracts the relevant concepts (e.g., `[time=2p.m.]`), and then through a confidence annotation module that assigns a confidence score. Next, a dialog manager integrates this semantic input into the current context, and produces the next action to be taken by the system in the form of the semantic output (e.g., `{request end_time}`). A language generation module produces the corresponding surface form, which is subsequently passed to a speech synthesis module and rendered as audio.

Galaxy communication infrastructure. While the pipeline shown in bold in Figure 1 captures the logical flow of information in the system, in practice the system components communicate via a centralized message-passing infrastructure – Galaxy (Seneff et al., 1998). Each component is implemented as a separate process that connects to a traffic router – the Galaxy hub. The messages are sent through the hub, which forwards them to the appropriate destination. The routing logic is described via a configuration script.

Speech recognition. Olympus uses the Sphinx decoding engine (Huang et al., 1992). A recognition server captures the audio stream, forwards it to a set of parallel recognition engines, and collects the corresponding recognition results. The set of best hypotheses (one from each engine) is then forwarded to the language understanding component. The recognition engines can also generate n-best lists, but that process significantly slows down the systems and has not been used live. Interfaces to connect Sphinx-II and Sphinx-III engines, as well as a DTMF (touch-tone) decoder to the recognition server are currently available. The individual recognition engines can use either n-gram- or grammar-based language models. Dialog-state specific as well as class-based language models are supported, and tools for constructing language and acoustic models from data are readily available. Most of the Olympus systems described in the next section use two gender-specific Sphinx-II recognizers in parallel. Other parallel decoder configurations can also be created and used.

Language understanding is performed by Phoenix, a robust semantic parser (Ward and Issar,

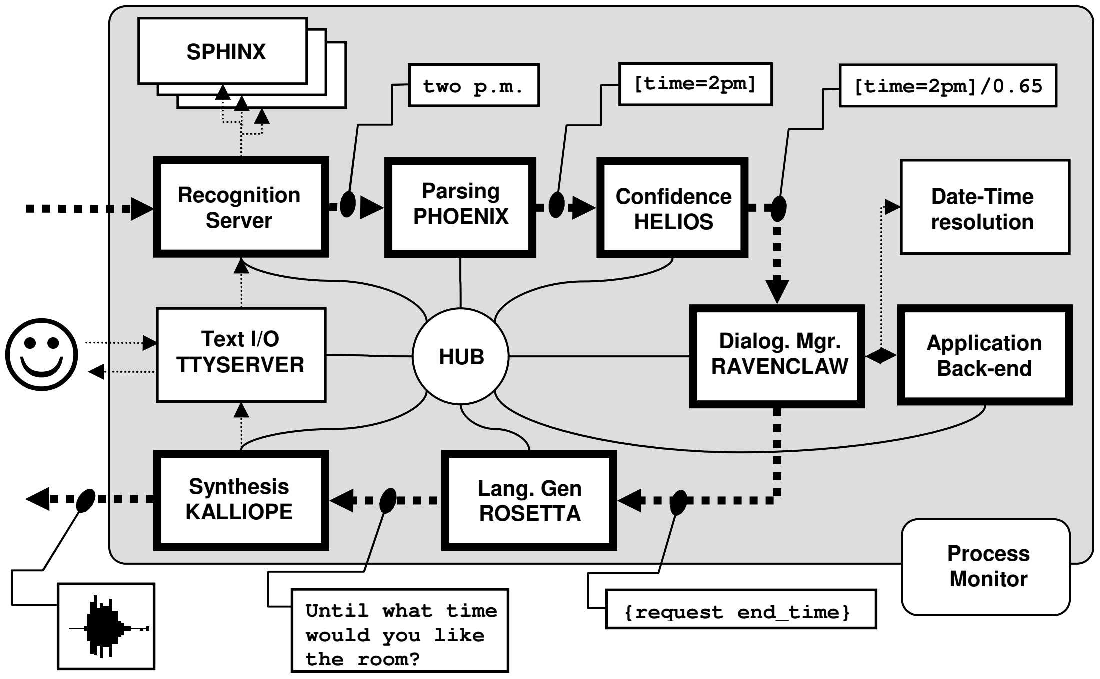


Figure 1. The Olympus dialog system reference architecture (a typical system)

1994). Phoenix uses a semantic grammar to parse the incoming set of recognition hypotheses. This grammar is assembled by concatenating a set of reusable grammar rules that capture domain-independent constructs like `[Yes]`, `[No]`, `[Help]`, `[Repeat]`, and `[Number]`, with a set of domain-specific grammar rules authored by the system developer. For each recognition hypothesis the output of the parser consists of a sequence of slots containing the concepts extracted from the utterance.

Confidence annotation. From Phoenix, the set of parsed hypotheses is passed to Helios, the confidence annotation component. Helios uses features from different knowledge sources in the system (e.g., recognition, understanding, dialog) to compute a confidence score for each hypothesis. This score reflects the probability of correct understanding, i.e. how much the system trusts that the current semantic interpretation corresponds to the user's intention. The hypothesis with the highest score is forwarded to the dialog manager.

Dialog management. Olympus uses the RavenClaw dialog management framework (Bohus and Rudnicky, 2003). In a RavenClaw-based dialog manager, the domain-specific dialog task is represented as a tree whose internal nodes capture the hierarchical structure of the dialog, and whose leaves encapsulate atomic dialog actions (e.g., asking a question, providing an answer, accessing a database). A domain-independent dialog engine executes this dialog task, interprets the input in the current dialog context and decides which action to engage next. In the process, the dialog manager may exchange information with other domain-specific agents (e.g., application back-end, database access, temporal reference resolution agent).

Language generation. The semantic output of the dialog manager is sent to the Rosetta template-based language generation component, which produces the corresponding surface form. Like the Phoenix grammar, the language generation templates are assembled by concatenating a set of predefined, domain-independent templates, with manually authored task-specific templates.

Speech synthesis. The prompts are synthesized by the Kalliope speech synthesis module. Kalliope can be currently configured to use Festival (Black and Lenzo, 2000), which is an open-source speech synthesis system, or Cepstral Swift (Cepstral 2005), a commercial engine. Finally, Kalliope also supports the SSML markup language.

Other components. The various components briefly described above form the core of the Olympus dialog system framework. Additional components have been created throughout the development of various systems, and, given the modularity of the architecture, can be easily reused. These include a telephony component, a text input-and-output interface, and a temporal reference resolution agent that translates complex date-time expressions (including relative references, holidays, etc.) into a canonical form. Recently, a Jabber interface was implemented to support interactions via the popular GoogleTalk internet messaging system. A Skype speech client component is also available.

Data Analysis. Last but not least, a variety of tools for logging, data processing and data analytics are also available as part of the framework. These tools have been used for a wide variety of tasks ranging from system monitoring, to trends analysis, to training of internal models.

A key characteristic shared by all the Olympus components is the clear separation between domain-independent programs and domain-specific resources. This decoupling promotes reuse and lessens the system development effort. To build a new system, one can focus simply on developing resources (e.g., language model, grammar, dialog task specification, generation templates) without having to do any programming. On the other hand, since all components are open-source, any part of the system can be modified, for example to test new algorithms or compare approaches.

4 Systems

To date, the Olympus framework has been used to successfully build and deploy several spoken dialog systems spanning different domains and interaction types (see Table 1). Given the limited space, we discuss only three of these systems in a bit more detail: Let's Go!, LARRI, and TeamTalk. More information about the other systems can be found in (RavenClaw-Olympus, 2007).

4.1 Let's Go!

The Let's Go! Bus Information System (Raux et al 2005; 2006) is a telephone-based spoken dialog system that provides access to bus schedules. Interaction with the system starts with an open prompt, followed by a system-directed phase where the user is asked the missing information. Each of the three or four pieces of information provided (origin, destination, time of travel, and optional bus route) is explicitly confirmed. The system knows 12 bus routes, and about 1800 place names.

Originally developed as an in-lab research system, Let's Go! has been open to the general public since March, 2005. Outside of business hours, calls to the bus company are transferred to Let's Go!, providing a constant flow of genuine dialogs (about 40 calls per weeknight and 70 per weekend night). As of March, 2007, a corpus of about 30,000 calls to the system has been collected and partially transcribed and annotated. In itself, this publicly available corpus constitutes a unique resource for the community. In addition, the system itself has been modified for research experiments (e.g., Raux et al., 2005, Bohus et al., 2006). Between-system studies have been conducted by running several versions of the system in parallel and picking one at random for every call. We have recently opened this system to researchers from other groups who wish to conduct their own experiments.

4.2 LARRI

LARRI (Bohus and Rudnicky, 2002a) is a multimodal system for support of maintenance and repair activities for F/A-18 aircraft mechanics. The system implements an Interactive Electronic Technical Manual.

LARRI integrates a graphical user interface for easy visualization of dense technical information (e.g., instructions, schematics, video-streams) with a spoken dialog system that facilitates information access and offers assistance throughout the execution of procedural tasks. The GUI is accessible via a translucent head-worn display connected to a wearable client computer. A rotary mouse (dial) provides direct access to the GUI elements.

After an initial log-in phase, LARRI guides the user through the selected task, which consists of a sequence of steps containing instructions, optionally followed by verification questions. Basic steps can include animations or short video sequences that can be accessed by the user through the GUI or through spoken commands. The user can also take the initiative and access the documentation, either via the GUI or by simple commands such as "go to step 15" or "show me the figure".

The Olympus architecture was easily adapted for this mobile and multi-modal setting. The wearable computer hosts audio input and output clients, as well as the graphical user interface. The Galaxy hub architecture allows us to easily connect these

System name	Domain / Description	Genre
RoomLine (Bohus and Rudnicky 2005)	telephone-based system that provides support for conference room reservation and scheduling within the School of Computer Science at CMU.	information access (mixed initiative)
Let's Go! Public (Raux et al 2005)	telephone-based system that provides access to bus schedule information in the greater Pittsburgh area.	information access (system initiative)
LARRI (Bohus and Rudnicky 2002)	multi-modal system that provides assistance to F/A-18 aircraft personnel during maintenance tasks.	multi-modal task guidance and procedure browsing
Intelligent Procedure Assistant (Aist et al 2002)	early prototype for a multi-modal system aimed at providing guidance and support to the astronauts on the International Space Station during the execution of procedural tasks and checklists.	multi-modal task guidance and procedure browsing
TeamTalk (Harris et al 2005)	multi-participant spoken language command-and-control interface for a team of robots in the treasure-hunt domain.	multi-participant command-and-control
VERA	telephone-based taskable agent that can be instructed to deliver messages to a third party and make wake-up calls.	voice mail / message delivery
Madeleine	text-based dialog system for medical diagnosis.	diagnosis
ConQuest (Bohus et al 2007)	telephone-based spoken dialog system that provides conference schedule information.	information access (mixed-initiative)
RavenCalendar (Stenchikova et al 2007).	multimodal dialog system for managing personal calendar information, such as meetings, classes, appointments and reminders (uses Google Calendar as a back-end)	information access and scheduling

Table 1. Olympus-based spoken dialog systems (shaded cells indicate deployed systems)

components to the rest of the system, which runs on a separate server computer. The rotary-mouse events from the GUI are rendered as semantic inputs and are sent to Helios which in turn forwards the corresponding messages to the dialog manager.

4.3 TeamTalk

TeamTalk (Harris et al., 2005) is a multi-modal interface that facilitates communication between a human operator and a team of heterogeneous robots, and is designed for a multi-robot-assisted treasure-hunt domain. The human operator uses spoken language in concert with pen-gestures on a shared live map to elicit support from teams of robots. This support comes in the forms of mapping unexplored areas, searching explored areas for objects of interest, and leading the human to said objects. TeamTalk has been built as a fully functional interface to real robots, including the Pioneer P2DX and the Segway RMP. In addition, it can interface with virtual robots within the high-fidelity USARSim (Balakirsky et al., 2006) simulation environment. TeamTalk constitutes an excellent platform for multi-agent dialog research.

To build TeamTalk, we had to address two challenges to current architecture. The multi-participant nature of the interaction required multiple dialog managers; the live map with pen-gestured references required a multi-modal integration. Again, the flexibility and transparency of the Olympus framework allowed for relatively simple solutions to both of these challenges. To accommodate multi-participant dialog, each robot in the domain is associated with its own RavenClaw-based dialog manager, but all robots share the other Olympus components: speech recognition, language understanding, language generation and speech synthesis. To accommodate the live map GUI, a Galaxy server was built in Java that could send the user's inputs to Helios and receive outputs from RavenClaw.

5 Research

The Olympus framework, along with the systems developed using it, provides a robust basis for research in spoken language interfaces. In this section, we briefly outline three current research efforts supported by this architecture. Information about other supported research can be found in (RavenClaw-Olympus, 2007).

5.1 Error handling

A persistent and important problem in today's spoken language interfaces is their lack of robustness when faced with understanding errors. This problem stems from current limitations in speech recognition, and appears across most domains and interaction types. In the last three years, we conducted research aimed at improving robustness in spoken language interfaces by: (1) endowing them with the ability to accurately detect errors, (2) developing a rich repertoire of error recovery strategies and (3) developing scalable, data-driven approaches for building error recovery policies[1]. Two of the dialog systems from Table 1 (Let's Go! and RoomLine) have provided a realistic experimental platform for investigating these issues and evaluating the proposed solutions.

With respect to **error detection**, we have developed tools for learning confidence annotation models by integrating information from multiple knowledge sources in the system (Bohus and Rudnicky, 2002b). Additionally, Bohus and Rudnicky (2006) proposed a data-driven approach for constructing more accurate beliefs in spoken language interfaces by integrating information across multiple turns in the conversation. Experiments with the RoomLine system showed that the proposed belief updating models led to significant improvements (equivalent with a 13.5% absolute reduction in WER) in both the effectiveness and the efficiency of the interaction.

With respect to **error recovery strategies**, we have developed and evaluated a large set of strategies for handling misunderstandings and non-understandings (Bohus and Rudnicky, 2005). The strategies are implemented in a task-decoupled manner in the RavenClaw dialog management framework.

Finally, in (Bohus et al., 2006) we have proposed a novel online-learning based approach for building **error recovery policies** over a large set of non-understanding recovery strategies. An empirical evaluation conducted in the context of the Let's Go! system showed that the proposed approach led to a 12.5% increase in the non-understanding recovery rate; this improvement was attained in a relatively short (10-day) time period.

The models, tools and strategies developed throughout this research can and have been easily reused in other Olympus-based systems.

5.2 Multi-participant conversation

Conversational interfaces are generally built for one-on-one conversation. This has been a workable assumption for telephone-based systems, and a useful one for many single-purpose applications. However this assumption will soon become strained as a growing collection of always-available agents (e.g., personal trainers, pedestrian guides, or calendar systems) and embodied agents (e.g., appliances and robots) feature spoken language interfaces. When there are multiple active agents that wish to engage in spoken dialog, new issues arise. On the input side, the agents need to be able to identify the addressee of any given user utterance. On the output side, the agents need to address the problem of channel contention, i.e., multiple participants speaking over each other.

Two architectural solutions can be envisioned: (1) the agents share a single interface that understands multi-agent requirements, or (2) each agent uses its own interface and handles multi-participant behavior. Agents that provide different services have different dialog requirements, and we believe this makes a centralized interface problematic. Furthermore, the second solution better fits human communication behavior and therefore is likely to be more natural and habitable.

TeamTalk is a conversational system that follows the second approach: each robot has its own dialog manager. Processed user inputs are sent to all dialog managers in the system; each dialog manager decides based on a simple algorithm (Harris et al., 2004) whether or not the current input is addressed to it. If so, an action is taken. Otherwise the input is ignored; it will be processed and responded to by another robot. On the output side, to address the channel contention problem, each RavenClaw output message is augmented with information about the identity of the robot that generated it. The shared synthesis component queues the messages and plays them back sequentially with the corresponding voice.

We are currently looking into two additional challenges related to multi-participant dialog. We are interested in how to address groups and subgroups in addition to individuals of a group, and we are also interested in how to cope with multiple humans in addition to multiple agents. Some experiments investigating solutions to both of these issues have been conducted. Analysis of the results and refinements of these methods are ongoing.

5.3 Timing and turn-taking

While a lot of research has focused on higher levels of conversation such as natural language understanding and dialog planning, low-level interactional phenomena such as turn-taking have not

[1] A policy specifies how the system should choose between different recovery strategies at runtime.

received as much attention. As a result, current systems either constrain the interaction to a rigid one-speaker-at-a-time style or expose themselves to interactional problems such as inappropriate delays, spurious interruptions, or turn over-taking (Raux et al., 2006). To a large extent, these issues stem from the fact that in common dialog architectures, including Olympus, the dialog manager works asynchronously from the real world (i.e., utterances and actions that are planned are assumed to be executed instantaneously). This means that user barge-ins and backchannels are often interpreted in an incorrect context, which leads to confusion, unexpected user behavior and potential dialog breakdowns. Additionally, dialog systems' low-level interactional behavior is generally the result of ad-hoc rules encoded in different components that are not precisely coordinated.

In order to investigate and resolve these issues, we are currently developing version 2 of the Olympus framework. In addition to all the components described in this paper, Olympus 2 features an Interaction Manager which handles the precise timing of events perceived from the real world (e.g., user utterances) and of system actions (e.g., prompts). By providing an interface between the actual conversation and the asynchronous dialog manager, Olympus 2 allows a more reactive behavior without sacrificing the powerful dialog management features offered by RavenClaw. Olympus 2 is designed so that current Olympus-based systems can be upgraded with minimal effort.

This novel architecture, initially deployed in the Let's Go system, will enable research on turn-taking and other low-level conversational phenomena. Investigations within the context of other existing systems, such as LARRI and TeamTalk, will uncover novel challenges and research directions.

6 Discussion and conclusion

The primary goal of the Olympus framework is to enable research that leads to technological and scientific advances in spoken language interfaces.

Olympus is however by no means a singular effort. Several other toolkits for research and development are available to the community. They differ on a number of dimensions, such as objectives, scientific underpinnings, as well as technological and implementation aspects. Several toolkits, both commercial, e.g., TellMe, BeVocal, and academic, e.g., Ariadne (2007), SpeechBuilder (Glass et al., 2004), and the CSLU toolkit (Cole, 1999), are used for rapid development. Some, e.g., CSLU and SpeechBuilder, have also been used for educational purposes. And yet others, such as Olympus, GALATEEA (Kawamoto et al., 2002) and DIPPER (Bos et al., 2003) are primarily used for research. Different toolkits rely on different theories and dialog representations: finite-state, slot-filling, plan-based, information state-update. Each toolkit balances tradeoffs between complexity, ease-of-use, control, robustness, flexibility, etc.

We believe the strengths of the Olympus framework lie not only in its current components, but also in its open, transparent, and flexible nature. As we have seen in the previous sections, these characteristics have allowed us to develop and deploy practical, real-world systems operating in a broad spectrum of domains. Through these systems, Olympus provides an excellent basis for research on a wide variety of spoken dialog issues. The modular construction promotes the transfer and reuse of research contributions across systems.

While desirable, an in-depth understanding of the differences between all these toolkits remains an open question. We believe that an open exchange of experiences and resources across toolkits will create a better understanding of the current state-of-the-art, generate new ideas, and lead to better systems for everyone. Towards this end, we are making the Olympus framework, as well as a number of systems and dialog corpora, freely available to the community.

Acknowledgements

We would like to thank all those who have brought contributions to the components underlying the Olympus dialog system framework. Neither Olympus nor the dialog systems discussed in this paper would have been possible without their help. We particularly wish to thank Alan W Black for his continued support and advice. Work on Olympus components and systems was supported in part by DARPA, under contract NBCH-D-03-0010, Boeing, under contract CMU-BA-GTA-1, and the US National Science Foundation under grant number 0208835. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.

References

Aist, G., Dowding, J., Hockey, B.A., Rayner, M., Hieronymus, J., Bohus, D., Boven, B., Blaylock, N., Campana, E., Early, S., Gorrell, G., and Phan, S., 2003. *Talking through procedures: An intelligent Space Station procedure assistant*, in Proc. of EACL-2003, Budapest, Hungary

Ariadne, 2007, *The Ariadne web-site,* as of January 2007, http://www.opendialog.org/.

Balakirsky, S., Scrapper, C., Carpin, S., and Lewis, M. 2006. *UsarSim: providing a framework for multi-robot performance evaluation,* in Proc. of PerMIS.

Black, A. and Lenzo, K., 2000. *Building Voices in the Festival Speech System*, http://festvox.org/bsv/, 2000.

Bohus, D., Grau Puerto, S., Huggins-Daines, D., Keri, V., Krishna, G., Kumar, K., Raux, A., Tomko, S., 2007. *Conquest – an Open-Source Dialog System for Conferences*, in Proc. of HLT 2007, Rochester, USA.

Bohus, D., Langner, B., Raux, A., Black, A., Eskenazi, M., Rudnicky, A. 2006. *Online Supervised Learning of Non-understanding Recovery Policies*, in Proc. of SLT-2006, Aruba.

Bohus, D., and Rudnicky, A. 2006. *A K-hypotheses + Other Belief Updating Model*, in Proc. of the AAAI Workshop on Statistical and Empirical Methods in Spoken Dialogue Systems, 2006.

Bohus, D., and Rudnicky, A., 2005. *Sorry I didn't Catch That: An Investigation of Non-understanding Errors and Recovery Strategies*, in Proc. of SIGdial-2005, Lisbon, Portugal.

Bohus, D., and Rudnicky, A., 2003. *RavenClaw: Dialog Management Using Hierarchical Task Decomposition and an Expectation Agenda,* in Proc. of Eurospeech 2003, Geneva, Switzerland.

Bohus, D., and Rudnicky, A., 2002a. *LARRI: A Language-based Maintenance and Repair Assistant,* in Proc. of IDS-2002, Kloster Irsee, Germany.

Bohus, D., and Rudnicky, A., 2002b. *Integrating Multiple Knowledge Sources in the CMU Communicator Dialog System,* Technical Report CMU-CS-02-190.

Bos, J., Klein, E., Lemon, O., and Oka, T., 2003. *DIPPER: Description and Formalisation of an Information-State Update Dialogue System Architecture*, in Proc. of SIGdial-2003, Sapporo, Japan

Cepstral, LLC, 2005. Swift^{TM}: Small Footprint Text-to-Speech Synthesizer, http://www.cepstral.com.

Cole, R., 1999. *Tools for Research and Education in Speech Science*, in Proc. of the International Conference of Phonetic Sciences, San Francisco, USA.

Glass, J., Weinstein, E., Cyphers, S., Polifroni, J., 2004. *A Framework for Developing Conversational Interfaces*, in Proc. of CADUI, Funchal, Portugal.

Harris, T. K., Banerjee, S., Rudnicky, A., Sison, J. Bodine, K., and Black, A. 2004. *A Research Platform for Multi-Agent Dialogue Dynamics*, in Proc. of The IEEE International Workshop on Robotics and Human Interactive Communications, Kurashiki, Japan.

Harris, T. K., Banerjee, S., Rudnicky, A. 2005. *Heterogenous Multi-Robot Dialogues for Search Tasks,* in AAAI Spring Symposium: Dialogical Robots, Palo Alto, California.

Huang, X., Alleva, F., Hon, H.-W., Hwang, M.-Y., Lee, K.-F. and Rosenfeld, R., 1992. *The SPHINX-II Speech Recognition System: an overview*, in Computer Speech and Language, 7(2), pp 137-148, 1992.

Kawamoto, S., Shimodaira, H., Nitta, T., Nishimoto, T., Nakamura, S., Itou, K., Morishima, S., Yotsukura, T., Kai, A., Lee, A., Yamashita, Y., Kobayashi, T., Tokuda, K., Hirose, K., Minematsu, N., Yamada, A., Den, Y., Utsuro, T., and Sagayama, S., 2002. *Open-source software for developing anthropomorphic spoken dialog agent*, in Proc. of PRICAI-02, International Workshop on Lifelike Animated Agents.

Raux, A., Langner, B., Bohus, D., Black, A., and Eskenazi, M. 2005, *Let's Go Public! Taking a Spoken Dialog System to the Real World,* in Proc. of Interspeech 2005, Lisbon, Portugal.

Raux, A., Bohus, D., Langner, B., Black, A., and Eskenazi, M. 2006 *Doing Research on a Deployed Spoken Dialogue System: One Year of Let's Go! Experience,* in Proc. of Interspeech 2006, Pittsburgh, USA.

RavenClaw-Olympus web page, as of January 2007: http://www.ravenclaw-olympus.org/.

Rudnicky, A., Thayer, E., Constantinides, P., Tchou, C., Shern, R., Lenzo, K., Xu W., and Oh, A., 1999. *Creating natural dialogs in the Carnegie Mellon Communicator system,* in Proc. of Eurospeech 1999.

Seneff, S., Hurley, E., Lau, R., Pao, C., Schmid, P., and Zue V. 1998 *Galaxy-II: A reference architecture for conversational system development,* in Proc. of ICSLP98, Sydney, Australia.

Stenchikova, S., Mucha, B., Hoffman, S., Stent, A., 2007. *RavenCalendar: A Multimodal Dialog System for Managing A Personal Calendar*, in Proc. of HLT 2007, Rochester, USA.

Ward, W., and Issar, S., 1994. *Recent improvements in the CMU spoken language understanding system,* in Proc. of the ARPA Human Language Technology Workshop, pages 213–216, Plainsboro, NJ.

Toward Evaluation that Leads to Best Practices: Reconciling Dialog Evaluation in Research and Industry

Tim Paek

Microsoft Research
One Microsoft Way
Redmond, WA 98052
timpaek@microsoft.com

Abstract

Dialog evaluation is approached in different ways by research and industry. While researchers have sought commensurable evaluation metrics that allow for comparison of disparate systems with varying tasks and domains, industry engineers have focused mostly on best practices and delivering a return-on-investment to customers. In this paper, we contend that the problem of finding commensurable metrics also applies to commercial evaluation, and critically survey four candidate metrics for commensurability. Finally, in light of the problems faced by the candidate metrics, we advocate a collaborative agenda for dialog evaluation based on using statistical meta-analysis for empirically establishing best practices from any evaluation metric.

1 Introduction

Since the beginning of speech recognition research, which started more than 50 years ago, people have dreamed of being able to talk to machines and appliances as if they were human. What began in academic institutions and industry laboratories gradually made its way into the marketplace around the mid 1990s with the commercial introduction of voice user interfaces (VUI) (Pieraccini & Lubensky, 2005). Most VUI applications were spoken dialogue systems for automating customer service tasks. Nowadays, hundreds of commercial systems are being deployed by companies each year, adhering to industry-wide standards and protocols established to ensure the interoperability of components and vendors, such as VoiceXML, CCXML, MRCP, etc. Unfortunately, as some researchers have pointed out (Pieraccini & Huerta, 2005), dialog systems in industry have been evolving on a parallel path with those in academic research, where usability and cost have been the primary goal of industry, and naturalness of interaction and freedom of expression have been the goal of research. Given that commercial systems are now beginning to embrace less-constrained interaction models, a call has been made for a "synergistic convergence" of architectures, abstractions and methods from both communities, lest research results become irrelevant to industry practice (Pieraccini & Huerta, 2005).

It is under this motivation that we critically survey the field of dialog evaluation in research and industry[1]. Dialog evaluation is a task common to both communities, but it has been approached in distinct ways. Research has exerted considerably effort and attention to devising evaluation metrics that allow for comparison of disparate systems with varying tasks and domains. In industry, system engineers generally do not gruel over what metric is best-suited for comparison of disparate systems. Because companies live and die by practical evaluation, what matters most is that they improve their customers' business; hence, beyond measures that evince return-on-investment (ROI), there is little focus on dialog evaluation metrics.

This paper endeavors to bridge the gap in understanding of dialog evaluation between academic research and industry. We explore what research and industry has to learn from each other, and how working together can advance the goals of both communities. We do this in three sections. In the first section, we describe differences in the way

[1] Though lines are often blurred, by "research" we mean academic institutions and industry laboratories whose focus is not immediate commercial gain.

evaluation is pursued by both communities. In particular, we discuss the academic search for commensurable metrics that allow for comparison of disparate systems. In considering evaluation in industry, we expound on the drive for VUI best practices. In the second section, we survey four candidate metrics for commensurability. Irrespective of whether they achieve that purpose, they are likely to be of practical interest to system engineers in industry. Finally, in the last section, we propose a collaborative agenda for dialog evaluation to advance the goals of both communities. In particular, we advocate statistical meta-analysis for empirically establishing best practices from any dialog evaluation metric.

2 Differences in Approach

In research, dialog evaluation is considered a hard problem. Several workshops and special issues of journals have already been devoted to this topic. On the other hand, in industry, dialog evaluation is considered relatively straightforward. Any dialog system worth the effort of deployment must ultimately deliver ROI. That ROI may be in terms of the cost savings accrued from automating what is typically handled by human operators, such as in call centers, or in terms of expanding the breadth and depth of customer service that has typically been privy to large enterprises. The focus in industry has not been so much on how best to evaluate a dialog system, but rather on how best to design them. In short, industry tends to focus more on best practices than on evaluation.

The difference in thinking cannot be fully attributed to the difference in goals propounded earlier (Pieraccini & Huerta, 2005); namely, that industry generally pursues usability and cost-effectiveness, whereas research pursues unconstrained spoken interaction under the assumption that usability would naturally follow. There is also a lack of understanding about what dialog evaluation in research has to offer industry, and in particular, for the kind of directed dialogs (which restrict what users can say but generally improve usability) that are common in commercial systems. In this section, we explore why dialog evaluation is considered so challenging in the research community, and demonstrate how many of the same issues that researchers face are also applicable to VUI engineers as well.

2.1 Commensurability Problem

Observing the success that both the speech recognition and spoken language understanding communities have enjoyed in advancing their technologies by establishing a controlled, objective and common evaluation framework (Pieraccini & Lubensky, 2005), dialog researchers have sought a similar evaluation framework for their work. This framework could not only be used to gauge technological progress in the field but also allow for the assessment of diverse systems of varying tasks and domains. Unfortunately, the research community has yet to agree upon such a framework. To date, researchers operate under on a variety of different frameworks, and new evaluation metrics are proposed all the time.

Part of the reason for this has to do with the complexity of the evaluation task. On the one hand, dialog systems are ultimately created for users, so usability factors such as satisfaction or likelihood of future use should be primary. On the other hand, because usability factors are subjective, they can be erratic and highly dependent on the complex interplay of user interface attributes (Kamm et al., 1999). So, designers have turned to objective metrics such as task completion time or dialog success rate (e.g., see Gibbon et al., 1998 for review). Due to the interactive nature of conversation however, these metrics do not always correspond to the most effective user experience (Hartikainen et al., 2004; Lamel et al., 2000). Objective evaluation of user experience can itself be highly uncertain or non-existent (Dybkjaer & Bernsen, 2001). Furthermore, in many cases, it is just not clear how to apply an objective metric. Even an ostensibly straightforward metric, such as task success, can be difficult to ascertain. For example, defining the "success" of a session with an intelligent tutoring system is no easy task, and may or may not have anything to do with student learning, depending on what constitutes the basis for comparison (either a human or a keyboard system).

The choice of evaluation metric depends on the purpose of the evaluation (Dybkjaer & Bernsen, 2001; Paek, 2001). Some researchers are more interested in achieving human-human conversation-like qualities in their systems than others. Because researchers have different purposes, they have developed a wide assortment of dialog evaluation metrics. As mentioned earlier, metrics can be subjective or objective, deriving from questionnaires or log files. They can vary in scale from the utterance level to the overall dialog (Glass et al., 2000). They can treat the system as a "black box" and describe only its external behavior (Eckert at al., 1998), or as a "glass box" and detail its internal processing. If one metric fails to suffice, several metrics can be combined (Walker et al., 1997). Finally, if all else fails to suffice, then new metrics can be developed.

Despite the diversity of metrics and purposes, researchers have wanted to compare their dialog systems against others. They have sought an evaluation metric or framework that could facilitate comparative judgments. In philosophical terms, they are seeking a measure of commensurability; two quantities are commensurable if both can be measured by the same units. But what units can allow one dialog system to be compared against another when they vary along so many different dimensions, such as components, interface attributes, domains and tasks? These different aspects can also interact with each other in highly complex ways.

Commensurability is not only a problem for research, but also for industry as well. Suppose that a commercially deployed system adhering to VUI best practices allows a customer to achieve a 90% task completion rate and a savings of $500 million dollars. Because the system consists of many architectural and interface attributes that may interact with one another, from the exact wording of the prompts to the dialog management strategies employed, how can system engineers really know if they found the optimal configuration? Perhaps given a new set of dialog strategies, or slightly different prompt wording, task completion could be significantly improved. The issue of commensurability still applies because ideally engineers would like to be able to say that the system they built, with the configuration that they arrived at, is somehow better than other systems that they, or even their competitors, could have designed. Of course, free market economics could be the judge, and engineers who provide higher ROI might be able to stake their claim on superiority. However, the factors that play a role in making a dialog system usable and efficacious can be multifaceted, and may even reach beyond the choices of the designer to the characteristics of the user population, or user profile, and usage patterns (Frostad, 2003).

2.2 Best Practices

System engineers in industry have implicitly dealt with the issue of commensurability by relying on VUI best practices. Best practices often emerge through trial-and-error and the test of time, and essentially serve as de facto industry standards (Balentine & Morgan, 2001). The need for best practices evolved from classical software engineering principles. Because system engineers were responsible for the complete specification of system behavior, they began applying software engineering principles such as requirements gathering, specification, design and coding, usability testing, and post-deployment tuning to make sure that their systems could scale into high-quality, commercial grade solutions (Pieraccini & Huerta, 2005). Along the way, engineers encountered problems, and as they began to notice the same problems appearing over again, they began to devise best practices for the design and deployment of VUI systems. As the industry has matured over the years, best practices have been collected into books (e.g., Balentine & Morgan, 2001; Cohen et al., 2005), and many platform providers offer seminars, training, and online resources for learning best practices (e.g., Frostad, 2003).

It is important to note that best practices are not the sole propriety of industry alone. Academic researchers, who have been building a multitude of systems under various government sponsored projects, such as DISC and DISC-2, have also developed their own best practices (e.g., Lamel et al., 2000; Dybkjaer & Bernsen, 2001).

Best practices often come in the form of practical dos and don'ts. For example, for telephony-based systems, almost all published literature in industry and research recommends that prompts be kept short and simple. Sometimes these practices, such as this one for prompts, are validated either directly or indirectly by experimental design. Various academic institutions have also pursued VUI design experiments, and published their findings, which often get cited in industry. For example, both the Dialogue Engineering Project[2] at CCRI, University of Edinburgh and the Stanford CHIMe Lab[3] are well-known to industry (e.g., Nass & Brave, 2005).

[2] http://www.ccir.ed.ac.uk/doc/ccir_dialogues.htm
[3] http://chime.stanford.edu/

The problem with best practices is that they are often not substantiated by rigorous experimental design, which is why the previously mentioned academic institutions have sought to conduct their research. In the worst case, best practices are based on the accumulated experience and intuition of system engineers and consultants, which unfortunately is prone to error and cannot be generalized beyond their limited experience. This can result in ostensibly contradictory recommendations. For example, whereas one best practice may advocate personifying the dialog system using the first person singular, another may advocate adhering as close as possible to a non-personified, touch-tone model. In this particular case, the contradiction stems from limited knowledge of the technologies available at the time; the first was made with HMIHY technology (reference) for mixed-initiative interaction in mind, whereas the other was not.

Even when best practices are based on experimental studies, they cannot be automatically generalized beyond the conditions and assumptions of the experimental design. Controlled experimental design dictates that in order to find a significant effect of a treatment, such as prompt wording or gender of voice, other factors should be held constant, such as the dialog flow of the system. When those other factors change, the effect of the treatment may change as well. Hence, results cannot be automatically generalized as best practices beyond their experimental settings. Furthermore, as many of the studies themselves point out, they are limited to the characteristics of their subject population. In fact, a common industry best practice is to conduct pilot usability studies on the domain task to better understand the needs and usage patterns of the expected user population (Balentine & Morgan, 2001).

The point here is not to discourage the use of best practices, but to highlight the need for rigorous validation of them. Incommensurability poses a problem for best practices because when disparate systems cannot be evaluated according to a common framework, it is hard to generalize system features or attributes into best practices. Ideally, dialog evaluation should foster the development of best practices.

3 Survey of Commensurable Metrics

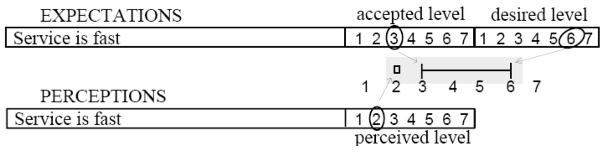


Figure 1. A graphical display of the gap between user expectations and perceptions for the SERVQUAL method.

In the research literature, several dialog evaluation metrics have been proposed which in some fashion or another could allow for the comparison of disparate systems. In this section, we critically survey four such metrics. Although other metrics might qualify as well, these metrics were chosen because they are likely to be of practical interest to system engineers in industry, regardless of whether they facilitate commensurability.

3.1 SERVQUAL

SERVQUAL is a SERVice QUALity evaluation method developed by marketing academics and applied to spoken dialogue systems by Hartikainen et al. (2004). SERVQUAL consists of a questionnaire and methods of data analysis. The questionnaire[4] provides a subjective measure of the gap between expectations and perceptions in five service quality dimensions: tangibles, reliability, responsiveness, assurance and empathy. Once questionnaire data is collected, two measures, a Measure of Service Superiority (MSS = Perceived level – Desired Level) and a Measure of Service Adequacy (MSA = Perceived Level – Acceptable Level), can be easily computed. Figure 1 shows how these two measures can then be used to display a "zone of tolerance" for users (Hartikainen et al., 2004). Graphical plots showing the relationship between performance and importance are also commonly used.

The SERQUAL method could be considered a commensurable metric because it evaluates the usability of dialog systems with respect to a common unit of measurement; namely, the gap between user expectations and perceptions. Even if disparate systems engender different expectations in users, perhaps because of dissimilar tasks and domains, they can still be compared against each with respect to how far off the reality of their perceived performance is from user expectation.

[4] Accessible online at http://www.cs.uta.fi/hci/spi/SERQUAL/.

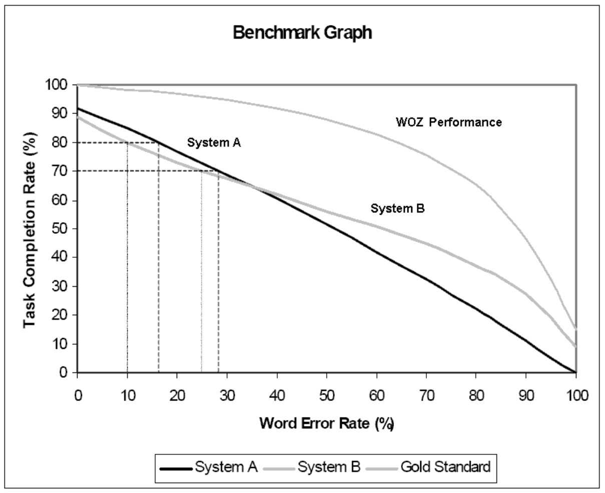


Figure 2. Comparison of task completion rate between two systems and a WOZ gold standard.

The primary problem with SERQUAL is that user perceptions and expectations can be unstable and easily susceptible to manipulation. For example, in conducting numerous experiments on user perception of speech interfaces, Nass and Brave (2005) note: “Reminding people that they depend on the interface for their success automatically makes the computer seem more intelligent… labeling a part of an interface as a specialist, conforming to gender stereotypes, flattering the user, or matching the user's personality also increases perceived competence. Indeed, people are so susceptible to manipulation that perceived intelligence is a very weak predictor of actual intelligence” (p.152). Without fully understanding the subtle factors that can easily influence user perception, it is possible to attribute service quality superiority to the wrong factor. A final concern is that focusing on user perceptions in evaluation may detract some from working out more serious technical flaws.

Despite the problems with SERQUAL, it has a strong tradition in marketing and may appeal to those in industry who need to pitch the value of their systems from a customer service standpoint. Researchers and engineers alike can also benefit from using SERVQUAL because it draws attention to how user perceptions can thwart even well-designed systems.

3.2 WOZ Gold Standard

Whereas SERQUAL measures the gap between user expectations and perceptions, Paek (2001) advocates using human performance in Wizard-of-Oz (WOZ) experiments as a gold standard for benchmarking systems. The idea is that once an evaluation metric (e.g., task completion) is selected, a WOZ experiment can be conducted that compares different treatments of interest (e.g., dialog repair strategies) along varying levels of word-error rate. The human wizards in these experiments never hear users directly, but instead receive output mediated by the recognizer and/or spoken language understanding unit. Although other researchers have recommended similar WOZ setups (Stuttle et al., 2004), the focus here is on quantifying the difference in performance between the system and a human gold standard, and using that as a commensurable metric. For example, Figure 2 displays the task completion rates of two dialog systems as well as a human wizard. The performance of each system is measured as the difference in density between that system and the WOZ. Notice that depending on the interval of interest in word-error rate, system A can be better than B and vice versa. The difference in density between the WOZ performance and the absolute upper-bound represents the difficulty of the domain task for human wizards, or the “benchmark complexity” (Paek, 2001).

Using human wizards as a gold standard allows for the comparison of disparate systems. If the difference between the performance of any system and a human wizard is small, then that might suggest that the system is performing very well for that domain task, regardless of what that domain task is itself. However, if the benchmark complexity is also small, that would suggest that the system may be performing well because the task is easy.

The major problem with using human wizards as a gold standard is the effort required to conduct WOZ experiments. It is not only time-consuming and costly, but technically challenging to insert a wizard into the right place in the processing of utterances and to make sure that they can effectively do their job. Once a wizard is in place, it can also be difficult to obtain data points along a wide range of word-error rates.

Despite these problems, having a human gold standard naturally lends itself to optimization, which is always of interest to industry. System engineers can identify which components are contributing the most to a performance metric by examining the density differences with and without particular components. Furthermore, if customers are willing to identify how much they might be willing to pay to achieve various levels of performance – i.e., if their utility functions are elicited, then it is possible to calculate average marginal costs by weighting density differences by their corresponding utilities (Paek, 2001).

3.3 SASSI

Noting the lack of psychometric validation of subjective usability measures often used in evaluations of spoken dialog systems, Hone & Graham (2000) propose a questionnaire measure for the Subjective Assessment of Speech System Interfaces (SASSI). They identify four weaknesses of subjective evaluation measures (e.g., questionnaire items), which are worth repeating here. First, the content and structure of these measures are for the most part arbitrary and based on intuition. Second, measures are not validated against other subjective or objective measures. This renders their construct validity suspect; that is, it difficult to tell if they really measure what they are intended to measure. Third, these measures do not report their reliability, both in terms of their test-retest stability across time, as well as the internal consistency of a group of measures for a particular construct. Finally, measures are commonly summed or averaged to obtain an overall score when such an approach can only be justified on the basis of evidence that all of the measures really do assess the same construct (Hone & Graham, 2000).

In order to build a psychometrically valid, reliable and sensitive questionnaire, SASSI was developed by first taking questionnaire items from established measures in the research literature, and then having users respond to those items with respect to four different speech applications. After collecting the data, exploratory factor analysis was conducted to find a set of theoretical constructs, or "factors". These factors are represented by a set of questionnaire items which tend to be highly correlated with each other. Six main factors in users' perceptions of speech applications were identified:

- System Response Accuracy: User's perceptions of the system as accurate and doing what they expect.
- Likeability: User's rating of the system as useful, pleasant and friendly.
- Cognitive Demand: The perceived amount of effort needed to interact with the system and feelings arising from this effort.
- Annoyance: User's rating of the system as repetitive, boring, irritating and frustrating.
- Habitability: The extent to which users knew what to do and what the system was doing.
- Speed: How quickly the system responded to user inputs.

It is important to note that this factor analysis was preliminary and did not benefit from multiple iterations. Hence, only the first three factors had internal consistency reliabilities, as measured by Cronbach's alpha, of $\alpha \geq 0.80$, which is typically required for widespread adoption.

Although the development of SASSI is definitely a promising start for creating valid and reliable subjective measures, it does not say much about how system features, such as prompt wording, influence the six factors identified. In other words, SASSI only tells system engineers what to measure (which is an important contribution), not how to design their systems. For that, statistical analyses relating system features to SASSI measures are needed. Nevertheless, the SASSI methodology represents a valuable, principled way of determining common units of measurement for comparing disparate systems. We return to this issue again in Section 4.

3.4 PARADISE

Perhaps the best-known general framework in research for dialog evaluation is PARADISE (PARAdigm for Dialogue System Evaluation) (Walker et al., 1997). PARADISE addresses three goals: 1) to support the comparison of multiple systems on the same domain task, 2) to provide a method for developing predictive models of user satisfaction as a function of system features, and 3) to provide a technique for making generalizations across systems about which features impact usability (Walker et al., 2000). Treating user satisfaction as the primary objective function, PARADISE derives a combined performance metric as a weighted linear combination of task-success measures and dialog costs, the latter consisting of two types: dialog efficiency metrics (e.g., elapsed time), and dialog quality metrics (e.g., mean recognition score). Deriving the metric simply involves model-building using multivariate linear regression.

Although PARADISE is geared towards comparing systems that perform the same domain task, it does provide a general framework for at least comparing those systems. The problem is that while PARADISE is a useful descriptive tool, its power to generalize has been somewhat limited. In pursuing the third goal of generalization – to figure out what system features really matters to users, PARADISE was applied to experimental data from three different dialog systems (Walker et al, 2000). Models trained on one system were then tested on the other two systems. Results showed that the models do indeed generalize well across the three systems. However, the three features that consistently appeared among the top predictive factors were mean recognition score, whether users reported that they had completed the task, and the percentage of recognition rejections. Unfortunately, this is not the kind of insight that leads to best practices, and most system engineers probably already knew that improving speech recognition and task completion (either in absolute terms or by user perception) would improve user satisfaction.

What is likely to be of practical interest to system engineers in industry about PARADISE is the usefulness of performing multivariate linear regression to predict measures of interest based on not only task success but measures of dialog efficiency and dialog quality. Because most of the PARADISE features can be automatically generated from data, apart from having users fill out a satisfaction survey, it is of almost no cost to perform a PARADISE analysis.

4 Evaluation That Leads to Best Practices

In the previous section, we critically surveyed four dialog evaluation metrics that could be considered candidates for commensurability. In light of the problems faced by these metrics, in this Section, we propose a collaborative agenda for dialog evaluation that fulfills the need in industry for best practices and the research pursuit of generalizations. Before considering the proposal, however, we reassess the value of commensurability.

4.1 Reassessing Commensurability

Although commensurability seems to be worthwhile, in looking closely at the desire to compare disparate systems of varying domains and tasks, it is important to separate the question of, "How is the dialog system doing relative to other systems?" from "How can the dialog system do better?" Answering the former question is truly a challenging task, and metrics like SERVQUAL and the WOZ gold standard offer interesting solutions. However, there is little to gain from answering this question other than bragging rights. The latter question of how to improve a dialog system is ultimately more beneficial to research, and does not necessarily require finding a commensurable metric.

In Section 2.1, we argued that commensurability was a problem for industry because system engineers would like to be able to say that the system they built is somehow better than other systems that they, or even their competitors, could have designed. However, system engineers can still say this without having to answer the question of how their system is doing relative to others. If they have established best practices for improving any dimension of their system, they can be assured that they have sought the optimal design.

The claim here is that the research community can benefit from focusing less on the question of relative performance and more on the question of how to improve dialog systems. Instead of trying to find commensurable metrics, we propose that the field should seek empirical and experimental evidence of factors that can improve any dialog metric, such as SASSI, regardless of domain or task. By doing so, the research community has more chance to influence industry best practices.

4.2 Proposal

In order to answer the question of how best to improve dialog systems, we propose pooling data from both research and industry to conduct meta-analyses. Meta-analysis, which is widely used in biomedicine and behavioral sciences, is the statistical analysis of a large collection of results from individual studies for the purpose of integrating the findings (Glass et al., 1981). By synthesizing results of related studies, the combined weight of evidence can be applied.

Meta-analysis for improving dialog systems involves three tasks: attribute identification, data coding, and statistical analysis. Attribute identification entails identifying all attributes of dialog systems that may have any effect on an evaluation metric of interest. For example, minute details such as the gender of the voice output, average and median word length of prompts, average latency to respond, etc. may influence metrics like task completion time. Once attributes have been identified, data pooled from research and industry can be coded by them, and once the data has been coded, it will not only be possible to conduct the kind of psychometric validation and reliability testing of metrics that distinguished SASSI, but also determine through correlation, regression and hypothesis-testing what system attributes influence any particular metric of interest, regardless of domain or task. For example, suppose SASSI scores are collected for a system. For each user interaction, a data entry would consist of the SASSI score, any other evaluation metrics of interest (e.g., task completion time), attributes of the system (e.g., gender of voice) and system interaction (e.g., number of confirmations used), and perhaps even attributes of the user (e.g., age group). Now imagine that every dialog system deployed provides this kind of data. With this data, it would be possible to learn, for instance, that prompts that flatter the user are consistently correlated with high SASSI likeability scores across all commercial and research systems. This provides a basis for empirically establishing best practices.

In order for this proposal to work, a large amount of data is required. Because the number of dialog systems built in research pales in comparison to the hundreds of systems that are commercially deployed in industry each year, researchers must work with system engineers to utilize the same metrics (e.g., the same questionnaires) and to code and pool data. While this task may seem Herculean, the result is of equal benefit to both research and industry: best practices for improving dialog systems that are empirically established.

5 Conclusion & Future Work

In this paper, we have examined the different ways in dialog evaluation is approached in research and industry. We critically surveyed four dialog evaluation metrics that could be considered candidates for commensurability. In light of problems faced by these metrics, and in reassessing the value of commensurability, we proposed a collaborative agenda for dialog evaluation based on using statistical meta-analysis for empirically establishing best practices from any evaluation metric. A meta-analysis is forthcoming as future work.

References

Balentine, B. & Morgan, D. 2001. How to Build a Speech Recognition Application: Second Edition: A Style Guide for Telephony Dialogues, Enterprise Integration Group.

Cohen, M., Giangola, J., Balogh, J. 2004. Voice Use Interface Design. Addison-Wesley Professional.

Dybkjaer, L. & Bernsen, N. 2001. Usability evaluation in spoken language dialogue systems. In ACL Workshop on Evaluation Methodologies for Language and Dialogue Systems.

Eckert, W., Levin, E., & Pieraccini, R. 1998. Automatic evaluation of spoken dialogue systems. In TWLT13, pp.99-110.

Frostad, K. 2003. Best practices in designing speech user interfaces.http://msdn.microsoft.com/library/en-us/dnnetspeech/html/vuibstprcf.asp?frame=true

Gibbon, D., Moore, R. & Winski, R. (Eds.). 1998. Handbook of standards and resources for spoken language systems. Walter de Bruyter, Berlin.

Glass, J., Polifroni, J., Seneff, S. & Zue, V. 2000. Data collection and performance evaluation of spoken dialogue systems: The MIT experience. In Proc. ICSLP.

Glass, G.; McGaw, B.; & Smith, M. 1981. Meta-analysis in Social Research. Beverly Hills: SAGE.

Hartikainen, M., Salonen, E. & Turunen, M. 2004. Subjective Evaluation of Spoken Dialogue Systems Using SERVQUAL Method. In Proc. Interspeech, pp.2273-2276.

Hone, K. & Graham, R. 2000. Towards a tool for the subjective assessment of speech system interfaces (SASSI). NLE, 6(3-4): 287-303.

Lamel, L., Minker, W., & Paroubek, P. 2000. Towards best practices in the development and evaluation of speech recognition components of a spoken language dialog system. NLE, 6(3-4): 305-322.

Kamm, C. Walker, M. & Litman, D. 1999. Evaluating spoken language systems. In Proc. AVIOS.

Nass, C. & Brave, S. 2005. Wired for Speech: How Voice Activates and Advances the Human-Computer Relationship. Cambridge, MA: MIT Press.

Paek, T. 2001. Empirical methods for evaluating dialog systems. In ACL Workshop on Evaluation Methodologies for Language and Dialogue Systems.

Pieraccini, R. & Huerta, J. 2005. Where do we go from here? Research and commercial spoken dialog systems. In Proc. SIGDIAL, pp. 1-10.

Pieraccini, R. & Lubensky, D. 2005. Spoken language communication with machines: the long and winding road from research to business. In Proc. IEA/AIE, pp. 6-15.

Stuttle, M., Williams, J. & Young, S. 2004. A framework for dialogue data collection with a simulated ASR channel. In Proc. Interspeech, pp. 241-244.

Walker, M., Litman, D., Kamm, C., & Abella, A. 1997 PARADISE: A general framework for evaluating spoken dialogue agents. In Proc. ACL/EACL, pp. 271-280.

Walker, M., Kamm, C. & Litman, D. 2000. Towards developing general models of usability with PARADISE. NLE, 6(3-4): 363-377.

Experiments on the France Telecom 3000 Voice Agency corpus: academic research on an industrial spoken dialog system[*]

Géraldine Damnati
France Télécom R&D
TECH/SSTP/RVA
2 av. Pierre Marzin
22307 Lannion Cedex 07, France
geraldine.damnati@orange-ftgroup.com

Frédéric Béchet **Renato De Mori**
LIA
University of Avignon
AGROPARC, 339 ch. des Meinajaries
84911 Avignon Cedex 09, France
frederic.bechet,renato.demori
@univ-avignon.fr

Abstract

The recent advances in speech recognition technologies, and the experience acquired in the development of WEB or Interactive Voice Response interfaces, have facilitated the integration of speech modules in robust Spoken Dialog Systems (SDS), leading to the deployment on a large scale of speech-enabled services. With these services it is possible to obtain very large corpora of human-machine interactions by collecting system logs. This new kinds of systems and dialogue corpora offer new opportunities for academic research while raising two issues: How can academic research take profit of the system logs of deployed SDS in order to build the *next generation* of SDS, although the dialogues collected have a dialogue flow constrained by the *previous SDS generation*? On the other side, what immediate benefits can academic research offer for the improvement of deployed system? This paper addresses these aspects in the framework of the deployed France Telecom 3000 Voice Agency service.

[*]This work is supported by the 6th Framework Research Programme of the European Union (EU), Project LUNA, IST contract no 33549. The authors would like to thank the EU for the financial support. For more information about the LUNA project, please visit the project home-page, www.ist-luna.eu.

1 Introduction

Since the deployment on a very large scale of the AT&T *How May I Help You?* (HMIHY) (Gorin et al., 1997) service in 2000, Spoken Dialogue Systems (SDS) handling a very large number of calls are now developed from an industrial point of view. Although a lot of the remaining problems (robustness, coverage, etc.) are still spoken language processing research problems, the conception and the deployment of such state-of-the-art systems mainly requires knowledge in user interfaces.

The recent advances in speech recognition technologies, and the experience acquired in the development of WEB or Interactive Voice Response interfaces have facilitated the integration of speech modules in robust SDS.

These new SDS can be deployed on a very large scale, like the France Telecom 3000 Voice Agency service considered in this study. With these services it is possible to obtain very large corpora of human-machine interactions by collecting system logs. The main differences between these corpora and those collected in the framework of evaluation programs like the DARPA ATIS (Hemphill et al., 1990) or the French Technolangue MEDIA (Bonneau-Maynard et al., 2005) programs can be expressed through the following dimensions:

- Size. There are virtually no limits in the amount of speakers available or the time needed for collecting the dialogues as thousands of dialogues are automatically processed every day and the system logs are stored. Therefore Dialog processing becomes similar to Broadcast News processing: the limit is not in the amount of data available, but rather in the amount of data that can be manually annotated.

- Speakers. Data are from *real* users. The speakers are not professional ones or have no reward for calling the system. Therefore their behaviors are not biased by the acquisition protocols. Spontaneous speech and speech affects can be observed.

- Complexity. The complexity of the services widely deployed is necessarily limited in order to guarantee robustness with a high automation rate. Therefore the dialogues collected are often short dialogues.

- Semantic model. The semantic model of such deployed system is task-oriented. The interpretation of an utterance mostly consists in the detection of application-specific entities. In an application like the France Telecom 3000 Voice Agency service this detection is performed by hand-crafted specific knowledge.

The AT&T *HMIHY* corpus was the first large dialogue corpus, obtained from a deployed system, that has the above mentioned characteristics. A service like the France Telecom 3000 Voice Agency service has been developed by a user interface development lab. This new kind of systems and dialogue corpora offer new opportunities for academic research that can be summarized as follows:

- How can academic research take profit of the system logs of deployed SDS in order to build the *next generation* of SDS, although the dialogues collected have a dialogue flow constrained by the *previous SDS generation*?

- On the other side, what immediate benefits can academic research offer for the improvement of deployed system, while waiting for the *next SDS generation*?

This paper addresses these aspects in the framework of the deployed FT 3000 Voice Agency service. Section 3 presents how the ASR process can be modified in order to detect and reject Out-Of-Domain utterances, leading to an improvement in the understanding performance without modifying the system. Section 4 shows how the FT 3000 corpus can be used in order to build stochastic models that are the basis of a new Spoken Language Understanding strategy, even if the current SLU system used in the FT 3000 service is not stochastic. Section 5 presents experimental results obtained on this corpus justifying the need of a tighter integration between the ASR and the SLU models.

2 Description of the France Telecom 3000 Voice Agency corpus

The France Telecom 3000 (*FT3000*) Voice Agency service, the first deployed vocal service at France Telecom exploiting natural language technologies, has been made available to the general public in October 2005. *FT3000* service enables customers to obtain information and purchase almost 30 different services and access the management of their services. The continuous speech recognition system relies on a bigram language model. The interpretation is achieved through the *Verbateam* two-steps semantic analyzer. *Verbateam* includes a set of rules to convert the sequence of words hypothesized by the speech recognition engine into a sequence of concepts and an inference process that outputs an interpretation label from a sequence of concepts.

2.1 Specificities of interactions

Given the main functionalities of the application, two types of dialogues can be distinguished. Some users call FT 3000 to activate some services they have already purchased. For such demands, users are rerouted toward specific vocal services that are dedicated to those particular tasks. In that case, the *FT3000* service can be seen as a unique automatic frontal desk that efficiently redirects users. For such dialogues the collected corpora only contain the interaction prior to rerouting. It can be observed in that case that users are rather familiar to the system and are most of the time regular users. Hence, they are more likely to use short utterances, sometimes just keywords and the interaction is fast (between one or two dialogue turns in order to be redirected to the demanded specific service).

Such dialogues will be referred as *transit* dialogues and represent 80% of the calls to the *FT3000* service. As for the 20% other dialogues, referred to as *other*, the whole interaction is proceeded within the *FT3000* application. They concern users that are more generally asking for information about a given service or users that are willing to purchase a new service. For these dialogues, the average utterance length is higher, as well as the average number of dialogue turns.

	other	**transit**
# dialogues	350	467
# utterances	1288	717
# words	4141	1454
av. dialogue length	3.7	1.5
av. utterance length	3.2	2.0
OOV rate (%)	3.6	1.9
disfluency rate (%)	2.8	2.1

Table 1: Statistics on the *transit* and *other* dialogues

As can be observed in table 1 the fact that users are less familiar with the application in the *other* dialogues implies higher OOV rate and disfluency rate[1]. An important issue when designing ASR and SLU models for such applications that are dedicated to the general public is to be able to handle both naive users and familiar users. Models have to be robust enough for new users to accept the service and in the meantime they have to be efficient enough for familiar users to keep on using it. This is the reason why experimental results will be detailed on the two corpora described in this section.

2.2 User behavior and OOD utterances

When dealing with real users corpora, one has to take into account the occurrence of Out-Of-Domain (OOD) utterances. Users that are familiar with a service are likely to be efficient and to strictly answer the system's prompts. New users can have more diverse reactions and typically make more comments about the system. By comments we refer to such cases when a user can either be surprised *what am I supposed to say now?*, irritated *I've already said that* or even insulting the system. A critical aspect for *other* dialogues is the higher rate of comments uttered by users. For the *transit* dialogues this phenomenon is much less frequent because users are familiar to the system and they know how to be efficient and how to reach their goal. As shown in table 2, 14.3% of the *other* dialogues contain at least one OOD comment, representing an overall 10.6% of utterances in these dialogues.

[1]by disfluency we consider here false starts and filled pauses

	other	**transit**
# dialogues	350	467
# utterances	1288	717
# OOD comments	137	24
OOD rate (%)	10.6	3.3
dialogues with OOD (%)	14.3	3.6

Table 2: Occurrence of Out-Of-Domain comments on the *transit* and *other* dialogues

Some utterances are just comments and some contain both useful information and comments. In the next section, we propose to detect these OOD sequences and to take this phenomenon into account in the global SLU strategy.

3 Handling Out-Of-Domain utterances

The general purpose of the proposed strategy is to detect OOD utterances in a first step, before entering the Spoken Language Understanding (SLU) module. Indeed standard Language Models (LMs) applied to OOD utterances are likely to generate erroneous speech recognition outputs and more generally highly noisy word lattices from which it might not be relevant and probably harmful to apply SLU modules.

Furthermore, when designing a general interaction model which aims at predicting dialogue states as proposed in this paper, OOD utterances are as harmful for state prediction as can be an out-of-vocabulary word for the prediction of the next word with an n-gram LM.

This is why we propose a new composite LM that integrates two sub-LMs: one LM for transcribing in-domain phrases, and one LM for detecting and deleting OOD phrases. Finally the different SLU strategies proposed in this paper are applied only to the portions of signal labeled as in-domain utterances.

3.1 Composite Language Model for decoding spontaneous speech

As a starting point, the comments have been manually annotated in the training data in order to easily separate OOD comment segments from in-domain ones. A specific bigram language model is trained for these comment segments. The comment LM was designed from a 765 words lexicon and trained on 1712 comment sequences.

This comment LM, called LM^{OOD} has been integrated in the general bigram LM^{G}. Comment sequences have been parsed in the training corpus and replaced by a *_OOD_* tag. This tag is added to the general LM vocabulary and bigram probabilities $P(\_OOD\_|w)$ and $P(w|\_OOD\_)$ are trained along with other bigram probabilities (following the principle of *a priori* word classes). During the decoding process, the general bigram LM probabilities and the LM^{OOD} bigram probabilities are combined.

3.2 Decision strategy

Given this composite LM, a decision strategy is applied to select those utterances for which the word lattice will be processed by the SLU component. This decision is made upon the one-best speech recognition hypotheses and can be described as follows:

1. If the one-best ASR output is a single *_OOD_* tag, the utterance is simply rejected.

2. Else, if the one-best ASR output contains an *_OOD_* tag along with other words, those words are processed directly by the SLU component, following the argument that the word lattice for this utterance is likely to contain noisy information.

3. Else (i.e. no *_OOD_* tag in the one-best ASR output), the word-lattice is transmitted to further SLU components.

It will be shown in the experimental section that this pre-filtering step, in order to decide whether a word lattice is worth being processed by the higher-level SLU components, is an efficient way of preventing concepts and interpretation hypothesis to be decoded from an uninformative utterance.

3.3 Experimental setup and evaluation

The models presented are trained on a corpus collected thanks to the *FT3000* service. It contains real dialogues from the deployed service. The results presented are obtained on the test corpus described in section 2.

The results were evaluated according to 3 criteria: the Word Error Rate (WER), the Concept Error Rate (CER) and the Interpretation Error Rate (IER). The CER is related to the correct translation of an utterance into a string of basic concepts. The IER is related to the global interpretation of an utterance in the context of the dialogue service considered. Therefore this last measure is the most significant one as it is directly linked to the performance of the dialogue system.

IER	**all**	**other**	**transit**
size	2005	717	1288
LM^{G}	16.5	22.3	13.0
LM^{G+OOD}	15.0	18.6	12.8

Table 3: Interpretation error rate according to the Language Model

Table 3 presents the IER results obtained with the strategy **strat1** with 2 different LMs for obtaining $\hat{W}$: LM^{G} which is the general word bigram model; and LM^{G+OOD} which is the LM with the OOD comment model. As one can see, a very significant improvement, 3.7% absolute, is achieved on the *other* dialogues, which are the ones containing most of the comments. For the *transit* dialogues a small improvement (0.2%) is also obtained.

4 Building stochastic SLU strategies

4.1 The FT3000 SLU module

The SLU component of the *FT3000* service considered in this study contains two stages:

1. the first one translates a string of words $W = w_1, \ldots, w_n$ into a string of elementary concepts $C = c_1, \ldots, c_l$ by means of hand-written regular grammars;

2. the second stage is made of a set of about 1600 inference rules that take as input a string of concepts C and output a global interpretation γ of

a message. These rules are ordered and the first match obtained by processing the concept string is kept as the output interpretation.

These message interpretations are expressed by an attribute/value pair representing a function in the vocal service.

The models used in these two stages are manually defined by the service designers and are not stochastic. We are going now to present how we can use a corpus obtained with such models in order to define an SLU strategy based on stochastic processes.

4.2 Semantic knowledge representation

The actual *FT3000* system includes semantic knowledge represented by hand-written rules. These rules can also be expressed in a logic form. For this reason, some basic concepts are now described with the purpose of showing how logic knowledge has been integrated in a first probabilistic model and how it can be used in a future version in which optimal policies can be applied.

The semantic knowledge of an application is a *knowledge base* (KB) containing a set of logic formulas. Formulas return truth and are constructed using constants which represent objects and may be typed, *variables*, *functions* which are mappings from tuples of objects to objects and *predicates* which represent relations among objects. An *interpretation* specifies which objects, functions and relations in the domain are represented by which symbol. Basic *inference problem* is to determine whether $KB \models F$ which means that KB entails a formula F.

In SLU, interpretations are carried on by binding variables and instantiating objects based on ASR results and inferences performed in the KB. Hypotheses about functions and instantiated objects are written into a Short Term Memory (STM).

A user goal is represented by a conjunction of predicates. As dialogue progresses, some predicates are grounded by the detection of predicate tags, property tags and values. Such a detection is made by the interpretation component. Other predicates are grounded as a result of inference. A user goal G is asserted when all the atoms of its conjunction are grounded and asserted true.

Grouping the predicates whose conjunction is the premise for asserting a goal G_i is a process that goes through a sequence of states: $S_1(G_i), S_2(G_i), \ldots$

Let Γ_k^i be the content of the STM used for asserting the predicates grounded at the *k-th* turn of a dialogue. These predicates are part of the premise for asserting the *i-th* goal.

Let G_i be an instance of the *i-th* goal asserted after grounding all the predicates in the premise.

Γ_k^i can be represented by a composition from a partial hypothesis Γ_{k-1}^i available at turn $k-1$, the machine action a_{k-1} performed at turn $k-1$ and the semantic interpretation γ_k^i i.e.:

$$\Gamma_k^i = \chi\left(\gamma_k^i, a_{k-1}, \Gamma_{k-1}^i\right)$$

$S_k(G_i)$ is an information state that can lead to a user's goal G_i and Γ_k^i is part of the premise for asserting G_i at turn k.

State probability can be written as follows:

$$P\left(S_k(G_i)|Y_k\right) = P\left(G_i|\Gamma_k^i\right) P\left(\Gamma_k^i|Y_k\right) \qquad (1)$$

where $P\left(G_i|\Gamma_k^i\right)$ is the probability that G_i is the type of goal that corresponds to the user interaction given the grounding predicates in Γ_k^i. Y_k is the acoustic features of the user's utterance at turn k.

Probabilities of states can be used to define a belief of the dialogue system.

A first model allowing multiple dialog state sequence hypothesis is proposed in (Damnati et al., 2007). In this model each dialog state correspond to a system state in the dialog automaton. In order to deal with flexible dialog strategies and following previous work (Williams and Young, 2007), a new model based on a Partially Observable Markov Decision Process (POMDP) is currently studied.

If no dialog history is taken into account, $P\left(\Gamma_k^i|Y\right)$ comes down to $P\left(\gamma_k^i|Y\right)$, γ_k^i being a semantic attribute/value pair produced by the Verbateam interpretation rules.

The integration of this semantic decoding process in the ASR process is presented in the next section.

5 Optimizing the ASR and SLU processes

With the stochastic models proposed in section 4, different strategies can be built and optimized. We are interested here in the integration of the ASR and SLU processes. As already shown by previous studies (Wang et al., 2005), the traditional sequential approach that first looks for the best sequence of words

$\hat{W}$ before looking for the best interpretation $\hat{\gamma}$ of an utterance is sub-optimal. Performing SLU on a word lattice output by the ASR module is an efficient way of integrating the search for the best sequence of words and the best interpretation. However there are real-time issues in processing word lattices in SDS, and therefore they are mainly used in research systems rather than deployed systems.

In section 3 a strategy is proposed for selecting the utterances for which a word lattice is going to be produced. We are going now to evaluate the gain in performance that can be obtained thanks to an integrated approach on these selected utterances.

5.1 Sequential *vs.* integrated strategies

Two strategies are going to be evaluated. The first one (*strat1*) is fully sequential: the best sequence of word $\hat{W}$ is first obtained with

$$\hat{W} = \underset{W}{\mathit{argmax}} P(W|Y)$$

Then the best sequence of concepts $\hat{C}$ is obtained with

$$\hat{C} = \underset{C}{\mathit{argmax}} P(C|\hat{W})$$

Finally the interpretation rules are applied to $\hat{C}$ in order to obtain the best interpretation $\hat{\gamma}$.

The second strategy (*strat2*) is fully integrated: $\hat{\gamma}$ is obtained by searching at the same time for $\hat{W}$ and $\hat{C}$ and $\hat{\gamma}$. In this case we have:

$$\hat{\gamma} = \underset{W,C,\gamma}{\mathit{argmax}} P(\gamma|C)P(C|W)P(W|Y)$$

The stochastic models proposed are implemented with a Finite State Machine (FSM) paradigm thanks to the AT&T FSM toolkit (Mohri et al., 2002).

Following the approach described in (Raymond et al., 2006), the SLU first stage is implemented by means of a word-to-concept transducer that translates a word lattice into a concept lattice. This concept lattice is rescored with a Language Model on the concepts (also encoded as FSMs with the AT&T GRM toolkit (Allauzen et al., 2003)).

The rule database of the SLU second stage is encoded as a transducer that takes as input concepts and output semantic interpretations γ. By applying this transducer to an FSM representing a concept lattice, we directly obtain a lattice of interpretations.

The SLU process is therefore made of the composition of the ASR word lattice, two transducers (word-to-concepts and concept-to-interpretations) and an FSM representing a Language Model on the concepts. The concept LM is trained on the *FT3000* corpus.

This strategy push forward the approach developped at AT&T in the *How May I Help You?* (Gorin et al., 1997) project by using richer semantic models than call-types and named-entities models. More precisely, the 1600 Verbateam interpretation rules used in this study constitute a rich knowledge base. By integrating them into the search, thanks to the FSM paradigm, we can jointly optimize the search for the best sequence of words, basic concepts, and full semantic interpretations.

For the strategy *strat1* only the best path is kept in the FSM corresponding to the word lattice, simulating a sequential approach. For *strat2* the best interpretation $\hat{\gamma}$ is obtained on the whole concept lattice.

error	*WER*	*CER*	*IER*
strat1	40.1	24.4	15.0
strat2	38.2	22.5	14.5

Table 4: Word Error Rate (WER), Concept Error Rate (CER) and Interpretation Error Rate (IER) according to the SLU strategy

The comparison among the two strategies is given in table 4. As we can see a small improvement is obtained for the interpretation error rate (IER) with the integrated strategy (*strat2*). This gain is small; however it is interesting to look at the Oracle IER that can be obtained on an n-best list of interpretations produced by each strategy (the Oracle IER being the lowest IER that can be obtained on an n-best list of hypotheses with a perfect Oracle decision process). This comparison is given in Figure 1. As one can see a much lower Oracle IER can be achieved with *strat2*. For example, with an n-best list of 5 interpretations, the lowest IER is 7.4 for *strat1* and only 4.8 for *strat2*. This is very interesting for dialogue systems as the Dialog Manager can use dialogue context information in order to filter such n-best lists.

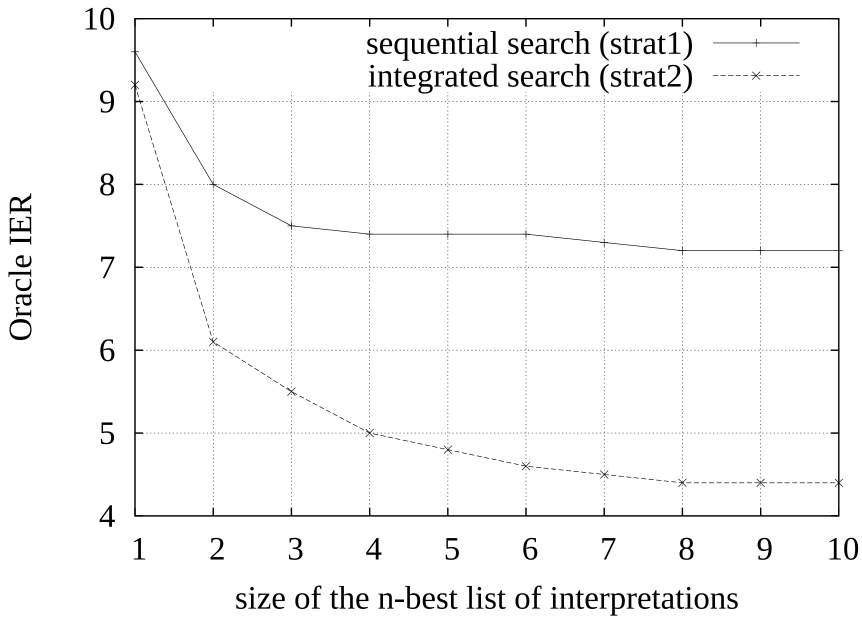


Figure 1: Oracle IER according to an n-best list of interpretations for strategies *strat1* and *strat2*

5.2 Optimizing WER, CER and IER

Table 4 also indicates that the improvements obtained on the WER and CER dimensions don't always lead to similar improvements in IER. This is due to the fact that the improvements in WER and CER are mostly due to a significant reduction in the insertion rates of words and concepts. Because the same weight is usually given to all kinds of errors (insertions, substitutions and deletions), a decrease in the overall error rate can be misleading as interpretation strategies can deal more easily with insertions than deletions or substitutions. Therefore the reduction of the overall WER and CER measures is not a reliable indicator of an increase of performance of the whole SLU module.

level	*1-best*	*Oracle hyp.*
WER	33.7	20.0
CER	21.2	9.7
IER	13.0	4.4

Table 5: Error rates on words, concepts and interpretations for the 1-best hypothesis and for the Oracle hypothesis of each level

These results have already been shown for WER by previous studies like (Riccardi and Gorin, 1998) or more recently (Wang et al., 2003). They are illustrated by Table 5 and Table 6. The figures shown in these tables were computed on the subset of utterances that were passed to the SLU component. Utterances for which an OOD has been detected are discarded. In Table 5 are displayed the error rates obtained on words, concepts and interpretations both on the 1-best hypothesis and on the Oracle hypothesis (the one with the lowest error rate in the lattice). These Oracle error rates were obtained by looking for the best hypothesis in the lattice obtained at the corresponding level (e.g. looking for the best sequence of concepts in the concept lattice). As for Table 6, the mentioned IER are the one obtained when applying SLU to the Oracles hypotheses computed for each level. As one can see the lowest IER (4.4) is not obtained on the hypotheses with the lowest WER (9.8) or CER (7.5).

	IER
from word Oracle	9.8
from concept Oracle	7.5
interpretation Oracle	4.4

Table 6: IER obtained on Oracle hypotheses computed at different levels.

6 Conclusion

This paper presents a study on the *FT3000* corpus collected from real users on a deployed general public application. Two problematics are addressed: How can such a corpus be helpful to carry on research on advanced SLU methods eventhough it has been collected from a more simple rule-based dialogue system? How can academic research translate into short-term improvements for deployed services? This paper proposes a strategy for integrating advanced SLU components in deployed services. This strategy consists in selecting the utterances for which the advanced SLU components are going to be applied. Section 3 presents such a strategy that consists in filtering Out-Of-Domain utterances during the ASR first pass, leading to significant improvement in the understanding performance.

For the SLU process applied to in-domain utterances, an integrated approach is proposed that looks simultaneously for the best sequence of words, concepts and interpretations from the ASR word lattices. Experiments presented in section 5 on real data show the advantage of the integrated approach towards the sequential approach. Finally, section 4 proposes a unified framework that enables to define a dialogue state prediction model that can be applied and trained on a corpus collected through an already deployed service.

References

Cyril Allauzen, Mehryar Mohri, and Brian Roark. 2003. Generalized algorithms for constructing statistical language models. In *41st Annual Meeting of the Association for Computational Linguistics (ACL'03), Sapporo, Japan*.

Helene Bonneau-Maynard, Sophie Rosset, Christelle Ayache, Anne Kuhn, and Djamel Mostefa. 2005. Semantic annotation of the french media dialog corpus. In *Proceedings of the European Conference on Speech Communication and Technology (Eurospeech)*, Lisboa, Portugal.

Geraldine Damnati, Frederic Bechet, and Renato De Mori. 2007. Spoken Language Understanding strategies on the France Telecom 3000 voice agency corpus. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Honolulu, USA.

A. L. Gorin, G. Riccardi, and J.H. Wright. 1997. How May I Help You ? In *Speech Communication*, volume 23, pages 113–127.

Charles T. Hemphill, John J. Godfrey, and George R. Doddington. 1990. The ATIS spoken language systems pilot corpus. In *Proceedings of the workshop on Speech and Natural Language*, pages 96–101, Hidden Valley, Pennsylvania.

Mehryar Mohri, Fernando Pereira, and Michael Riley. 2002. Weighted finite-state transducers in speech recognition. *Computer, Speech and Language*, 16(1):69–88.

Christian Raymond, Frederic Bechet, Renato De Mori, and Geraldine Damnati. 2006. On the use of finite state transducers for semantic interpretation. *Speech Communication*, 48,3-4:288–304.

Giuseppe Riccardi and Allen L. Gorin. 1998. Language models for speech recognition and understanding. In *Proceedings of the International Conference on Spoken Langage Processing (ICSLP)*, Sidney, Australia.

Ye-Yi Wang, A. Acero, and C. Chelba. 2003. Is word error rate a good indicator for spoken language understanding accuracy? In *Automatic Speech Recognition and Understanding workshop - ASRU'03*, St. Thomas, US-Virgin Islands.

Ye-Yi Wang, Li Deng, and Alex Acero. 2005. Spoken language understanding. In *Signal Processing Magazine, IEEE*, volume 22, pages 16–31.

Jason D. Williams and Steve Young. 2007. Partially observable markov decision processes for spoken dialog systems. *Computer, Speech and Language*, 21:393–422.

Experiences of an In-Service Wizard-of-Oz Data Collection for the Deployment of a Call-Routing Application

Mats Wirén,[1] Robert Eklund,[1] Fredrik Engberg[2] and Johan Westermark[2]

[1]Research and Development
TeliaSonera
SE–123 86 Farsta, Sweden

[2]Customer Integrated Solutions
TeliaSonera
SE–751 42 Uppsala, Sweden

firstname.lastname@teliasonera.com

Abstract

This paper describes our experiences of collecting a corpus of 42,000 dialogues for a call-routing application using a Wizard-of-Oz approach. Contrary to common practice in the industry, we did not use the kind of automated application that elicits some speech from the customers and then sends all of them to the same destination, such as the existing touch-tone menu, without paying attention to what they have said. Contrary to the traditional Wizard-of-Oz paradigm, our data-collection application was fully integrated within an existing service, replacing the existing touch-tone navigation system with a simulated call-routing system. Thus, the subjects were real customers calling about real tasks, and the wizards were service agents from our customer care. We provide a detailed exposition of the data collection as such and the application used, and compare our approach to methods previously used.

1 Background and introduction

Spoken-dialogue systems for applications such as customer care increasingly use statistical language models (SLMs) and statistically-based semantic classification for recognition and analysis of utterances. A critical step in designing and deploying such a system is the initial data collection, which must provide a corpus that is both representative of the intended service and sufficiently large for development, training and evaluation.

For at least 20 years, Wizard-of-Oz methodology has been regarded as a superior (though not unproblematic) method of collecting high-quality, machine-directed speech data in the absence of a runnable application.[1] Normally, these data will be useful for several purposes such as guiding dialogue design and training speech recognizers. Still, the Wizard-of-Oz option is often dismissed in favour of simpler methods on the ground that it does not scale well in terms of cost and time (for example, Di Fabbrizio et al. 2005). Consequently, Wizard-of-Oz has typically been used for data collections that are more limited in the number of subjects involved or utterances collected. One exception from this is the data collection for the original AT&T "How May I Help You" system (Gorin et al. 1997; Ammicht et al. 1999), which comprised three batches of transactions with live customers, each involving up to 12,000 utterances. Other well-known instances are "Voyager" (Zue et al. 1989) and the individual ATIS collections (Hirschman et al. 1993) which involved up to a hundred subjects or (again) up to 12,000 utterances.

While it is true that Wizard-of-Oz is a labour-intensive method, the effort can often be motivated on the ground that it enables significant design and evaluation to be carried out before implementation, thereby reducing the amount of re-design necessary for the actual system. However, one should also bear in mind the crucial advantage brought about by the possibility in a production environment of running the Wizard-of-Oz collection *in-service* rather than in a closed lab setting. As we shall discuss, the fact that real customers with real problems are involved instead of role-playing subjects with artificial tasks circumvents the key methodological problem that has been raised as an argument against Wizard-of-Oz, namely, lack of realism.

[1] For backgrounds on Wizard-of-Oz methodology, see Dahlbäck et al. (1993) and Fraser & Gilbert (1991).

The aim of this paper is to describe our experiences of running a Wizard-of-Oz collection in a production environment with real customers, with the double purpose of guiding dialogue design and collecting a sufficient amount of data for the first training of a speech recognizer. We also review what other options there are for the initial data collection and compare our Wizard-of-Oz approach with those.

The rest of this paper is organized as follows: Section 2 describes the call-routing problem and our particular domain. Section 3 gives an overview of the options for the initial data collection and the major trade-offs involved in selecting a method. Section 4 describes the application that was developed for our Wizard-of-Oz data collection, whereas Section 5 describes the actual data collection, summary statistics for the collected data and some experimental results obtained. Section 6 contains a discussion of our overall experiences.

2 The call-routing task and domain

Call routing is the task of directing a caller to a service agent or a self-serve application based on their description of the issue. Increasingly, speech-enabled routing is replacing traditional touch-tone menues whereby callers have to navigate to the appropriate destinations.

The domain of interest in this paper is (the entrance to) the TeliaSonera[2] residential customer care in Sweden, comprising the entire range of services offered: fixed and mobile telephony, broadband and modem-based Internet, IP telephony, digital television, triple play, etc. Around 14 million calls are handled annually, and before the speech-enabled call-routing system was launched in 2006, touch-tone navigation was used throughout. The speech-enabled system involves an SLM-based speech recognizer and a statistically-based classifier.[3] The task of the classifier is to map a spoken utterance to an application category which corresponds to a self-serve application, (a queue to) a human agent, a disambiguation category or a discourse category. Whereas self-serve applications and service agents are the desired goals to reach, disambiguation and discourse categories correspond to intermediate states in the routing dialogue. More specifically, disambiguation categories correspond to cases where the classifier has picked up *some* information about the destination, but needs to know more in order to route the call. Discourse categories correspond to domain-independent utterances such as greetings ("Hi, my name is John Doe"), channel checks ("Hello?") and meta questions ("Who am I talking to?"). Altogether, there are 124 application categories used by the current classifier.

3 Options for initial data collection

Basically, there are three options for making the initial data collection for a call-routing application: to collect human–human dialogues in a call center, to use an automated data-collection application, or to use a Wizard-of-Oz approach. We shall now describe each of these.

3.1 Human–human dialogues

The simplest possible approach to the initial data collection is to record conversations between service agents and customers in a call center. This is an inexpensive method since it does not require any data-collection application to be built. Also, there is no customer impact. However, the data obtained tend not to be sufficiently representative, for two reasons: First, typically only a subset of the services of a call center is carried out by human agents, and hence many services will not be covered. Second, the characteristics of human–human conversations differ from those of human–machine interaction. Still, this option has sometimes been preferred on the grounds of simplicity and lack of negative customer impact.

3.2 Automated applications

Due to the nature of the task, it is easy to put out a fully automated mock-up system in a live service that engages in the initial part of a call-routing dialogue. Typically, such a system will play an open prompt, record the customers' speech, play another prompt saying that the system did not understand, again record the speech, and finally direct all calls to a single destination, such as a general-skills service agent or the entry to the existing touch-tone menu. We estimate that a system of this kind could be implemented and integrated into a call center in about a person week. An example of this approach is the AT&T "Ghost Wizard" (referred to in Di Fabbrizio et al. 2005).

[2] TeliaSonera (**www.teliasonera.com**) is the largest telco in the Scandinavian –Baltic region.

[3] The speech recognizer and classifier are delivered by Nuance (**www.nuance.com**).

This basic approach can be improved upon by detecting silences and touch-tone events, and in these cases playing designated prompts that try to get the caller on track. Furthermore, if data from previous call-routing applications are available, it is possible to use these to handle domain-independent utterances. Such utterances correspond to discourse categories as mentioned in Section 2, and the idea then is to play prompts that encourage the caller to describe the issue. A description of such an approach is provided by Di Fabbrizio et al. (2005).

A problem with the automated approach is that customer impact can be quite negative, since the application does not actually do anything except for recording their speech (possibly through several turns), and routing them to a "dummy" destination where they will have to start over. Of course, one way of avoiding this is to include a human in the loop who listens to the customer's speech and then routes the call to the right destination. Apparently, this is the approach of Di Fabbrizio et al. (2005), which consequently is not fully automated.

Apart from customer impact, the problem with an automated system is that we do not learn the full story about caller behaviour. In particular, since typically only a minority of callers will state their issue in an unambiguous way within the given few turns, less information about the callers' actual issues will be obtained. In particular, for callers who completely fail to speak or who give no details about their issue, we will have no possibility of finding out what they wanted and why they failed. Furthermore, since the system lacks the ability to respond intelligently to in-domain utterances, no follow-up dialogue such as disambiguation can be collected.

3.3 Wizard-of-Oz

Although Wizard-of-Oz is arguably the best method for collecting machine-directed data in the absence of a running application, it is not without methodological problems. The basic critique has always been aimed at the lack of realism (for example, von Hahn 1986). In a thorough analysis, Allwood & Haglund (1992) point out that in a Wizard-of-Oz simulation, both the subjects and the wizard(s) are playing roles, occupied and assigned. The researcher acting as the wizard is occupying the role of a researcher interested in obtaining "as natural as possible" language and speech data, while playing the role of the system. The subject, on the other hand, is occupying the role of a subject in a scientific study, and playing the role of a client (or similar), communicating with a system while carrying out tasks that are not genuine to the subject, but given to them by the experiment leader (who might be identical with the wizard).

It turns out, however, that a traditional Wizard-of-Oz approach with made-up tasks according to a scenario is anyway not an option when collecting data for deploying a call-routing system. The reason for this is that we want to learn not just *how* callers express themselves, but also *what kind of tasks they have*, which obviously rules out pre-written scenarios. If the existing system uses touch-tone navigation, usually not too much can be ascertained about this, and trying to design a set of tasks just by looking at the existing destinations would miss the point.

By instead integrating a Wizard-of-Oz application in an existing, live service, we can circumvent the key methodological problems, while addressing all the problems of the previously described approaches and even obtaining some independent advantages:

1. Since the callers' experience will be like that of the intended application, albeit with human speech understanding, the customer impact will be at least as good. In fact, it is even possible to issue a kind of guarantee against maltreatment of customers by instructing the wizards to take over calls that become problematic (this is further discussed in Section 4).
2. Since real customers are involved, no role-playing from the point of view of the subjects takes place, and hence the data become highly realistic.
3. The fact that scenarios are superfluous—or even run counter to the goal of the data collection—means that the main source of methodological problems disappears, and that the data collection as such is considerably simplified compared to traditional Wizard-of-Oz.
4. By letting service agents be wizards, we move away even further from role-playing, given that the interaction metaphor in speech-enabled call routing is natural-language dialogue with a (general-skills) service agent.

5. Service agents possess the expertise necessary for a call-routing wizard: they know when additional information is required from the caller, when a call is ready for routing, and where to actually route the call. Hence, wizard guidelines and training become less complex than in traditional Wizard-of-Oz.[4]
6. Service agents have excellent skills in dealing with customers. Hence, during the data collection they will be able to provide valuable feedback on dialogue and prompt design that can be carried over to the intended application.

In spite of these advantages, Wizard-of-Oz appears to have been used only very rarely for collecting call-routing data. The sole such data collection that we are aware of was made for the original AT&T "How May I Help you" system (Gorin et al. 1997; Ammicht et al. 1999). The one disadvantage of the Wizard-of-Oz approach is that it is more laborious than automated solutions, mainly because several person months of wizard work is required. On the other hand, as we have seen, it is still less laborious than a traditional Wizard-of-Oz, since there are no scenarios and since wizard guidelines can be kept simple.

4 Data-collection application

Our data-collection application consists of two parts: The first part is the Prompt Piano Client (PPC), which is running on the service agent's PC. This is essentially a GUI with "keys" corresponding to prerecorded prompts by which the wizard interacts with the caller, thereby simulating the intended system. The PPC interface is shown in PLATE 1. The second part is the Prompt Piano Server (PPS), which is an IVR (interactive voice response) server with a Dialogic telephony board, running Envox, Nuance and Dialogic software. This handles playing of prompts as well as recording of calls. Two kinds of recordings are made: call logs (that is, the callers' speech events as detected by the Nuance speech recognizer) and complete dialogues ("open mic").

To set up a data collection, the contact center solution is modified so that a percentage of the incoming calls to the customer care is diverted to the PPS. The PPS in turn transfers each call to a wizard (that is, to a PPC) using tromboning. Allocation of the wizards is performed by the Telia CallGuide contact center platform using skill-based routing. Whenever a wizard answers a call, two audio streams are established, one from the customer to the wizard so that she can hear the customer's speech, and one from an audio source in the PPS to the customer. An initial open prompt is played automatically by the PPS, and the wizard is then free to start playback of prompts. This is realized by sending control messages from the PPC to the audio source on the PPS via TCP/IP, while listening to the customer throughout.

Depending on the caller's response, different things will happen: If the caller provides an unambiguous description of the issue, the wizard will transfer the call to the correct queue and end the recording by pressing the "end / route customer" button. This signals to the PPS that the call should be released using the Explicit Call Transfer (ECT) supplementary service, freeing the two channels used for the tromboned call in the PPS.

If, on the other hand, the caller does not provide an unambiguous description of the issue, the wizard will play a follow-up prompt aimed at getting more information from the caller by choosing from the buttons/prompts situated to the right (fields II and III of the GUI; see Plate 1). These parts of the GUI are fully configurable; the number and layout of buttons as well as the names of sound files for the corresponding prompts are declared separately. (Declarations include specifying whether the prompt associated with a particular button allows barge-in or not.) Thus, it is possible not just to vary individual prompts, but also to simulate call-routing dialogues to various depths by varying the number of buttons/prompts.

Apart from routing the call, a possible action of the wizard is to enter into the call. This is realized by establishing a two-way direct audio stream with the customer, enabling the parties to talk to each other. As pointed out in Section 3.3, one purpose of this is to let wizards take over calls that are problematic, thereby making sure that callers do not get maltreated during the data collection and reducing the risk that they hang up. A similar functionality was available in the data-collection application for AT&Ts "How May I Help You" system (Walker et al. 2000).

[4] Furthermore, as a side effect, it is possible to facilitate the subsequent process of manually tagging the data by keeping track of where each call is routed.

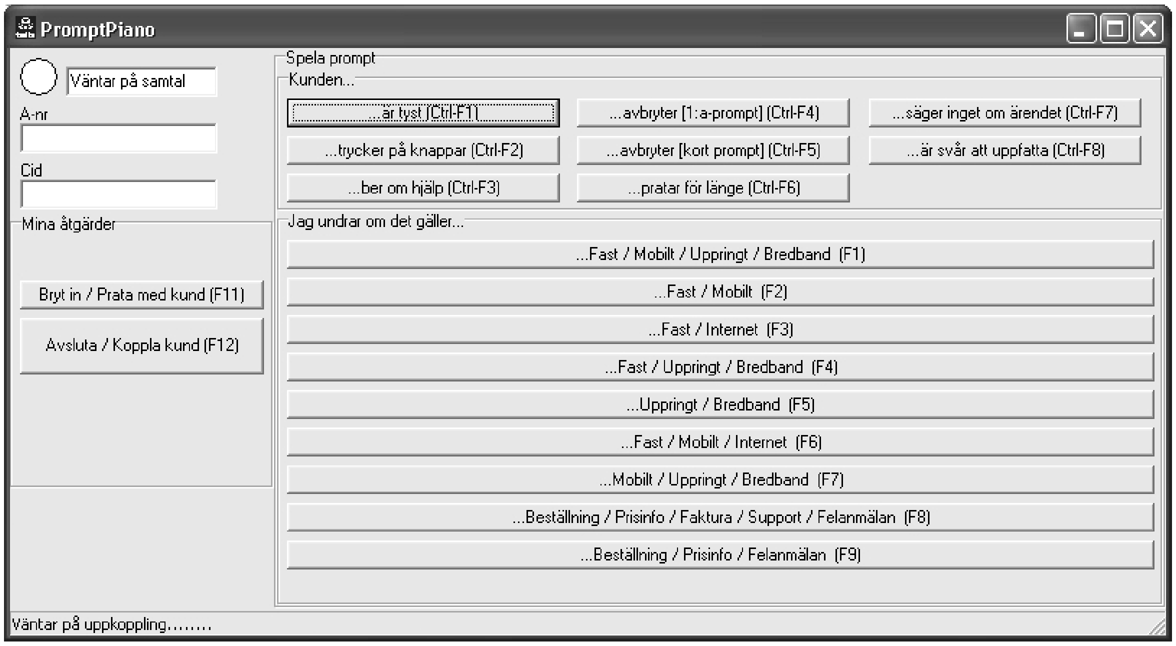

PromptPiano
Väntar på samtal
A-nr
Cid
Mina åtgärder
Bryt in / Prata med kund (F11)
Avsluta / Koppla kund (F12)
Spela prompt
Kunden...
...är tyst (Ctrl-F1)
...avbryter [1:a-prompt] (Ctrl-F4)
...säger inget om ärendet (Ctrl-F7)
...trycker på knappar (Ctrl-F2)
...avbryter [kort prompt] (Ctrl-F5)
...är svår att uppfatta (Ctrl-F8)
...ber om hjälp (Ctrl-F3)
...pratar för länge (Ctrl-F6)
Jag undrar om det gäller...
...Fast / Mobilt / Uppringt / Bredband (F1)
...Fast / Mobilt (F2)
...Fast / Internet (F3)
...Fast / Uppringt / Bredband (F4)
...Uppringt / Bredband (F5)
...Fast / Mobilt / Internet (F6)
...Mobilt / Uppringt / Bredband (F7)
...Beställning / Prisinfo / Faktura / Support / Felanmälan (F8)
...Beställning / Prisinfo / Felanmälan (F9)
Väntar på uppkoppling........

PLATE 1: The Prompt Piano Client interface as configured towards the end of the data collection. The interface is divided into three fields with buttons. *I:* The leftmost field provides caller information, like **A-nr** (the phone number the customer is calling from) and **Cid** (the phone number the customer provides as reason for the call). The wizard has two option buttons, **Mina åtgärder** ('my actions'), at hand: the button **Bryt in / Prata med kund** ('barge-in/talk to client') which is used for entering into the call, and the button **Avsluta / Koppla kund** ('end/route customer') which is used to terminate the recording prior to routing the call to the appropriate destination. (Both of these options are associated with prompts being played.) *II:* The second field, **Kunden…** ('the customer…'), contains buttons corresponding to renewed open prompts for the purpose of error-handling, **…är tyst** ('… is silent'), **…trycker på knappar** ('uses the touch-tone keypad'), **…ber om hjälp** ('asks for help'), **…avbryter** ('interrupts'), **…pratar för länge** ('talks for too long'), **…säger inget om ärendet** ('doesn't say anything about the reason for the call'), **…är svår att uppfatta** ('is hard to understand'). *III:* The third field, **Jag undrar om det gäller…** ('I would like to know if it is about…'), contains buttons corresponding to disambiguation prompts asking for additional information, e.g. whether the customer's reason for the call is about fixed ('fast') or mobile ('mobilt') telephony, broadband ('bredband') or something else. All buttons also have hot-key possibilities for agents who prefer this over point-and-click.

With the exception of the initial open prompt, the wizards have full control over when and in what order prompts are played and actions are executed. Thus, whereas an automated system will start playing the next prompt after an end-of-speech timeout typically within the range of 0.75–1.5 seconds, a wizard may decide to impose longer delays if she considers that the caller has not yet yielded the turn. On the other hand, the wizard may also respond more rapidly. Thus, the problem of response delays, which has sometimes had distorting impact in Wizard-of-Oz simulations, does not appear in our application (cf. Oviatt et al. 1992).

The PPS application was developed in the Envox graphical scripting language, which makes it possible to write event-driven applications controlled from an external source such as the PPC, and also supports Nuance call logging (for recording customer utterances) and Dialogic transaction recording (for recording entire conversations between two parties, in this case the customer and the PPS, or the customer and the wizard).[5] Design, implementation and testing of the Prompt Piano (PPC and PPS) took four person weeks.

The agents/wizards were involved in the development from the very start to ensure that the application (and in particular the GUI) was

[5] VXML was not used since it appeared that real-time control of an IVR from an external source would then have been more difficult to implement. Furthermore, VXML browsers generally have no support for features such as transaction recording during tromboned transfer and delayed invocation of the ECT supplementary service in conjunction with call transfer. Hence, in a VXML framework, additional components would be required to solve these tasks.

optimized according to their needs and wishes. The Prompt Piano GUI was reconfigured several times during the course of the data collection, both for the purpose of carrying out prompt-design experiments and in response to (individual or group) requests for changes by the agents/wizards.

5 Data collection

5.1 Overview

The purpose of the data collection was twofold: to obtain speech data that could be used for initial training of the speech recognizer, and to obtain data that could be used to guide dialogue design of the intended application. Thus, whereas the former only involved caller responses to open prompts, the latter required access to complete call-routing dialogues, including error-handling and disambiguation.

Organization. Ten wizards were used for the data collection. Initially, one week was used for training of the wizards and basic tuning of the prompts. This process required four person weeks (not all wizards were present all the time). After a break of three weeks, the data collection then went on for five weeks in a row, with the ten wizards acquiring around 42,000 call-routing dialogues. (This figure includes around 2,000 useable dialogues that were collected during the initial week.) This was more than had been anticipated, and much more than the 25,000 that had been projected as a minimum for training, tuning and evaluation of the speech recognizer. Thus, although 50 person weeks were used by the wizards for the actual collection, 32 person weeks would actually have been sufficient to reach the minimum of 25,000 dialogues. On average, 195 dialogues were collected per working day per wizard (mean values ranging from 117 dialogues per day to 317 dialogues per day; record for a wizard on a single day was 477).

Barge-in. Initially, barge-in was allowed for all prompts. However, it turned out to be useful to have one very short prompt with barge-in disabled, just asking the caller to state the reason for the call. The main usage of this was in cases where callers were repeatedly barging in on the system to the extent that the system could not get its message through.

Utterance fragments. As a consequence of, on the one hand, wizards having full control over when and whether to start playing a prompt and, on the other hand, the speech recognizer having a fixed end-of-speech timeout, it would sometimes happen that more than one sound file would be recorded between two prompts in the Nuance call logs. An example of this would be: "Eeh, I... I'm wondering whether... can you tell me the pricing of broadband subscriptions?", where both of the two silent pauses would trigger the end-of-speech timeout. Although this constitutes a mismatch between data collection and final system, in practice this caused no problem: on the contrary, the sound files were simply treated as separate utterances for the purpose of training the speech recognizer, which means that the most informative fragment, typically at the end, was not lost. In addition, these data are potentially very valuable for research on turn-taking (in effect, intelligent end-of-speech detection).

Wizards entering into calls. The event of wizards taking over calls in order to sort out problematic dialogues occurred on average in 5% of the calls. The figure was initially a bit higher, presumably because the wizards were less skillful in using the prompts available, and because the prompts were less well-developed. As a side-effect of this, we have obtained potentially very valuable data for error-handling, with both human–machine and human–human data for the same callers and issues (compare Walker et al., 2000).

Post-experimental interviews. We also used the facility of letting wizards take over calls as a way of conducting post-experimental interviews. This was achieved by having wizards route the calls to themselves and then handle the issue, whereupon the wizard would ask the caller if they would accept being interviewed. In this way, we were able to assess customer satisfaction on the fly with respect to the intended system and even getting user feedback on specific design features already during the data collection.

5.2 Experiments

Several design experiments were run during the data collection. Here, we shall only very briefly describe one of them, in which we compared two styles of disambiguation prompts, one completely open and one more directed. As can be seen in TABLE 1, utterances following the open disambiguation prompt are on average 3.6 times longer than utterances following the directed prompt.

Prompt	Utterances and Words			Disfluency			Concepts					
	Utts	Words	Words /Utts	Disfl	Disfl /Utts	Disfl /Words	Concepts In	Concepts Out	DIFFs Total	DIFFs Change	DIFFS /Utts	DIFFS /Words
Directed	118	216	1.8	19	0.16	0.09	136	244	108	0	0.9	0.5
Open	121	791	6.5	72	0.6	0.09	144	248	122	18	1.01	0.15

TABLE 1. Summary statistics for the **directed prompt** ('I need some additional information about the reason for your call. Is it for example about an order, price information or support?'), and the **open prompt** ('Could you please tell me a little bit more about the reason for you call?') prompts. Totals and ratios are given for utterances/words, disfluencies and number of concepts acquired before the disambiguation prompt was played ("In") and after the customer had replied to the disambiguation prompt ("Out"). Also, ratios are given for number of concepts compared to number of utterances and words, as well as totals and ratios for the differences (DIFFs) between concepts in and concepts out, i.e., how many concepts you "win" by asking the disambiguation prompt.

Furthermore, in order to see to what extent these prompts also made callers provide more information, we manually tagged the transcribed utterances with semantic categories. Following the evaluation methodology suggested by Boye & Wirén (2007, Section 5), we then computed the difference with respect to "concepts" for utterances immediately following and preceding the two kinds of prompts.

Although the number of concepts gained is only slightly higher[6] for the open prompt (as a function of concepts per utterance), there are some palpable differences between the directed and the open prompt. One, shown in TABLE 1, is that there are no instances where an already instantiated concept (e.g. `fixedTelephony`) is changed to something else (e.g. `broadband`), while this happens 18 times following the open prompt. The other, not shown in TABLE 1, is that, following the directed prompt, one never "gains" more than one new concept, while there are 26 instances following the open prompt where the gain is two concepts, and even two instances where the gain is three concepts (which also means that one concept is changed).

Finally, when one analyses the syntactic characteristics following the two different types of prompts, there is an obvious shift from the telegraphic "noun-only" responses that amount to more than 70% of the directed prompt responses, to the responses following the open prompt, where 40% are complete sentences and 21% are noun phrases. Also, the syntax is more varied following the open prompt.[7]

6 Discussion

We claimed in Section 3.3 that by using an in-service Wizard-of-Oz data collection, we have been able to effectively overcome all problems of the alternative methods discussed there. A relevant question is then if there are any remaining, independent problems of the approach described here.

On the methodological side, there is clearly a certain amount of role playing left in the sense that service agents are acting as the system (albeit a system whose interaction metaphor is a service agent!). Interestingly, we noticed early on that the agents sometimes failed in properly simulating the intended system in one respect: Since they would often grasp what the caller wanted before he or she had finished speaking, they would start playing the next prompt so early that they were barging in on the caller. Thus, in their willingness to provide quick service, they were stepping outside of their assigned role. However, they soon learnt to avoid this, and it was never a problem except for the first few days.

Apart from this, the main disadvantage of Wizard-of-Oz collections clearly is the amount of work involved compared to the other methods. As we have seen, the Prompt Piano design and implementation took four person weeks, training of the wizards took another four person weeks, and collection of 25,000 dialogues required 32 person weeks—hence altogether 40 person weeks (although we actually used 50 person weeks, since we went on collecting more data). This could be compared with possibly a single person week required for the fully automated approach. The more

[6] However, the difference is not statistically significant, either using a *t* test (two-sampled, two-tailed: *p*=0.16 with equal variances assumed; *p*=0.158 equal variances not assumed) or Mann-Whitney *U* test (two-tailed: *p*=0.288).

[7] The distributions are, in descending order, for the **directed prompt**: Noun=85, Sentence=11, Yes/No=8, Noun Phrase=8, no response=3, Yes/No+Noun=2, Adverbial Phrase=1, Adjective Phrase=1; for the **open prompt**: Sentence=49, Noun Phrase=26, Noun=24, Verb Phrase=11, Adjective Phrase=5, Adverbial Phrase=2, no response=2, Yes/No=1, Interjection=1.

elaborate automated methods would come somewhere in between, also depending on whether a human agent is used for routing callers or not.

In the TeliaSonera case, the main desiderata favouring Wizard-of-Oz were highly representative data, no negative customer impact and need for early evaluation and design, particularly because this was the first deployment of natural-language call routing in Scandinavia. In other words, it was decided to accept a higher *initial* cost in return for reduced costs downstream, due to higher quality and less re-design of the implemented system.

It is impossible to quantify the downstream savings made by choosing Wizard-of-Oz since we have no baseline. However, one indication of the quality of the data is the initial performance of the classifier of the deployed system. (By "initial", we mean the period during which no data from the live system had yet been used for training or updating of the system.) In our case, the initial accuracy was 75%, using 113 application categories. We regard this as a high figure, also considering that it was achieved in spite of several new products having been introduced in the meantime that were not covered by the speech recognizer. The initial training of the speech recognizer and classifier used 25,000 utterances. As a comparison, when an additional 33,000 utterances (mostly from the live system) had been used for training, the accuracy increased to 85%.

Acknowledgements

Many colleagues have provided invaluable help and support throughout this project. Here we can only mention some of them: Johan Boye, Joakim Gustafson, Linda Bell, Fredrik Byström, Robert Sandberg, Erik Näslund, Erik Demmelmaier, Viktoria Ahlbom, Inger Thall and Marco Petroni. Last but not least we thank our skilled wizards: Christina Carlson, Marie Hagdorn, Gunilla Johannisson, Ana-Maria Loriente, Maria Mellgren, Linda Norberg, Anne Tärk, Mikael Wikner, Eva Wintse and Jeanette Öberg.

References

Allwood, Jens & Björn Haglund. 1992. Communicative Activity Analysis of a Wizard of Oz Experiment. Internal Report, PLUS ESPRIT project P5254.

Ammicht, Egbert, Allen Gorin & Tirso Alonso. 1999. Knowledge Collection For Natural Language Spoken Dialog Systems. *Proc. Eurospeech*, Budapest, Hungary, Volume 3, pp. 1375–1378.

Boye, Johan & Mats Wirén. 2007. Multi-slot semantics for natural-language call routing systems. *Proc. Bridging the Gap: Academic and Industrial Research in Dialog Technology. NAACL Workshop*, Rochester, New York, USA.

Dahlbäck, Nils, Arne Jönsson & Lars Ahrenberg, Wizard of Oz Studies — Why and How. 1993. *Knowledge-Based Systems*, vol. 6, no. 4, pp. 258–266. Also in: Mark Maybury & Wolfgang Wahlster (eds.). 1998. *Readings in Intelligent User Interfaces*, Morgan Kaufmann.

Di Fabbrizio, Giuseppe, Gokhan Tur & Dilek Hakkani-Tür. 2005. Automated Wizard-of-Oz for Spoken Dialogue Systems. *Proc. Interspeech*, Lisbon, Portugal, pp. 1857–1860.

Fraser, Norman M. & G. Nigel Gilbert. Simulating speech systems. 1991. *Computer Speech and Language*, vol. 5, pp. 81–99.

Gorin, A. L., G. Riccardi & J. H. Wright. 1997. How may I help you? *Speech Communication*, vol. 23, pp. 113–127.

von Hahn, Walther. 1986. Pragmatic considerations in man–machine discourse. *Proc. COLING*, Bonn, Germany, pp. 520–526.

Hirschman, L., M. Bates, D. Dahl, W. Fisher, J. Garofolo, D. Pallett, K. Hunicke-Smith, P. Price, A. Rudnicky & E. Tzoukermann. 1993. Multi-Site Data Collection and Evaluation in Spoken Language Understanding. *Proc. ARPA Human Language Technology*, Princeton, New Jersey, USA, pp. 19–24 .

Oviatt, Sharon, Philip Cohen, Martin Fong & Michael Frank. 1992. A rapid semi-automatic simulation technique for investigating interactive speech and handwriting. *Proc. ICSLP*, Banff, Alberta, Canada, pp. 1351–1354.

Walker, Marilyn, Irene Langkilde, Jerry Wright, Allen Gorin & Diane Litman. 2000. Learning to Predict Problematic Situations in a Spoken Dialogue System: Experiments with How May I Help You? *Proc. North American Meeting of the Association for Computational Linguistics* (*NAACL*), pp. 210–217.

Zue, Victor, Nancy Daly, James Glass, David Goodine, Hong Leung, Michael Phillips, Joseph Polifroni, Stephanie Seneff & Michael Soclof. 1989. The Collection and Preliminary Analysis of a Spontaneous Speech Database. *Proc. DARPA Speech and Natural Language Workshop*, pp. 126–134.

AdaRTE: An Extensible and Adaptable Architecture for Dialog Systems

L.M. Rojas-Barahona
Dip. Informatica e Sistemistica
Università di Pavia
Pavia, 27100, IT
linamaria.rojas@unipv.it

T. Giorgino
Dip. Informatica e Sistemistica
Università di Pavia
Pavia, 27100, IT
toni.giorgino@unipv.it

Abstract

Dialog Systems have been proven useful to provide the general public with access to services via speech devices. In this paper, we present AdaRTE, an Adaptable Dialog Architecture and Runtime Engine. AdaRTE uses dynamic Augmented Transition Networks and enables the generation of different backend formats; for instance, it supports VoiceXML generation to guarantee portability and standards compliance. The scope of AdaRTE is to provide a ground for deploying complex adaptable dialogs such as those found in the patient-care domain, and for experimenting with innovative speech solutions including Natural Language Processing. AdaRTE is an extensive architecture for dialog representation and interpretation, which helps developers to layout dialog interactions through a high level formalism whilst allowing the inclusion of voice applications best-practices.

1 Introduction

Dialog technologies have been widely applied in different domains. Previous to Voice Browsers (VB), proprietary technology was the response to vocal applications deployment. The speech systems adopted either custom code, or proprietary dialog-manager based solutions. Linear script, state transition networks and plan-based were among the available technologies for dialog management systems. Generally, the deployment of any of these techniques requires heavily scripted solutions. On the other hand, the multitude of dialog technology vendors naturally resulted in a proliferation of incompatible languages across vendors and platforms.

More recently, the advent of VoiceXML allows to deploy dialog systems in a Web-based environment (McGlashan et al., 2004). Its delivery contributed to reduce the proliferation of incompatible dialog formalisms by offering one standard for voice applications. However, VoiceXML has inherent limitations which are well analyzed in (Mittendorfer et al., 2002), such as its declarative and static structure, difficulty accessing remote resources (databases and ontologies) and lack of means for efficient and heavy computation. Furthermore, VoiceXML does not allow an explicit visualization of the dialog flow because of its form-filling mechanism and, like web based technologies, has to be generated by other code dynamically.

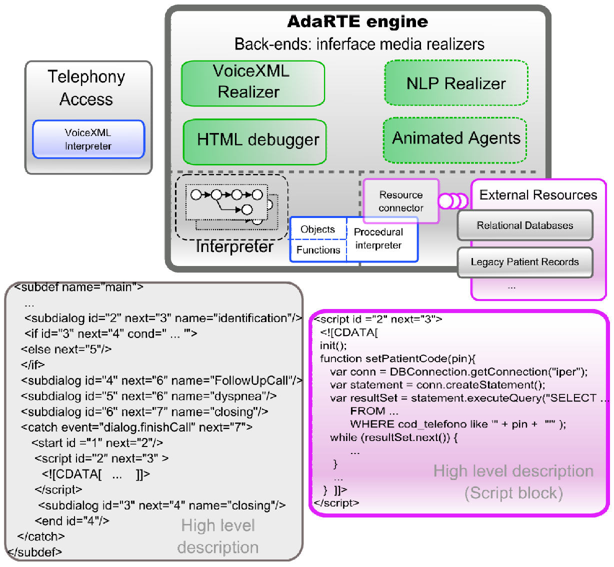
Perhaps the strongest limit pointed out by the research community is that VoiceXML does not directly support neither dynamic natural language understanding and generation, nor multimodality. As a consequence, extensions to VoiceXML has been proposed in literature: DialogXML was applied to car telematics services; in this approach, the VB was extended to support NLP KANTOO generated grammars (Hataoka et al., 2004). Other VoiceXML-generative approaches are presented in (Hamerich et al., 2004) which follows a database-oriented approach, and (Di Fabbrizio and Lewis, 2004) which is seemingly targeted towards customer care tasks with sophisticated call routing, rather than the structured enquiry data collection tasks found in chronic patient management. We believe that a big effort should still be done in adapting dialog systems best practices (Balentine and Morgan, 2001), such as confirmation strategy, adaptability, mixed initiative, usable speech interfaces for users and graphical interfaces for developers in VoiceXML-based frameworks. Commonly, the process of deploying dialog systems was complicated, costly, time demanding and required speech technology experts. A leaner development methodology is particularly necessary when considering domains in which available resources for development are limited; such a case is that of the health care domain, in which voice applications have been used for several home-care interventions successfully. (Young et al., 2001; Giorgino et al., 2004; Bickmore et al., 2006).

In this paper, we present an architecture devised to overcome all these issues. Features were thought to reduce dialog system development effort through reuse, support for hierarchical network specification, adaptable decision takers and best practices adoption. We built AdaRTE, which implements these features for dialog deployment, and we present results obtained through the partial prototyping of two telephony-linked systems: the first inspired by the Chronic Obstructive Pulmonary Disease (COPD) care (Young et al., 2001), and the second by the Homey dialog system for hypertensive patient home management (Giorgino et al., 2004). Our effort was mainly focused on health care dialogs systems, since our solution is targeted at offering a standards-compliant way of deploying dialog systems, whose additional peculiarities are extensibility, support for complex dialog flows and low-cost development.

2 AdaRTE Architecture

The architecture we propose (figure 1) is primarily composed of a dialog interpreter, a runtime engine and an interface media realizer for backends generation. A running system interacts with users which can be grouped in three main role categories: Application developers, patients and case managers, i.e. case manager nurses.

The dialog flow and structure are represented in a well-defined XML formalism (XML dialog description).



```
<subdef name="main">
   ...
   <subdialog id="2" next="3" name="identification"/>
   <if id="3" next="4" cond=" ... "">
  <else next="5"/>
  </if>
  <subdialog id="4" next="6" name="FollowUpCall"/>
  <subdialog id="5" next="6" name="dyspnea"/>
  <subdialog id="6" next="7" name="closing"/>
  <catch event="dialog.finishCall" next="7">
    <start id ="1" next="2"/>
     <script id="2" next="3" >
       <![CDATA[   ...   ]]>
     </script>
      <subdialog id="3" next="4" name="closing"/>
     <end id="4"/>
 </catch>
</subdef>
```

```
<script id ="2" next="3">
  <![CDATA[
  init();
  function setPatientCode(pin){
    var conn = DBConnection.getConnection("iper");
    var statement = conn.createStatement();
    var resultSet = statement.executeQuery("SELECT ...
         FROM ...
         WHERE cod_telefono like '" + pin +  "'" );
    while (resultSet.next()) {
         ...
     }
     ...
   }  ]]>
</script>
```

Figure 1: AdaRTE architecture block diagram

tion). To cooperate with standards-based speech recognition software and respond to telephone-originated events, AdaRTE acts as a web server, generating VoiceXML or HTML code dynamically. Prompts, questions and other elements are the nodes (here named blocks) of an Augmented Transition Network (ATN) that specifies the flow of the conversation. Blocks are represented in the description by XML tags. When the system is started, the XML dialog description is read by AdaRTE which maintains an internal representation of the dialog, and executes it when a call comes in. Consequently, it activates the dialog blocks in sequence or according to a specific criterion, constructs prompts, interprets the answers returned by the caller through the voice platform, and interacts with external resources as appropriate.

Usually, an ATN is associated with a context and, here, we call this structure subdialog. When a call is setup, the main subdialog is retrieved and started; in its turn, it can invoke other subdialogs, and so forth. If the execution flow reaches the end of the main subdialog, the call is terminated. Subdialogs can also terminate unexpectedly if an exception occurs, and, in this case, an exception handler is executed.

In addition, we grant the application of *best-practices* through the configuration of thresholds and n-best lists confirmation strategy related to questions. Adaptability i.e. flexibility according to users experience with the system, is reached by using *containers*. They are used for common tasks, in which one of several subdialogs are selected according to a specified policy such as randomly, in sequence, ordered by call number, according to any externally defined schedule or a criterion based on reinforcement learning techniques. Inclusion of procedural code at user-level is essential for flexibility, interoperability, and ease of programming. AdaRTE allows to embed snippets of code, which are written in the ECMAScript standard language, into script blocks. These user-written code is run in a separate execution environment with extensive facilities and standard libraries. This also enables access to external resources, including databases, ontologies, or any other commodity library, i.e a probabilistic-based library.

Currently, semantic recognition is implemented either inside the engine itself or leveraging the context-free grammar (CFG) formats offered by the VB. However, we strongly believe in the necessity of integrating a more elaborated semantic recognition solution by supporting NLP and more expressive grammars. In addition, since the architecture of AdaRTE is extensible, it would be possible to integrate any other backend. For instance, we foresee the adoption of multimodality through the generation of an enriched markup language which would be understood by an external animation generator module. Work is in progress toward these directions.

Differing from other VoiceXML-generative frameworks, AdaRTE is oriented to the medical domain, which requires adaptable dialogs with complex structures. Also, it offers a new level of flexibility to developers by allowing external resources access inside script blocks. Moreover, the high level dialog description is intuitive, thus simple dialogs could be implemented by not expert authors. Finally, AdaRTE was thought to be a standard-compliant architecture for incremental adoption of voice formalisms, i.e. lexicalized grammar-based NLPs.

3 Results

Currently, the AdaRTE framework is operative. It has been beta-tested with two realistic health care dialog systems, derived by actual systems deployed and validated in the previous years. The first one is based on a prototype based on the TLC-COPD dialog deployed by the Boston MISU group and others (Young et al., 2001). For this specific example, we used Tellme Studio[1] as VSP. This pilot's deployment demanded less than two weeks of man effort. The fulfilled activities were database schema definition and data preparation together with the dialog deployment. This dialog is executed in English language and uses keypad touch-tone (DTMF) interaction.

The second test case is the partial reimplementation of the Homey dialog system. Homey had been deployed for the management of hypertensive patients (Giorgino et al., 2004). The system included an extensive Electronic Health Records system with storage of personal data and profiles, in order to support dialog adaptability. Reengineering part of the Homey proprietary dialog manager to the AdaRTE architecture took approximately three weeks. The development of this prototype involved the following activities: VSP evaluation, database definition and grammars and dialog deployment. Unlike the TLC-COPD pilot, this system uses speech rather than DTMF input. We built grammars by using the Nuance GSL language and SRGS grammar formats. The language of the dialog is Italian and the dialog was deployed by using Voxpilot as VSP[2]. The expressiveness of the dialog formalism yielded an important reduction of time invested in developing both prototypes whilst facilitating component reuse in each dialog.

4 Future enhancements

A large body of research on the optimization of spoken interfaces is available (Walker et al., 1997). Some of the results of the research have been condensed into best practices (Balentine and Morgan, 2001). For example, more complex confirmation strategies with respect to simple "yes/no" answers should be adopted. Inclusion of such techniques into custom-developed systems is complex. A big advantage in using the interpretable and high-level dialog representation language proposed in this work is that

[1]Tellme Studio. https://studio.tellme.com/

[2]VoxBuilder. http://www.voxbuilder.com/

such “dialog practices” can be incorporated seamlessly into the underlying dialog interpretation logic, removing the burden from the dialog developer.

Currently, we have a strong commitment on the integration of a more elaborated semantic interpretation mechanism by integrating AdaRTE with a NLP application that supports more expressive grammars. In this way, not only recognition does not depend on the grammars supported by VBs, but also more natural dialogs will be supported, so patients perception of the dialogs will improve. In addition, a multimodal extension of AdaRTE through the implementation of a facial expressions and gestures realizer should be considered for future research as well as the extension of automated discourse planning facilities.

5 Conclusion

We have presented an architecture for next-generation dialog representation and interpretation and built an engine for dialog deployment. AdaRTE supports high-level dialog representations whilst implicity takes care of aspects and best practices that are non considered in the current voice and multimodal standards i.e VoiceXML. It supports VoiceXML to communicate to VBs as one of the interpretation and generation backends. We have reengineered two health-care dialog prototypes, chosen as real world test cases, by using the novel architecture and showed that dialog development time is remarkably optimized with respect to customized coding.

The AdaRTE system is motivated not only as a reliable platform for dialog deployment, but also as a framework for incorporating advanced features of speech recognizers, including increased support to adaptability, natural language understanding and generation, and multimodality.

Acknowledgments

The authors thank Dr Robert Friedman (Medical Information Systems Unit, Boston, MA) for information on the TLC-COPD dialogs, and Timothy Bickmore (College of Computer and Information Science, Northeastern University, Boston, MA) for the discussions that led to the AdaRTE project.

References

Bruce Balentine and David P. Morgan. 2001. *How to Build a Speech Recognition Application. A Style Guide for Telephony Dialogues*, Second ed. EIG Press, San Ramon, CA.

Timothy Bickmore, Toni Giorgino, Nancy Green, Rosalind Picard 2006. Special Issue on Dialog Systems for Health Commu-nication. *Journal of Biomedical Informatics Elsevier*,39:465–467.

Giuseppe Di Fabbrizio and Charles Lewis. 2004. Florence: a Dialogue Manager Framework for Spoken Dialogue Systems. *In: Proceedings of the 8th International Conference on Spoken Language Processing (ICSLP 2004)*,Jeju, Jeju Island, Korea.

Toni Giorgino, Ivano Azzini, Carla Rognoni, Silvana Quaglini, Mario Stefanelli, Roberto Gretter and Daniele Falavigna. 2004. Automated spoken dialogue system for hyper-tensive patient home management. *International Journal of Medical Informatics. Elsevier*,74:159–167.

Hamerich Stefan, Schubert Volker, Schless Volker, de Cordoba Ricardo, Pardo Jose d'Haro Luis, Kladis Basilis, Kocsis Otilia and Igel Stefan. 2004. Semi-Automatic Generation of Dialogue Applications in the GEMINI Project. *Sigdial.*

Nobuo Hataoka, Yasunari Obuchi, Teruko Mitamura and Eric Nyberg. 2004. Robust speech dialog interface for car telematics service. *ieeecnc*,331–335.

Scott McGlashan, Burnett Daniel C, Carter Jerry, Danielsen Peter, Ferrans Jim, Hunt Andrew, Bruce Lucas, Porter Brad, Rehor Ken, Tryphonas Steph . 2004. Voice Extensible Markup Language (VoiceXML) Version 2.0. http://www.w3.org/TR/voicexml20/.

Markus Mittendorfer, Georg Niklfeld, Werner Winiwarter 2002. Making the voice web smarter-integrating intelligent component technologies and VoiceXML. *In: Proceedings of the Second International Conference on Web Information Systems Engineering*, 2:126-131.

Marilyn A. Walker, Diane J. Litman, Candace A. Kamm and Alicia Abella. 1997. *PARADISE: A Framework for Evaluating Spoken Dialogue Agents. In: Proceedings of ACL/EACL.*,271-280.

Melissa Young, David Sparrow, Daniel Gottlieb, Alfredo Selim, Robert H. Friedman. 2001. A telephone-linked computer system for COPD care. *Chest*,119:1565–1575.

Multi-slot semantics for natural-language call routing systems

Johan Boye and **Mats Wirén**
TeliaSonera R&D
Vitsandsgatan 9
SE-123 86 Farsta, Sweden
johan.boye@teliasonera.com, mats.wiren@teliasonera.com

Abstract

Statistical classification techniques for natural-language call routing systems have matured to the point where it is possible to distinguish between several hundreds of semantic categories with an accuracy that is sufficient for commercial deployments. For category sets of this size, the problem of maintaining consistency among manually tagged utterances becomes limiting, as lack of consistency in the training data will degrade performance of the classifier. It is thus essential that the set of categories be structured in a way that alleviates this problem, and enables consistency to be preserved as the domain keeps changing. In this paper, we describe our experiences of using a two-level multi-slot semantics as a way of meeting this problem. Furthermore, we explore the ramifications of the approach with respect to classification, evaluation and dialogue design for call routing systems.

1 Introduction

Call routing is the task of directing callers to a service agent or a self-service that can provide the required assistance. To this end, touch-tone menus are used in many call centers, but such menus are notoriously difficult to navigate if the number of destinations is large, resulting in many misdirected calls and frustrated customers. Natural-language call routing provides an approach to come to terms with these problems. The caller gets the opportunity to express her reasons for calling using her own words, whereupon the caller's utterance is automatically categorized and routed.

This paper focuses on experiences obtained from the deployment of a call-routing application developed for the TeliaSonera residential customer care.[1] The application was launched in 2006, replacing a previous system based on touch-tone menus. The customer care annually handles some 14 million requests and questions concerning a wide range of products in fixed telephony, mobile telephony, modem-connected Internet, broadband, IP telephony and digital TV.

The crucial step in any call routing application is classification, that is, the mapping of natural-language utterances to categories that correspond to routing destinations. Early systems used quite small numbers of categories. For example, the original "How May I Help You" system had 15 categories (Gorin et al. 1997), the system of Chu-Carroll and Carpenter (1999) had 23 categories, and Cox and Shahshahani (2001) had 32. Nowadays, it is possible to distinguish between several hundreds of categories with high accuracy (see, for example, Speech Technology Magazine 2004). The TeliaSonera system currently distinguishes between 123 categories with an accuracy of 85% (using a speech recognizer and classifier developed by Nuance[2]). Moreover, according to our experiments the same classification technology can be

[1] TeliaSonera (www.teliasonera.com) is the largest telecom operator in the Nordic–Baltic region in Europe.
[2] www.nuance.com.

used to distinguish between 1,500 categories with 80% accuracy.[3]

For large category sets like these, the problem of maintaining consistency among manually tagged utterances becomes limiting, as lack of consistency in the training data will degrade performance of the classifier. The problem is exacerbated by the fact that call-routing domains are always in a state of flux: Self-services are being added, removed, modified, split and merged. Organizational changes and product development regularly call for redefinitions of human expertise areas. All of these changes must be accommodated in the category set. Hence, it must be possible to update this set efficiently and at short intervals.

To meet this problem, it is crucial that the set of categories be structured in a way that facilitates the task of manual tagging and enables consistency to be preserved. However, in spite of the fact that the size of category sets for call routing have increased dramatically since the original "How May I Help You" system, we are not aware of any papers that systematically discuss how such large sets should be structured in order to be efficiently maintainable. Rather, many papers in the call-routing literature consider the call routing problem as an abstract classification task with atomic categories at a single level of abstraction. Such atomic categories are typically taken to correspond to departments and self-services of the organization to which the call center belongs. In a real-life implementation, the situation is often more complicated. At TeliaSonera, we have adopted a two-level multi-slot semantics as a way of maintaining modularity and consistency of a large set of categories over time.

The aim of this paper is to share our experiences of this by providing a detailed description of the approach and its implications for classification, dialogue design and evaluation. The rest of the paper is organized as follows: Section 2 describes the multi-slot category system. Sections 3–5 outline consequences of the multi-slot semantics for disambiguation, classification and evaluation, respectively. Section 6 concludes.

2 What's in a category?

2.1 Motivation

As pointed out above, call-routing domains are always to some extent moving targets because of constant changes with respect to products and organization. It would be cumbersome to manually re-tag old data each time the category set is updated. Retagging the training data for the statistical classifier might introduce inconsistencies into the training set and degrade classifier performance. Thus, it is a good idea to define *two* sets of categories at different levels; one set of *semantic* categories reflecting the contents of the utterance, and one set of *application* categories reflecting how the call should be handled. These two sets of categories are related by means of a many-to-one mapping from the semantic domain to the application domain. Figure 1 gives the general picture.

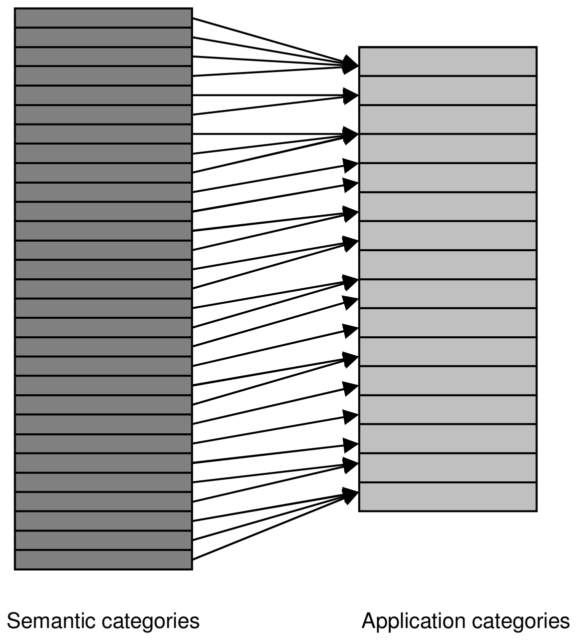


Figure 1: Mapping between semantic categories and application categories.

The utterances in the training set for the automatic classifier are manually categorized using semantic categories. The automatic classifier can be trained to work either in the semantic domain or in the application domain (see further Section 4).

[3] In both cases, the classifier was trained on 60,000 utterances.

2.2 Semantic categories

In the TeliaSonera system, semantic categories are triples of the form

(*family, intention, object*)

where *family* is the general product family which the call concerns (e.g. fixed telephony, mobile telephony, broadband, etc.), *intention* represents the nature of the request (e.g. order, want-info, change-info, activate, want-support, report-error, etc.), and *object* represents more specifically what the call is about (e.g. particular names of products, or concepts like “telephone number”, “SIM card”, or “password”). Currently there are 10 families, about 30 intentions, and about 170 objects that span the semantic domain.

Some (in fact, the majority) of the possible triples are disallowed because they are nonsensical. For instance, it is not meaningful to combine “fixed telephony” in the *family* slot with “SIM card” in the *object* slot. To cater for this, we have defined a set of combination rules weeding out the illegal combinations of values. These rules disallow about 80% of the possible combinations, leaving about 10,000 permissible semantic triples. Of these 10,000 triples, about 1,500 have actually turned up in real data.

The three-slot structure of categories is very useful when performing manual tagging of the training material for the statistical classifier. Although there are 10,000 categories, the person performing the tagging needs only to keep track of about 210 concepts (10 families + 30 intentions + 170 objects). In contrast, it is safe to say that an unstructured category system containing 10,000 atomic categories would be quite impractical to use.

In addition, the combination rules can further alleviate the manual tagging task. It is straightforward to implement a tagging tool that allows the human tagger to select a value for one semantic slot, and then restrict the selection for the other slots only to include the possible values. For example, if “fixed telephony” is chosen for the *family* slot, “SIM card” would not appear among the possible values for the *object* slot. This approach has been successfully adopted in the project.

2.3 Application categories

There is one application category for each type of action from the system. Actions come in two flavors; either the call is routed (in the cases where the caller has given sufficient information), or the system asks a counter-question in order to extract more information from the caller. That is, application categories can be labeled either as *routing categories* or *disambiguation categories.* For convenience, names of application categories are also triples, chosen among the set of semantic triples that map to that application category.

2.4 Information ordering

Each slot in a semantic triple can take the value *unknown*, representing the absence of information. For instance, the most accurate semantic category for the caller utterance “Broadband”[4] is (*broadband*, *unknown*, *unknown*), since nothing is known about the intention of the caller or the specific topic of the request. Thus, in the information ordering, “unknown” is situated below all other values.

There are also some intermediate values in the information ordering. The value *telephony* represents “either fixed telephony or mobile telephony”, and has been incorporated in the category set since many callers tend not be explicit about this point. In the same vein, *internet* represents “either broadband or modem-connected internet”, and *billing* represents the disjunction of a whole range of billing objects, some of which can be handled by a self-service and some can not.

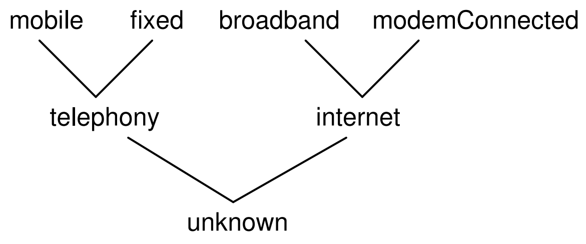


Figure 2: Parts of the semantic information ordering.

The information ordering extends naturally to triples. In particular, the triple (*unknown*, *unknown*,

[4] Many callers express themselves in this telegraphic fashion.

unknown) represents complete absence of information.

3 Disambiguation

The caller's request might be ambiguous in one sense or another, in which case the system will need to perform disambiguation by asking a follow-up question. This might either be a general question encouraging the user to describe his request in greater detail, or a directed question of the type "Would that be fixed telephony or mobile telephony?"

Ambiguous utterances might be represented in at least two fundamentally different ways. In vector-based approaches, routing destinations and input utterances alike are represented by vectors in a multi-dimensional space. An input utterance is routed to a specific destination if the vector representation of the utterance is close to that of the destination. An ambiguous utterance is characterized by the fact that the Euclidean distances from the utterance vector to the n closest routing destination vectors are roughly the same.

Chu-Carroll and Carpenter (1999) describe a method of disambiguation, where disambiguation questions are dynamically constructed on the basis of an analysis of the differences among the closest routing destination vectors. However, it is not clear that the disambiguation questions produced by their proposed method would make sense in all possible situations. Furthermore, their method does not take into account the fact that some ambiguities tend to be more important and arise more often than others. We think it is worthwhile to concentrate on these important cases (in terms of prompt design, speech recognition grammar construction, etc.), rather than trying to solve every conceivable ambiguity, most of which would never appear in real life.

As previously mentioned, in the TeliaSonera system we have chosen another way of treating ambiguities, namely that certain application categories are *disambiguation categories*; they represent foreseen, frequently occurring, ambiguous input utterances. The three-slot structure of categories provides a handy way of identifying ambiguous cases; they are represented by triples where one or more slots are *unknown,* or where some slot has an intermediate value, like *telephony* or *internet.* Examples of such ambiguous utterances are "broadband" (*broadband-unknown-unknown*) and "I want to have a telephone subscription" (*telephony-order-subscription*). All categories that represent ambiguities have pre-prepared disambiguation questions, speech recognition grammars, and dialogue logic to handle the replies from the callers.

Of course, there are still problematic cases where an utterance can not be assigned any unique category with any tolerable level of confidence, neither a routing category nor a disambiguation category. In those cases, the system simply rephrases the question: "*Sorry, I didn't quite understand that. Could you please rephrase?*"

4 Classification

4.1 Atomic vs. multi-slot classification

For the purpose of automatic classification of utterances, there are at least two different views one may adopt. In one view, the "atomic" view, the three-slot structure of category names is considered as merely a linguistic convention, convenient only when manually tagging utterances (as discussed in Section 2.1). When adopting this view, we still regard the categories to be distinct atomic entities as concerns automatic classification. For instance, to the human eye it is obvious that two categories like (*internet, order, subscription*) and (*broadband, order, subscription*) are related, but the automatic classifier just considers them to be any two categories, each with its separate set of training examples.

An alternative view, the "multi-slot view", is to see the category as actually consisting of three slots, each of which should be assigned a value independently. This means that a separate classifier is needed for each of the three slots.

It is not clear which view is preferable. An argument in favor of the multi-slot view is the following: If some categories have the same value in one slot, then these categories are semantically related in some way. Most likely this semantic relation is reflected by the use of common words and phrases; for instance, expressions like "order" and "get a new" presumably are indicative for all categories having the value *order* in the *intention* slot. Therefore, classifying each slot separately would be a way to take a priori semantic knowledge into account.

To this, proponents of the atomic view may respond that such similarities between categories would emerge anyway when using a single classifier that decides the entire semantic triple in one go (provided that enough training data is available). In addition, if each slot is categorized separately, it is not certain that the resulting three values would constitute a permissible semantic triple (as mentioned in Section 2.1, about 80% of the possible combinations are illegal). In contrast, if a single classifier is used, the result will always be a legal triple, since only legal triples appear in the training material.

The statistical classifier actually used in the live call routing system treats categories as atomic entities and, as mentioned in the introduction, it works well. The encouraging numbers bear out that the "atomic" view is viable when lots of data is at hand. On the other hand, if training data is sparse, one might consider using a hand-written, rule-based classifier, and in these cases the multi-slot view seems more natural.

4.2 Rule-based multi-slot classification

To obtain a baseline for the performance of the statistical classifier used in the live system, we implemented an alternative classifier that solves the classification task using hand-written rules. Thus, the purpose of this was to investigate the performance of a naïve classification method, and use that for comparison with other methods. In addition, the rule-based classifier provides an example of how the multi-slot approach can support the inclusion of human a priori domain knowledge into the classification process.

The rule-based classifier has three kinds of rules: Firstly, phrase-spotting rules associate a word or a phrase with a value for a semantic slot (i.e. a *family*, an *intention*, or an *object*). Rules of the second kind are domain axioms that encode invariant relationships, such as the fact that *object=SIMcard* implies *family=mobileTelephony*. Finally, rules of the third kind specify how semantic values can be combined into a legal semantic triple (these rules are also used for manual tagging, as mentioned in Section 2.1). Each semantic value is also (manually) given a score that reflects its information content; a higher score means that the value contains more information. For instance, the value *subscription* has a lower information score than have the names of specific subscription types that TeliaSonera offers its customers.

The classifier works in three phases, which we will demonstrate on a running example. In the first phase, it applies the phrase-spotting rules to the input sentence, returning a list of slot-value pairs. For instance, the input sentence "I want to order a new SIM card" would yield the list [*intention=order, object=SIMcard*], using rules triggering on the phrases "order" and "SIM card" in the input sentence.

Secondly, the classifier adds semantic components as a result of applying the domain axioms to members of the list. Using the domain axiom mentioned above, the semantic component *family=mobileTelephony* would be added to the list, due to the presence of *object=SIMcard.* Thus, after the two first phases, the intermediate result in this example is [*intention=order, object=SIMcard, family=mobileTelephony*].

In the final phase, semantic components are selected from the list to form a semantic triple. In the example, this step is straightforward since the list contains exactly one value for each component, and these values are combinable according to the combination rules. The final result is:

(*mobileTelephony, order, SIMcard*)

In cases where the semantic values in the list are not combinable (a situation often originating from a speech recognition error), one or several values have got to be relaxed to *unknown.* According to our experiments, the best heuristic is to first relax the *object* component and then the *intention* component. For example, in the list [*family = fixedTelephony, intention=order, object=SIMcard*], the first and third elements are not combinable; thus this list yields the triple:

(*fixedTelephony, order, unknown*)

In the case where some slots are not filled in with a value, the values of those slots are set to *unknown.* Thus, the list [*family=fixedTelephony, intention=order*] would also yield the semantic triple above.

Finally, consider the case where the input list contains more than one value for one or several slots. In this case, the algorithm picks the value with the highest information content score. For instance, consider the utterance "I want to have a broadband subscription, this eh ADSL I've read about". After the first two phases, the algorithm has found *family=broadband, intention=order,* and two possible values for the *object* slot, namely *object=subscription* and *object=ADSL.* Since the latter has higher information score, the final result is:

(*broadband, order, ADSL*)

The rule-based classifier was developed in about five man-weeks, and contains some 3,000 hand-written rules. When evaluated on a set of 2,300 utterances, it classified 67% of the utterances correctly. Thus, not surprisingly, its performance is significantly below the statistical classifier used in the deployed system. Still, the rule-based approach might be a viable alternative in less complex domains. It might also be usable for data collection purposes in early prototypes of natural-language call routing systems.

5 Evaluation of call-routing dialogues

5.1 Motivation

An important issue in the development of any dialogue system is the selection of an evaluation metric to quantify performance improvements. In the call-routing area, there have been many technical papers specifically comparing the performance of classifiers, using standard metrics such as accuracy of the semantic categories obtained over a test corpus (see e.g. Kuo and Lee, 2000, and Sarikaya et al., 2005). Accuracy is then stated as a percentage figure showing the degree of the categories that have been completely correctly classified, given that categories are atomic. There have also been some design-oriented papers that try to assess the effects of different prompt styles by looking at the proportion of routable versus unroutable calls given callers' first utterances. Thus, both of these strands of work base their evaluations on binary divisions between correct/incorrect and routable/unroutable, respectively. Furthermore, they both constitute utterance-based metrics in the sense that they focus on the outcome of a single system–caller turn.

An excellent example of a design-oriented call-routing paper is Williams and Witt (2004), which among other things compares open and directed prompt styles in the initial turn of the dialogue. Williams and Witt divide callers' responses into *Routable* (if the utterance contained sufficient information for the call to be routed) or *Failure* (if the utterance did not contain sufficient information for routing). Depending on why a call is not routable, Williams and Witt further subdivide instances of *Failure* into three cases: *Confusion* (utterances such as "Hello?" and "Is this a real person?"), *Agent* (the caller requests to speak to a human agent), and *Unroutable* (which corresponds to utterances that need disambiguation). Thus, Williams and Witt's performance metric uses altogether four labels. (In addition, they have three labels related to non-speech events: silence, DTMF and hang-up. Since such events are not handled by the classifier, they fall outside of the scope of this paper.)

Although all of Williams' and Witt's measures are needed in evaluating call-routing dialogue, the field clearly needs more in-depth evaluation. In particular, we need *more fine-grained metrics* in order to probe more exactly to what extent *Failure* actually means that the dialogue is off track. Furthermore, given that call-routing dialogues typically consist of between one and (say) five turns, we need not just utterance-based metrics, but also *dialogue-based metrics* — in other words, being able to evaluate the efficiency of an overall dialogue.

5.2 Utterance-based metrics

When assessing the performance of classification methods, it is perfectly reasonable to use the binary distinction correct/incorrect if only few categories are used. In such a context it can be assumed that different categories correspond to different departments of the organization, and that a misclassification would lead the call being routed the wrong way. However, with a richer category system, it is important to realize that the classifier can be partially correct. For instance, if the caller expresses that he wants technical support for his broadband connection, then the information that the purpose of the call has something to do with broadband is surely better than no information at all. If the system obtains this information, it could ask a directed follow-up question: *OK broadband. Please tell me if your call concerns an order, billing, deliveries, support, error report, or something else,* or something to that effect. Otherwise, the system can only restate the original question.

In the field of task-oriented dialogue, several evaluation metrics have been put forward that go beyond a simple division into correct/incorrect. In particular, *concept accuracy* (Boros et al. 1996) is an attempt to find a semantic analogue of word accuracy as used in speech recognition. Basically, the idea is to compute the degree of correctness of a semantic analysis based on a division of the representation into subunits, and by taking into account insertions, deletions and replacements of these subunits.

Making use of our multi-slot semantics, we can take subunits to correspond to semantic slot values. An insertion has occurred if the classifier spuriously has added information to some slot value (e.g. if the classifier outputs the value *broadband* for the *family* slot, when the correct value is *internet* or *unknown*). Conversely, a deletion has occurred when semantic triple output from the classifier contains a slot value which is situated lower than the correct value in the information ordering (a part of which is depicted in Figure 2). Finally, a replacement has occurred when the computed slot value and the correct slot value are unrelated in the information ordering.

By using concept accuracy as an evaluation metric for classifiers rather than the binary distinction correct/incorrect, we can arrive at more informative assessments. This possibility is brought about by the multi-slot structure of categories.

5.3 Dialogue-based metrics

In the literature, there have also been proposals for dialogue-based metrics. In particular, Glass et al. (2000) put forward two such metrics, *query density* (*QD*) and *concept efficiency* (*CE*). Query density is the mean number of new "concepts" introduced per user query, assuming that each concept corresponds to a slot–filler pair in the representation of the query. For example, a request such as "I'd like a flight from Stockholm to Madrid on Sunday afternoon" would introduce three new concepts, corresponding to departure, destination and time. Query density thus measures the rate at which the *user communicates content*. In contrast, concept efficiency measures the average number of turns it takes for a concept to be successfully understood by the system. Concept efficiency thus measures the rate at which the *system understands content*.

Using the multi-slot semantics, we can adapt the notions of query density and concept efficiency in order to arrive at a more fine-grained performance metric for call routing. The basic idea is to regard every element in the semantic triple as one "concept". We can then obtain a measure of how information increases in the dialogue by computing the difference between triples in each user utterance, where "difference" means that the values of two corresponding elements are not equal.

An example of computing query density is given below. We assume that the value of the semantic triple is initially (*unknown*, *unknown*, *unknown*).

System: Welcome to TeliaSonera. How may I help you?
Caller: Fixed telephony.
(fixedTelephony, unknown, unknown)
`1 new concept`
System: Could you tell me some more about what you want to do?
Caller: I can't use my broadband while I'm speaking on the phone.(*broadband, reportProblem, lineOrPhone*)
`3 new concepts`

Note that query density and concept efficiency are both applicable on a per-utterance basis as well as on the whole dialogue (or indeed arbitrary stretches of the dialogue). To compute these measures for the whole dialogue, we simply compute the mean number of new concepts introduced per user utterance and the average number of turns it takes for a concept to be successfully understood, respectively.

The principal application of this methodology is to measure the effectiveness of system utterances. When using a fine-grained system of categories, it is important that callers express themselves at a suitable level of detail. Too verbose user utterances are usually difficult to analyse, but too telegraphic user utterances are not good either, as they most often do not contain enough information to route the call directly. Therefore it is very important to design system utterances so as to make users give suitably expressive descriptions of their reasons for calling.

By using the query density metric it is possible to asses the effectiveness (in the above sense) of different alternative system utterances at various points in the dialogue, most notably the first system utterance. Again, this possibility is brought about by the multi-slot structure of categories. It is also possible to evaluate more general dialogue strategies over longer stretches of dialogue (e.g. the use of general follow-up questions like "*Could you please tell me some more about what you want to do*" as opposed to more directed questions like "*Please tell me if your call concerns an order, billing, deliveries, support, error report, or something else*"). By calculating the average query density over a number of consecutive utterances, it is possible to compare the relative merits of different such dialogue strategies.

We have not yet adopted this metric for evaluation of dialogues from the live system. However, elsewhere we have applied it to dialogues from the initial Wizard-of-Oz data collection for the TeliaSonera call routing system (Wirén et al. 2007). Here, we used it to compare two styles of disambiguation prompts, one completely open and one more directed.

6 Concluding remarks

In the literature, the natural-language call routing problem is often presented as the problem of classifying spoken utterances according to a set of atomic categories. The hypothesis underlying this paper is that this view is inadequate, and that there is a need for a more structured semantics. We base our claims on experiences gathered from the development and deployment of the TeliaSonera call center, for which we developed a multi-slot system of categories.

A multi-slot semantics offers several advantages. First of all, it makes the set of categories manageable for human taggers, and provides a means to break down the tagging task into subtasks. Furthermore, we have shown how multi-slot semantics for call-routing systems allows straightforward division of categories into routing categories and disambiguation categories, the possibility of multi-slot categorization, and the use of more fine-grained evaluation metrics like concept accuracy and query density.

Acknowledgements

This work has benefited greatly from discussions on category systems and classification with Marco Petroni, Linda Broström, Per-Olof Gällstedt, Alf Bergstrand and Erik Demmelmaier, and we thank them all. We would also like to thank Robert Sandberg and Erik Näslund for their support of this work.

References

Boros, M., Eckert, W., Gallwitz, F., Görz, G., Hanrieder, G. and Niemann, H. (1996). Towards understanding spontaneous speech: Word accuracy vs. concept accuracy. *Proc. Fourth International Conference on Spoken Language Processing* (*ICSLP*), pp. 1009–1012.

Chu-Carroll, J. and Carpenter, B. (1999) Vector-based natural language call routing. *Computational linguistics,* 25(3), pp. 361-388.

Cox, S. and Shahshahani, B. (2001). A comparison of some different techniques for vector based call-routing. *Proc. Eurospeech*, Aalborg, Denmark.

Glass, J., Polifroni, J., Seneff, S. and Zue, V. Data collection and performance evaluation of spoken dialogue systems: The MIT experience. In *Proc. Sixth International Conference on Spoken Language Processing* (*ICSLP*), Beijing, China.

Gorin, A., Riccardi, G., and Wright, J. (1997) How may I help you?. *Journal of Speech Communication,* 23, pp. 113-127.

Kuo, H-K J. and Lee, C-H. (2000) Discriminative training in natural language call routing. *Proc. Sixth International Conference on Spoken Language Processing* (*ICSLP*), Beijing, China.

Sarikaya, R, Kuo, H-K J., Goel, V. and Gao, Y. (2005) Exploiting unlabeled data using multiple classifiers for improved natural language call-routing. *Proc. Interspeech*, Lisbon, Portugal.

Speech Technology Magazine (2004) Q&A with Bell Canada's Belinda Banks, senior associate director, customer care. *Speech Technology Magazine,* vol 9, no 3.

Williams, Jason D. and Witt, Silke M. (2004). A comparison of dialog strategies for call routing. *International Journal of Speech Technology* 7(1), pp. 9–24.

Wirén, M., Eklund, R., Engberg, F. and Westermark, J. (2007). Experiences of an in-service Wizard-of-Oz data collection for the deployment of a call-routing application. *Proc. Bridging the gap: Academic and industrial research in dialog technology. NAACL workshop*, Rochester, New York, USA.

Enhancing commercial grammar-based applications using robust approaches to speech understanding

Matthieu Hébert
Network ASR R+D, Nuance Communications
1500, Université, Suite 935, Montréal, Québec, H3A 3T2, Canada
hebert@nuance.com

Abstract

This paper presents a series of measurements of the accuracy of speech understanding when grammar-based or robust approaches are used. The robust approaches considered here are based on statistical language models (SLMs) with the interpretation being carried out by phrase-spotting or robust parsing methods. We propose a simple process to leverage existing grammars **and** logged utterances to upgrade grammar-based applications to become more robust to out-of-coverage inputs. All experiments herein are run on data collected from deployed directed dialog applications and show that SLM-based techniques outperform grammar-based ones without requiring any change in the application logic.

1 Introduction

The bulk of the literature on spoken dialog systems is based on the simple architecture in which the input speech is processed by a statistical language model-based recognizer (SLM-based recognizer) to produce a word string. This word string is further processed by a robust parser (Ward, 1990) or call router (Gorin et al, 1997) to be converted in a semantic interpretation. However, it is striking to see that a large portion of deployed commercial applications do not follow this architecture and approach the recognition/interpretation problem by relying on hand-crafted rules (context-free grammars - CFGs). The apparent reasons for this are the up-front cost and additional delays of collecting domain-specific utterances to properly train the SLM (not to mention semantic tagging needed to train the call router) (Hemphill et al, 1990; Knight et al, 2001; Gorin et al, 1997). Choosing to use a grammar-based approach also makes the application predictable and relatively easy to design. On the other hand, these applications are usually very rigid: the users are allowed only a finite set of ways to input their requests and, by way of consequences, these applications suffer from high out-of-grammar (OOG) rates or out-of-coverage rates.

A few studies have been published comparing grammar-based and SLM-based approaches to speech understanding. In (Knight et al, 2001), a comparison of grammar-based and robust approaches is presented for a user-initiative home automation application. The authors concluded that it was relatively easy to use the corpus collected during the course of the application development to train a SLM which would perform better on out-of-coverage utterances, while degrading the accuracy on in-coverage utterances. They also reported that the SLM-based system showed slightly lower word error rate but higher semantic error rate for the users who know the application's coverage. In (Rayner et al, 2005), a rigorous test protocol is presented to compare grammar-based and robust approaches in the context of a medical translation system. The paper highlights the difficulties to construct a clean experimental set-up. Efforts are spent to control the *training set* of both approaches to have them align. The *training sets* are defined as the set of data available to build each system: for a grammar-based system, it might be a series of sample dialogs. (ten Bosch, 2005) presents experiments comparing grammar-based and SLM-based systems for naïve users and an expert user. They conclude that the SLM-based system is most effective in reducing the error rate for naïve users. Recently (see (Balakrishna et al, 2006)), a process was presented to automatically build SLMs from a wide variety of sources (in-service data, thesaurus, WordNet and world-wide web). Results on data from commercial speech applications presented therein echo earlier results (Knight et al, 2001) while reducing the effort to build interpretation rules.

Most of the above studies are not based on data collected on deployed applications. One of the conclusions from previous work, based on the measured fact that in-coverage accuracy of the grammar-based systems was far better than the SLM one, was that as people get more experience with the applications, they will naturally learn its coverage and gravitate towards it. While this can be an acceptable option for some types of applications (when the user population tends to be experienced or captive), it certainly is not a possibility for large-scale commercial applications that are targeted at the general public. A few examples of such applications are public transit schedules and fares information, self-help applications for utilities, banks, telecommunications business, and etc. Steering application design and research based on in-coverage accuracy is not suitable for these types of applications because a large fraction of the users are naïves and tend to use more natural and unconstrained speech inputs.

This paper exploits techniques known since the 90's (SLM with robust parsing, (Ward, 1990)) and applies them to build robust speech understanding into existing large scale directed dialog grammar-based applications. This practical application of (Ward, 1990; Knight et al, 2001; Rayner et al, 2005; ten Bosch, 2005) is cast as an upgrade problem which must obey the following constraints.

1. No change in the application logic and to the voice user interface (VUI)

2. Roughly similar CPU consumption

3. Leverage existing grammars

4. Leverage existing transcribed utterances

5. Simple process that requires little manual intervention

The first constraint dictates that, for each context, the interpretation engines (from the current and upgraded systems) must return the same semantics (i.e. same set of slots).

The rest of this paper is organized as follows. The next Section describes the applications from which the data was collected, the experimental set-up and the accuracy measures used. Section 3 describes how the semantic truth is generated. The main results of the upgrade from grammar-based to SLM-based recognition are presented in Section 4. The target audience for this paper is composed of application developers and researchers that are interested in the robust information extraction from directed dialog speech applications targeted at the general public.

2 Applications, corpus and experimental set-up

2.1 Application descriptions

As mentioned earlier, the data for this study was collected on deployed commercial directed dialog applications. AppA is a self-help application in the internet service provider domain, while AppB is also a self-help application in the public transportation domain. Both applications are grammar-based directed dialogs and receive a daily average of 50k calls. We will concentrate on a subset of contexts (dialog states) for each application as described in Table 1. The *mainmenu* grammars (each application has its own *mainmenu* grammar) contain high-level targets for the rest of the application and are active once the initial prompt has been played. The *command* grammar contains universal commands like "help", "agent", etc. The *origin* and *destination* grammars contain a list of $\sim$ 2500 cities and states with the proper prefixes to discriminate origin and destination. *num_type_passenger* accepts up to nine passengers of types adults, children, seniors, etc. Finally *time* is self explanatory. For each application, the prompt directs the user to provide a specific

Context	Description	Active grammars	Training sentences	Testing utts
AppA_MainMenu	Main menu for the application	mainmenu and commands	5000 (350)	5431 (642)
AppB_MainMenu	Main menu for the application	mainmenu and commands	5000 (19)	4039 (987)
AppB_Origin	Origin of travel	origin, destination and commands	5000 (20486)	8818 (529)
AppB_Passenger	Number and type of passenger	num_type_passenger and commands	1500 (32332)	2312 (66)
AppB_Time	Time of departure	time and commands	1000 (4102)	1149 (55)

Table 1: Description of studied contexts for each application. Note that the AppB_Origin context contains a *destination* grammar: this is due to the fact that the same set of grammars was used in the AppB_Destination context (not studied here). "Training" contains the number of training sentences drawn from the corpus and used to train the SLMs. As mentioned in Sec. 2.3, in the case of word SLMs, we also use sentences that are covered by the grammars in each context as backoffs (see Sec. 2). The number of unique sentences covered by the grammars is in parenthesis in the "Training" column. The "Testing" column contains the number of utterances in the test set. The number of those utterances that contain no speech (noise) is in parenthesis.

piece of information (directed dialog). Each grammar fills a single slot with that information. The information contained in the utterance "two adults and one child" (AppB_Passenger context) would be collapsed to fill the **num_type_passenger** slot with the value "Adult2_Child1". From the application point of view, each context can fill only a very limited set of slots. To keep results as synthesized as possible, unless otherwise stated, the results from all studied contexts will be presented per application: as such results from all contexts in AppB will be pooled together.

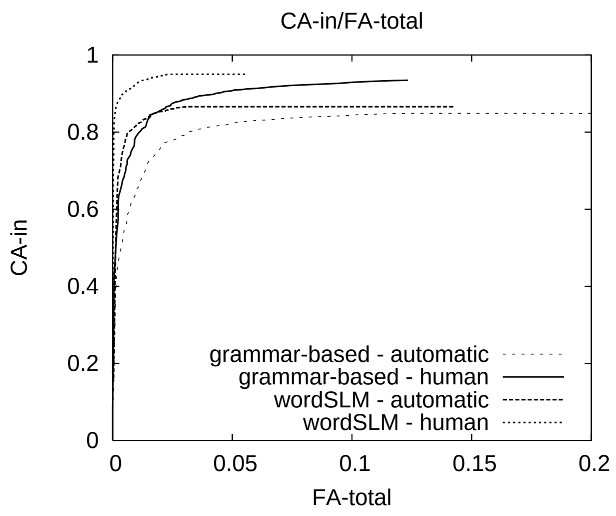
2.2 Corpus description

Table 1 presents the details of the corpus that we have used for this study. As mentioned above the entire corpora used for this study is drawn from commercially deployed systems that are used by the general public. The user population reflects realistic usage (expert vs naïve), noise conditions, handsets, etc. The training utterances do not contain noise utterances and is used primarily for SLM training (no acoustic adaptation of the recognition models is performed).

2.3 Experimental set-up description

The baseline system is the grammar-based system; the recognizer uses, on a per-context basis, the grammars listed in Table 1 in parallel. The SLM systems studied all used the same interpretation engine: robust parsing with the grammars listed in Table 1 as rules to fill slots. Note that this allows the application logic to stay unchanged since the set of potential slots returned within any given context is the same as for the grammar-based systems (see first constraint in Sec. 1). Adhering to this experimental set-up also guarantees that improvements measured in the lab will have a direct impact on the raw accuracy of the deployed application.

We have considered two different SLM-based systems in this study: standard SLM (wordSLM) and class-based SLM (classSLM) (Jelinek, 1990; Gillett and Ward, 1998). In the classSLM systems, the classes are defined as the rules of the interpretation engine (i.e. the grammars active for each context as defined in Table 1). The SLMs are all trained on a per-context basis (Xu and Rudnicky, 2000; Goel and Gopinath, 2006) as bi-grams with Witten-Bell discounting. To insure that the wordSLM system covered all sentences that the grammar-based system does, we augmented the training set of



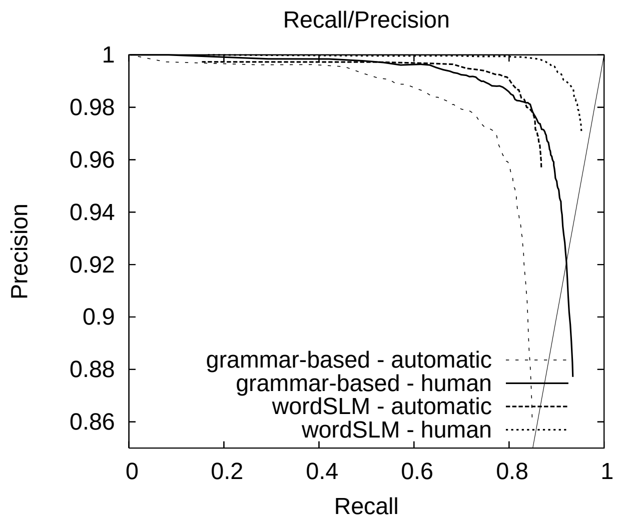


Figure 1: ROC curves for AppA_MainMenu with the automatic or human-generated truth. In each the grammar-based and SLM-based systems are compared.

the wordSLM (see Table 1) with the list of sentences that are covered by the baseline grammar-based system. This acts as a backoff in case a word or bi-gram is not found in the training set (not to be confused with bi-gram to uni-gram backoffs found in standard SLM training). This is particularly helpful when a little amount of data is available for training the wordSLM (see Sec. 4.3).

2.4 Accuracy measures

Throughout this paper, we will use two sets of measures. This is motivated by the fact that application developers are familiar with the concepts of correct/false acceptance at the utterance level. For information extraction (slot filling) from utterances, these concepts are restrictive because an utterance can be partly correct or wrong. In this case we prefer a more relevant measure from the information retrieval field: precision and recall on a per-slot basis. We use the following definitions.

- CA-in = #utts that had ALL slots correct (slot name and value) / #utts that are in-coverage (i.e. truth has at least a slot filled)
- FA-total = #utts that had at least one erroneous slot (slot name or value) / total #utts
- Precision = #slot correct slots (slot name and value) / #slots returned by system
- Recall = #slot correct slots (slot name and value) / #slots potential slots (in truth)

Since applications use confidence extensively to guide the course of dialogue, it is of limited interest to study forced-choice accuracy (accuracy with no rejection). Hence, we will present receiver operating characteristic (ROC) curves. The slot confidence measure is based on redundancy of a slot/value pair across the NBest list. For CA-in and FA-total, the confidence is the average confidence of all slots present in the utterance. Note that in the case where each utterance only fills a single slot, CA-in = Recall.

3 Truth

Due to the large amount of data processed (see Table 1), semantic tagging by a human may not be available for all contexts (orthographic transcriptions are available however). We need to resort to a more automatic way of generating the truth files while maintaining a strong confidence in our measurements. To this end, we need to ensure that any automatic way of generating the truth will not bias the results towards any of the systems.

The automatic truth can be generated by simply using the robust parser (see Sec. 2.3) on the orthographic transcriptions which are fairly cheap to acquire. This will generate a semantic interpretation for those utterances that contain fragments that parse rules defined by the interpretation engine. The human-generated truth is the result of semantically tagging all utterances that didn't yield a full parse by one of the rules for the relevant context.

Figure 1 presents the ROC curves of human and automatic truth generation for the grammar-based and wordSLM systems. We can see that human semantic tagging increases the accuracy substantially, but this increase doesn't seem to favor one system over the other. We are thus led to believe that in our case (very few well defined non-overlapping classes) the automatic truth generation is sufficient. This would not be the case, for example if for a given context a *time* grammar and *number* were active classes. Then, an utterance like "seven" might lead to an erroneous slot being automatically filled while a human tagger (who would have access to the entire dialog) would tag it correctly.

In our experiments, we will use the human semantically tagged truth when available (AppA_MainMenu and AppB_Origin). We have checked that the conclusions of this paper are not altered in any way if the automatic semantically tagged truth had been used for these two contexts.

4 Results and analysis

4.1 Out-of-coverage analysis

Context (#utts)	grammar-based	SLM-based
AppA_MainMenu	1252	1086
AppB_MainMenu	1287	1169
AppB_Origin	1617	1161
AppB_Passenger	492	414
AppB_Time	327	309

Table 2: Number of utterances out-of-coverage for each context.

Coverage is a function of the interpretation engine. We can readily analyze the effect of going from a grammar-based interpretation engine (grammars in Table 1 are in parallel) to the robust approach (rules from grammars in Table 1 are used in robust parsing). This is simply done by running the interpretation engine on the orthographic transcriptions. As expected, the coverage increased. Table 2 shows the number of utterances that didn't fire any rule for each of the interpretation engines. These include noise utterances as described in Table 1. If we remove the noise utterances, going from the grammar-based interpretation to an SLM-based one reduces the out-of-coverage by 31%. This result is interesting because the data was collected from directed-dialog applications which should be heavily guiding the users to the grammar-based system's coverage.

4.2 Results with recognizer

The main results of this paper are found in Figure 2. It presents for grammar-based, wordSLM and classSLM systems the four measurements mentioned in Sec.2.4 for AppA and AppB. We have managed, with proper Viterbi beam settings, to keep in the increase in CPU (grammar-based system $\rightarrow$ SLM-based system) between 0% and 24% relative. We can see that the wordSLM is outperforming the classSLM. The SLM-based systems outperform the grammar-based systems substantially ($\sim 30-50\%$ error rate reduction on most of the confidence domain). The only exception to this is the classSLM in AppA: we will come back to this in Sec. 4.4. This can be interpreted as a different conclusion than those of (Knight et al, 2001; ten Bosch, 2005). The discrepancy can be tied to the fact that the data we are studying comes from a live deployment targeted to the general public. In this case, we can make the hypothesis that a large fraction of the population is composed of naïve users. As mentioned in (ten Bosch, 2005), SLM-based systems perform better than grammar-based ones on that cross-section of the user population.

One might argue that the comparison between the grammar-based and wordSLM systems is unfair because the wordSLM intrinsically records the *a priori* probability that a user says a specific phrase while the grammar-based system studied here didn't benefit from this information. In Sec. 4.4, we will address this and show that *a priori* has a negligible effect in this context.

Note that these impressive results are surprisingly easy to achieve. A simple process could be as follows. An application is developed using grammar-based paradigm. After a limited deployment or pilot with real users, a wordSLM is built from transcribed (orthographic) data from the field. Then the recog-

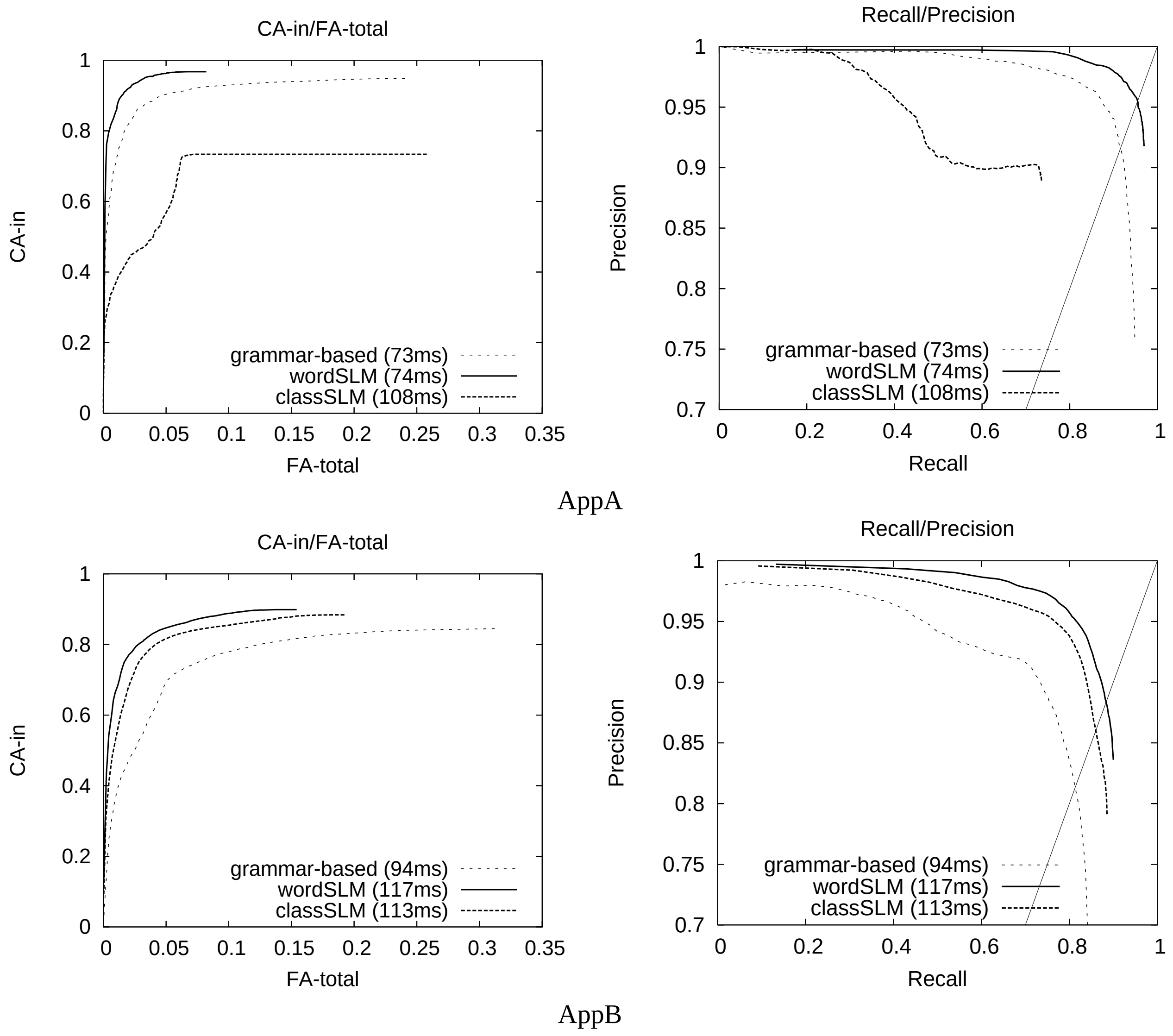
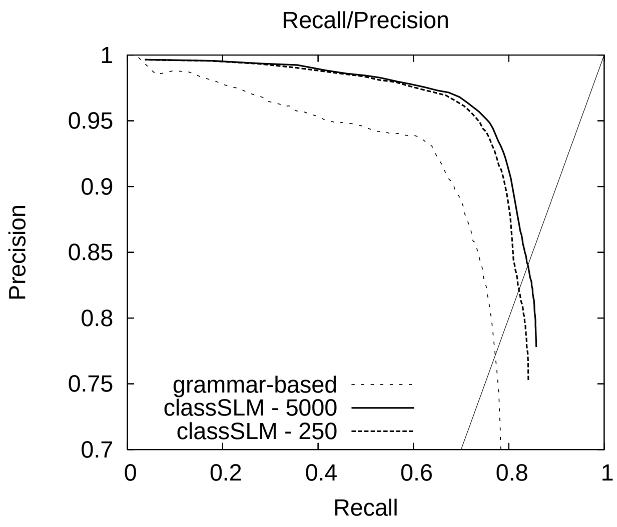


Figure 2: ROC curves for AppA (top) and AppB (bottom). In parenthesis is the average time for the recognition and interpretation.

nition and interpretation engines are upgraded. The grammars built in the early stages of development can largely be re-used as interpretation rules.

4.3 Amount of training data for SLM training

For the remaining Sections, we will use precision and recall for simplicity. We will discuss an extreme case where only a subset of 250 sentences from the standard training set is used to train the SLM. We have run experiments with two contexts: AppA_MainMenu and AppB_Origin. These contexts are useful because a) we have the human-generated truth and b) they represent extremes in the complexity of grammars (see Section 2). On one hand, the grammars for AppA_MainMenu can cover a total of 350 unique sentences while AppB_Origin can cover over 20k. As the amount of training data for the SLMs is reduced from 5000 down to 250 sentences, the accuracy for AppA_MainMenu is only perceptibly degraded for the wordSLM and classSLM systems on the entire confidence domain (not shown here). On the other hand, in the case of the more complex grammar (class), it is a different story which highlights a second regime. For AppB_Origin, the precision and recall curve is presented on Figure 3. In the case of classSLM (left),



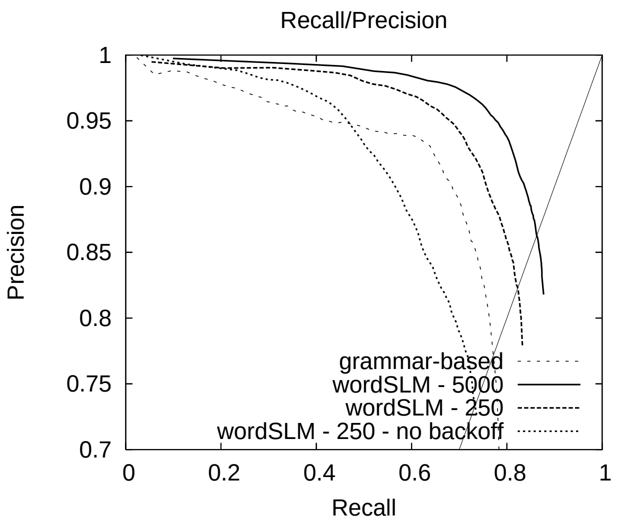


Figure 3: Precision and recall for the AppB_Origin context as the amount of training data for the SLMs is reduced. On the left, classSLM systems are presented; on the right it is the wordSLM.

even with very little training data, the accuracy is far better than the grammar-based system and only slightly degraded by reducing the size of the training set. In the case of wordSLM (right), we can still see that the accuracy is better than the grammar-based system (refer to "wordSLM - 250" on the graph), but the reduction of training data has a much more visible effect. If we remove the sentences that were drawn from the grammar-based system's coverage (backoff - see Sec. 2.3), we can see that the drop in accuracy is even more dramatic.

4.4 Coverage of interpretation rules and priors

As seen in Sec. 4.2, the classSLM results for AppA are disappointing. They, however, shed some light on two caveats of the robust approach described here. The first caveat is the coverage of the interpretation rules. As described in Sec. 2, the SLM-based systems' training sets and interpretation rules (grammars from Table 1) were built in isolation. This can have a dramatic effect: after error analysis of the classSLM system's results, we noticed a large fraction of errors for which the recognized string was a close (semantically identical) variant of a rule in the interpretation engine ("cancellations" vs "cancellation"). In response, we implemented a simple tool to increase the coverage of the grammars (and hence the coverage of the interpretation rules) using the list of words seen in the training set. The criteria for selection is based on common stem with a word in the grammar.

The second caveat is based on fact that the classSLM suffers from a lack of prior information once the decoding process enters a specific class since the grammars (class) do not contain priors. The wordSLM benefits from the full prior information all along the search. We have solved this by training a small wordSLM **within** each grammar (class): for each grammar, the training set for the small wordSLM is composed of the set of fragments from all utterances in the main training set that fire that specific rule. Note that this represents a way to have the grammar-based and SLM-based systems share a common *training set* (Rayner et al, 2005).

In Figure 4, we show the effect of increasing the coverage and adding priors in the grammars. The first conclusion comes in comparing the grammar-based results with and without increased coverage (enhanced+priors in figure) and priors. We see that the ROC curves are one on top of the other. The only differences are: a) at low confidence where the enhanced+priors version shows better precision, and b) the CPU consumption is greatly reduced (73ms $\rightarrow$ 52ms). When the enhanced+priors version of the grammars (for classes and interpretation rules) is used in the context of the classSLM system, we can see that there is a huge improvement in the accuracy: this shows the importance of keeping the SLM

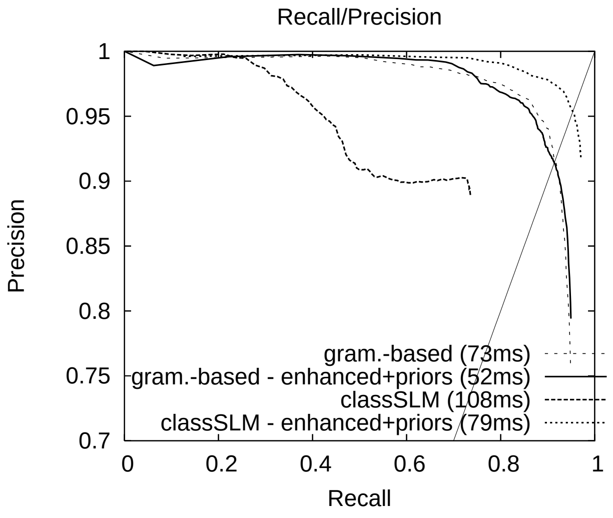


Figure 4: ROC curves for AppA showing the effect of increasing the grammar coverage and adding prior information in the grammars.

and interpretation rules in-sync. The final classSLM ROC curve (Figure 4) is now comparable with its wordSLM counter-part (Figure 2 upper right graph).

5 Conclusion

We have demonstrated in this paper that grammar-based systems for commercially deployed directed dialog applications targeted at the general public can be improved substantially by using SLMs with robust parsing. This conclusion is different than (Rayner et al, 2005) and can be attributed to that fact that the general public is likely composed of a large portion of naïve users. We have sketched a very simple process to upgrade an application from using a grammar-based approach to a robust approach when in-service data and interpretation rules (grammars) are available. We have also shown that only a very small amount of data is necessary to train the SLMs (Knight et al, 2001). Class-based SLMs should be favored in the case where the amount of training data is low while word-based SLMs should be used when enough training data is available. In the case of non-overlapping classes, we have demonstrated the soundness of automatically generated semantic truth.

6 Acknowledgements

The author would like to acknowledge the helpful discussions with M. Fanty, R. Tremblay, R. Lacouture and K. Govindarajan during this project.

References

W. Ward. 1990. The CMU Air Travel Information Service: Understanding spontaneous speech . *Proc. of the Speech and Natural Language Workshop*, Hidden Valley PA, pp. 127–129.

A.L. Gorin, B.A. Parker, R.M. Sachs and J.G. Wilpon. 1997. How may I help you?. *Speech Communications*, 23(1):113–127.

C. Hemphill, J. Godfrey and G. Doddington. 1990. The ATIS spoken language systems and pilot corpus. *Proc. of the Speech and Natural Language Workshop*, Hidden Valley PA, pp. 96–101.

S. Knight, G. Gorrell, M. Rayner, D. Milward, R. Koeling and I. Lewin. 2001. Comparing grammar-based and robust approaches to speech understanding: a case study. *Proc. of EuroSpeech*.

M. Rayner, P. Bouillon, N. Chatzichrisafis, B.A. Hockey, M. Santaholma, M. Starlander, H. Isahara, K. Kanzaki and Y. Nakao. 2005. A methodology for comparing grammar-based and robust approaches to speech understanding. *Proc. of EuroSpeech*.

L. ten Bosch. 2005. Improving out-of-coverage language modelling in a multimodal dialogue system using small training sets. *Proc. of EuroSpeech*.

M. Balakrishna, C. Cerovic, D. Moldovan and E. Cave. 2006. Automatic generation of statistical language models for interactive voice response applications. *Proc. of ICSLP*.

J. Gillett and W. Ward. 1998. A language model combining tri-grams and stochastic context-free grammars. *Proc. of ICSLP*.

F. Jelinek. 1990. Readings in speech recognition, Edited by A. Waibel and K.-F. Lee , pp. 450-506. Morgan Kaufmann, Los Altos.

W. Xu and A. Rudnicky. 2000. Language modeling for dialog system. *Proc. of ICSLP*.

V. Goel and R. Gopinath. 2006. On designing context sensitive language models for spoken dialog systems. *Proc. of ICSLP*.

WIRE: A Wearable Spoken Language Understanding System for the Military

Helen Hastie **Patrick Craven** **Michael Orr**

Lockheed Martin Advanced Technology Laboratories
3 Executive Campus
Cherry Hill, NJ 08002
{hhastie, pcraven, morr}@atl.lmco.com

Abstract

In this paper, we present the WIRE system for human intelligence reporting and discuss challenges of deploying spoken language understanding systems for the military, particularly for dismounted warfighters. Using the PARADISE evaluation paradigm, we show that performance models derived using standard metrics can account for 68% of the variance of User Satisfaction. We discuss the implication of these results and how the evaluation paradigm may be modified for the military domain.

1 Introduction

Operation Iraqi Freedom has demonstrated the need for improved communication, intelligence, and information capturing by groups of dismounted warfighters (soldiers and Marines) at the company level and below. Current methods of collecting intelligence are cumbersome, inefficient and can endanger the safety of the collector. For example, a dismounted warfighter who is collecting intelligence may stop to take down notes, including his location and time of report or alternatively try to retain the information in memory. This information then has to be typed into a report on return to base. The authors have developed a unique, hands-free solution by capturing intelligence through spoken language understanding technology called WIRE or Wearable Intelligent Reporting Environment. Through WIRE, users simply speak what they see, WIRE understands the speech and automatically populates a report. The report format we have adopted is a SALUTE report which stands for the information fields: Size, Activity, Location, Unit, Time and Equipment. The military user is used to giving information in a structure way, therefore, information entry is structured but the vocabulary is reasonably varied, an example report is "Size is three insurgents, Activity is transporting weapons." These reports are tagged by WIRE with GPS position and time of filing. The report can be sent in real-time over 802.11 or radio link or downloaded on return to base and viewed on a C2 Interface. WIRE will allow for increased amounts of digitized intelligence that can be correlated in space and time to predict adverse events. In addition, pre and post-patrol briefings will be more efficient, accurate and complete. Additionally, if reports are transmitted in real time, they have the potential to improve situational awareness in the field.

This paper discusses the challenges of taking spoken language understanding technology out of the laboratory and into the hands of dismounted warfighters. We also discuss usability tests and results from an initial test with Army Reservists.

2 System Overview

WIRE is a spoken language understanding system that has a plug-and-play architecture (Figure 1) that allows for easy technology refresh of the different components. These components pass events to each other via an event bus. The speech is collected by an audio server and passed to the Automatic Speech Recognizer (ASR) server, which is responsible for converting the audio waveform into an N-best list. The Natural Language (NL) under

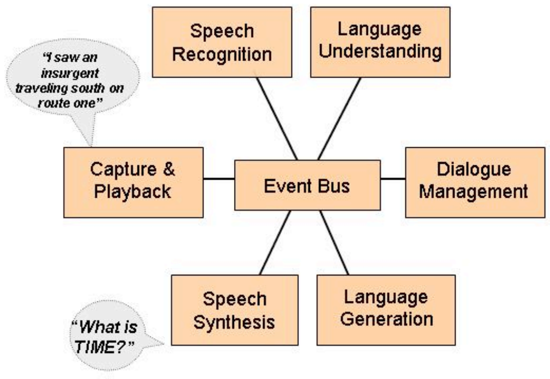


Figure 1. WIRE System Architecture

standing component executes a named-entity tagger to tag and retain key text elements within the each candidate N-best list element. The sets of tagged entities are then parsed using a bottom-up chart parser. The chart parser validates each named entity tag sequence and generates a syntactic parse tree. A heuristic is then applied to select the best parse tree from the N-best list as the representative spoken text. After a parse tree is selected, a semantic parser is used to prune the parse tree and produce a semantic frame—a data structure that represents the user's spoken text. The semantic frame is then passed through a rule-based filter that translates text as necessary for processing, e.g., converting text numbers to digits.

The semantic frame is then passed to the Dialogue Manager which decides what action to take based on the most recent utterance and its context. If the system is to speak a reply, the natural language generation component generates a string of text that is spoken by the Text-To-Speech engine (TTS).

The WIRE spoken language understanding system was fully developed by the authors with the exception of the ASR, called Dynaspeak™, which was developed by SRI International (Franco et al., 2002) and the TTS engine from Loquendo S.p.A. Grammars for the ASR and NL have to be written for each new domain and report type.

In order for the system to adapt to the user's environment, there are two modes of operation. *Interactive* mode explicitly confirms what the user says and allows the user to ask the system to read back certain fields or the whole report. Alternatively, in *stealth* mode, the user simply speaks the report and WIRE files it immediately. In both cases, audio is recorded as a back-up for report accuracy.

3 Challenges of Deployment to Dismounted Warfighters

The goal of WIRE is to provide a means of reporting using an interface that is conceptually easy to use through natural language. This is particularly challenging given the fluid nature of war and the constant emergence of new concepts such as different types of Improvised Explosive Devices (IEDs) or groups of insurgents. Another challenge is that each unit has its own idiosyncrasies, call signs and manner of speaking. Because WIRE is a limited-domain system and it is not possible to incorporate all of this variability, we found training to be a key factor in user and system performance and acceptance.

A new challenge that phone-based or desk-top systems have yet to face is the need for a *mobile* spoken language understanding system that can be worn by the user. From a software perspective, WIRE has to have a small footprint. From a hardware perspective, the system has to be lightweight, robust, and rugged and must integrate with existing equipment. Wearable computing is constantly evolving and eventually WIRE will be able to run on a system as small as a button. We have also been working with various companies to create a USB noise-canceling microphone similar to what the military user is accustomed to.

4 Experiment Design

Fifteen Army Reservists and three former Marines participated in WIRE usability tests in a laboratory environment. The Reservists predominately provide drill-instructor support for Army basic training groups. The session began with a brief introduction to the WIRE system. Following that, participants reviewed a series of self-paced training slides. They then completed two sets of four scenarios, with one set completed in stealth mode and the other in interactive mode. A total of 523 utterances were collected. Participants were asked to complete five-question surveys at the end of each set of scenarios. For the regression model described below, we averaged User Satisfaction scores for both types of interaction modes.

We adopted the PARADISE evaluation method (Walker et al., 1997). PARADISE is a “decision-theoretic framework to specify the relative contribution of various factors to a system’s overall performance.” Figure 2 shows the PARADISE model which defines system performance as a weighted function of task-based success measures and dialogue-based cost measures. Dialogue costs are further divided into dialogue efficiency measures and qualitative measures. Weights are calculated by correlating User Satisfaction with performance.

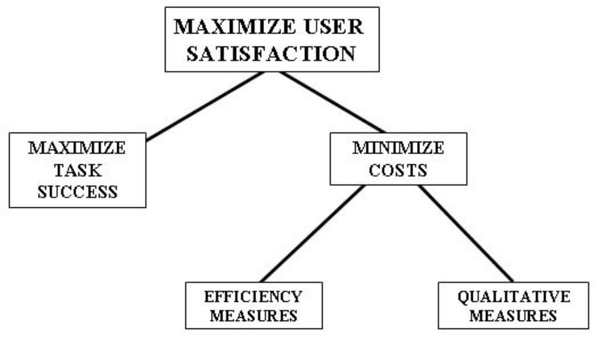


Figure 2. PARADISE Model (Walker et al., 1997)

The set of metrics that were collected are:

- **Dialogue Efficiency Measures:** User Turns, Average Length of Utterance, Average Response Latency and Platform.
- **Dialogue Quality Measures:** Word Accuracy.
- **Task Success Measures:** Report Accuracy, Field Correctness for Size, Activity, Location, Unit, Time and Equipment.
- **User Satisfaction:** Average of User Expertise, User Confidence, System Trust, Task Ease, Future Use.

User Satisfaction is the average of responses from a survey of five questions on a five-point Likert scale with five being the highest rating. These questions include:

- **Q1**: I knew what I could say at any point (User Expertise).
- **Q2**: I knew what I was doing at any point in the dialog (User Confidence).
- **Q3**: I trusted that WIRE accurately captured my report information (System Trust).
- **Q4**: I felt like I could create and file a report quickly (Task Ease).
- **Q5**: I would recommend that this system be fielded (Future Use).

These questions are modified from the more traditional User Satisfaction questions (Walker et al., 2001) that include *TTS Performance* and *Expected Behavior*. TTS Performance was substituted because the voice is of such a high quality that it sounds just like a human; therefore, the question is no longer relevant. Expected Behavior was substituted for this study because WIRE is mostly user initiative for the reporting domain.

The Task Success metric was captured by Report Accuracy. This was calculated by averaging the correctness of each field over the number of fields attempted. Field correctness was scored manually as either 1 or 0, depending on whether the report field was filled out completely correctly based on user’s intent. Partial credit was not given.

Various platforms were used in the experiment, including laptops, tablet PCs and wearable computers. The Platform metric reflects the processing power with 0 being the highest processing power and 1 the less powerful wearable computers.

5 Experimental Results

We applied the PARADISE model using the metrics described above by performing multiple linear regression using a backward coefficient selection method that iteratively removes coefficients that do not help prediction. The best model takes into account 68% of the variance of User Satisfaction (p=.01). Table 1 gives the metrics in the model with their coefficients and p values. Note that the data set is quite small (N=18, df=17), which most likely affected the results.

Table 1. Predictive Power and Significance of Metrics

Metric	**Standardized β Coefficients**	***p* value**
User Turns	-0.633	0.01
Unit Field Correctness	0.735	0.00
Platform	-0.24	0.141

Results show an average User Satisfaction of 3.9 that is broken down into 4.09 for interactive mode and 3.73 for stealth. The lowest medium user satisfaction score was for System Trust (3.5), the highest for Task Ease (4.5).

Speech recognition word accuracy is 79%, however, Report Accuracy, which is after the speech has been processed by the NL, is 84%. Individual field correctness scores varied from 93% for Activity to 75% for Location. From previous tests, we have found that word accuracy increases through user training and experience up to 95%.

6 Interpretation and Discussion

These initial results show that the User Turns metric is negatively predictive of User Satisfaction. This is intuitive as the more user turns it takes to complete a report the less satisfied the user. (Walker et al., 2001) have similar findings for the Communicator data where Task Duration is negatively predictive of User Satisfaction in their model (coefficient -0.15).

Secondly, Unit Field Correctness is predictive of User Satisfaction. Given this model and the limited data set, this metric may represent task completion better than overall Report Accuracy. During the test, the user can visually see the report before it is sent. If there are mistakes then this too will affect User Satisfaction. This is similar to findings by (Walker et al., 2001) who found that Task Completion was positively predictive of User Satisfaction (coefficient 0.45).

Finally, Platform is negatively predictive, in other words: the higher the processing power (scored 0) the higher the User Satisfaction and the lower the processing power (scored 1) the lower the User Satisfaction. Not surprisingly, users prefer the system when it runs on a faster computer. This means that the success of the system is likely dependent on an advanced wearable computer. There have been recent advances in this field since this experiment. These systems are now available with faster Intel processors and acceptable form factor and battery life.

The User Satisfaction results show that areas of improvement include increasing the trust in the user (Q3). This challenge has been discussed previously for military applications in (Miksch et al., 2004) and may reflect tentativeness of military personnel to accept new technology. Trust in the system can be improved by putting the system in "interactive" mode, which explicitly confirms each utterance and allows the user to have the system read back the report before sending it. A Wilcoxon signed-rank test ($Z = 2.12$, $p < .05$) indicated that scores for this question were significantly higher for interactive mode (M = 3.93) than stealth mode (M=3.27).

Our current evaluation model uses User Satisfaction as a response variable in line with previous PARADISE evaluations (Walker et al., 2001). However, User Satisfaction may not be the most appropriate metric for military applications. Unlike commercial applications, the goal of a military system is not to please the user but rather to complete a mission in a highly effective and safe manner. Therefore, a metric such as mission effectiveness may be more appropriate. Similarly, (Forbes-Riley and Litman, 2006) use the domain-specific response variable, of student learning in their evaluation model.

An obvious extension to this study is to test in more realistic environments where the users may be experiencing stress in noisy environments. Initial studies have been performed whereby users are physically exerted. These studies did not show a degradation in performance. In addition, initial tests outside in noisy and windy environments emphasize the need for a high quality noise canceling microphone. Further, more extensive tests of this type are needed.

In summary, we have presented the WIRE spoken language understanding system for intelligence reporting, and we have discussed initial evaluations using the PARADISE methods. Through advances in spoken language understanding, hardware and microphones, this technology will soon transition out of the laboratory and into the field to benefit warfighters and improve security in conflict regions.

Acknowledgments

Thanks to the Army Reservist 1/417th Regt, 1st BDE 98th Div (IT).

References

Forbes-Riley, K. and Litman, D.J. "Modeling User Satisfaction and Student Learning in a Spoken Dialogue Tutoring System with Generic, Tutoring, and User Affect Parameters." *HLT-NAACL*, 2006.

Franco, H., Zheng, J., Butzberger, J., Cesari, F., Frandsen, M., Arnold, J., Rao, R., Stolcke, A., and Abrash, V. "Dynaspeak™: SRI International's scalable speech recognizer for embedded and mobile systems." *HLT*, 2002.

Miksch, D., Daniels, J.J., and Hastie, H. (2004). "Establishing Trust in a Deployed Spoken Language System for Military Domains." *In Proc. of AAAI Workshop*, 2004.

Walker, M.A., Litman, D., Kamm, C. and Abella, A. "PARADISE: A Framework for Evaluating Spoken Dialogue Agents." *ACL*, 1997.

Walker, M.A., Passonneau, R., and Boland, J.E. "Quantitative and Qualitative Evaluation of DARPA Communicator Spoken Dialogue Systems." *ACL*, 2001.

Different measurements metrics to evaluate a chatbot system

Bayan Abu Shawar
IT department
Arab Open University
[add]
b_shawar@arabou-jo.edu.jo

Eric Atwell
School of Computing
University of Leeds
LS2 9JT, Leeds-UK
eric@comp .leeds.ac.uk

Abstract

A chatbot is a software system, which can interact or "chat" with a human user in natural language such as English. For the annual Loebner Prize contest, rival chatbots have been assessed in terms of ability to fool a judge in a restricted chat session. We are investigating methods to train and adapt a chatbot to a specific user's language use or application, via a user-supplied training corpus. We advocate open-ended trials by real users, such as an example Afrikaans chatbot for Afrikaans-speaking researchers and students in South Africa. This is evaluated in terms of "glass box" dialogue efficiency metrics, and "black box" dialogue quality metrics and user satisfaction feedback. The other examples presented in this paper are the Qur'an and the FAQchat prototypes. Our general conclusion is that evaluation should be adapted to the application and to user needs.

1 Introduction

"Before there were computers, we could distinguish persons from non-persons on the basis of an ability to participate in conversations. But now, we have hybrids operating between person and non persons with whom we can talk in ordinary language." (Colby 1999a). Human machine conversation as a technology integrates different areas where the core is the language, and the computational methodologies facilitate communication between users and computers using natural language.

A related term to machine conversation is the chatbot, a conversational agent that interacts with users turn by turn using natural language. Different chatbots or human-computer dialogue systems have been developed using text communication such as Eliza (Weizenbaum 1966), PARRY (Colby 1999b), CONVERSE (Batacharia etc 1999), ALICE[1]. Chatbots have been used in different domains such as: customer service, education, web site help, and for fun.

Different mechanisms are used to evaluate Spoken Dialogue Systems (SLDs), ranging from glass box evaluation that evaluates individual components, to black box evaluation that evaluates the system as a whole McTear (2002). For example, glass box evaluation was applied on the (Hirschman 1995) ARPA Spoken Language system, and it shows that the error rate for sentence understanding was much lower than that for sentence recognition. On the other hand black box evaluation evaluates the system as a whole based on user satisfaction and acceptance. The black box approach evaluates the performance of the system in terms of achieving its task, the cost of achieving the task in terms of time taken and number of turns, and measures the quality of the interaction, normally summarised by the term 'user satisfaction', which indicates whether the user " gets the information s/he wants, is s/he comfortable with the system, and gets the information within acceptable elapsed time, etc." (Maier et al 1996).

The Loebner prize[2] competition has been used to evaluate machine conversation chatbots. The Loebner Prize is a Turing test, which evaluates the ability of the machine to fool people that they are talking to human. In essence, judges are allowed a short chat (10 to 15 minutes) with each chatbot, and asked to rank them in terms of "naturalness".

ALICE (Abu Shawar and Atwell 2003) is the Artificial Linguistic Internet Computer Entity, first

[1] http://www.alicebot.org/
[2] http://www.loebner.net/Prizef/loebner-prize.html

Bridging the Gap: Academic and Industrial Research in Dialog Technologies Workshop Proceedings, pages 89–96,
NAACL-HLT, Rochester, NY, April 2007. ©2007 Association for Computational Linguistics

implemented by Wallace in 1995. ALICE knowledge about English conversation patterns is stored in AIML files. AIML, or Artificial Intelligence Mark-up Language, is a derivative of Extensible Mark-up Language (XML). It was developed by Wallace and the Alicebot free software community during 1995-2000 to enable people to input dialogue pattern knowledge into chatbots based on the A.L.I.C.E. open-source software technology.

In this paper we present other methods to evaluate the chatbot systems. ALICE chtabot system was used for this purpose, where a Java program has been developed to read from a corpus and convert the text to the AIML format. The Corpus of Spoken Afrikaans (Korpus Gesproke Afrikaans, KGA), the corpus of the holy book of Islam (Qur'an), and the FAQ of the School of Computing at University of Leeds[3] were used to produce two KGA prototype, the Qur'an prototype and the FAQchat one consequently.

Section 2 presents Loebner Prize contest, section 3 illustrates the ALICE/AIMLE architecture. The evaluation techniques of the KGA prototype, the Qur'an prototype, and the FAQchat prototype are discussed in sections 4, 5, and 6 consequently. The conclusion is presented in section 7.

2 The Loebner Prize Competition

The story began with the "imitation game" which was presented in Alan Turing's paper "Can Machine think?" (Turing 1950). The imitation game has a human observer who tries to guess the sex of two players, one of which is a man and the other is a woman, but while screened from being able to tell which is which by voice, or appearance. Turing suggested putting a machine in the place of one of the humans and essentially playing the same game. If the observer can not tell which is the machine and which is the human, this can be taken as strong evidence that the machine can think.

Turing's proposal provided the inspiration for the Loebner Prize competition, which was an attempt to implement the Turing test. The first contest organized by Dr. Robert Epstein was held on 1991, in Boston's Computer Museum. In this incarnation the test was known as the Loebner contest, as Dr. Hugh Loebner pledged a $100,000 grand prize for the first computer program to pass the test. At the beginning it was decided to limit the topic, in order to limit the amount of language the contestant programs must be able to cope with, and to limit the tenor. Ten agents were used, 6 were computer programs. Ten judges would converse with the agents for fifteen minutes and rank the terminals in order from the apparently least human to most human. The computer with the highest median rank wins that year's prize. Joseph Weintraub won the first, second and third Loebner Prize in 1991, 1992, and 1993 for his chatbots, PC Therapist, PC Professor, which discusses men versus women, and PC Politician, which discusses Liberals versus Conservatives. In 1994 Thomas Whalen (Whalen 2003) won the prize for his program TIPS, which provides information on a particular topic. TIPS provides ways to store, organize, and search the important parts of sentences collected and analysed during system tests.

However there are sceptics who doubt the effectiveness of the Turing Test and/or the Loebner Competition. Block, who thought that "the Turing test is a sorely inadequate test of intelligence because it relies solely on the ability to fool people"; and Shieber (1994), who argued that intelligence is not determinable simply by surface behavior. Shieber claimed the reason that Turing chose natural language as the behavioral definition of human intelligence is "exactly its open-ended, freewheeling nature", which was lost when the topic was restricted during the Loebner Prize. Epstein (1992) admitted that they have trouble with the topic restriction, and they agreed "every fifth year or so … we would hold an open-ended test - one with no topic restriction." They decided that the winner of a restricted test would receive a small cash prize while the one who wins the unrestricted test would receive the full $100,000.

Loebner in his responses to these arguments believed that unrestricted test is simpler, less expensive and the best way to conduct the Turing Test. Loebner presented three goals when constructing the Loebner Prize (Loebner 1994):

- "No one was doing anything about the Turing Test, not AI." The initial Loebner Prize contest was the first time that the Turing Test had ever been formally tried.
- Increasing the public understanding of AI is a laudable goal of Loebner Prize. "I believe that this contest will advance AI and

[3] http://www.comp.leeds.ac.uk

serve as a tool to measure the state of the art.”
- Performing a social experiment.

The first open-ended implementation of the Turing Test was applied in the 1995 contest, and the prize was granted to Weintraub for the fourth time. For more details to see other winners over years are found in the Loebner Webpage[4].

In this paper, we advocate alternative evaluation methods, more appropriate to practical information systems applications. We have investigated methods to train and adapt ALICE to a specific user’s language use or application, via a user-supplied training corpus. Our evaluation takes account of open-ended trials by real users, rather than controlled 10-minute trials.

3 The ALICE/AIML chatbot architecture

AIML consists of data objects called AIML objects, which are made up of units called topics and categories. The topic is an optional top-level element; it has a name attribute and a set of categories related to that topic. Categories are the basic units of knowledge in AIML. Each category is a rule for matching an input and converting to an output, and consists of a pattern, which matches against the user input, and a template, which is used in generating the Alice chatbot answer. The format structure of AIML is shown in figure 1.

```
< aiml version=”1.0” >
< topic name=” the topic” >

<category>
<pattern>PATTERN</pattern>
<that>THAT</that>
<template>Template</template>
</category>
    ..
    ..
</topic>
</aiml>
```

The <that> tag is optional and means that the current pattern depends on a previous bot output.

Figure 1. AIML format

[4] http://www.loebner.net/Prizef/loebner-prize.html

The AIML pattern is simple, consisting only of words, spaces, and the wildcard symbols _ and *. The words may consist of letters and numerals, but no other characters. Words are separated by a single space, and the wildcard characters function like words. The pattern language is case invariant. The idea of the pattern matching technique is based on finding the best, longest, pattern match. Three types of AIML categories are used: *atomic category*, are those with patterns that do not have wildcard symbols, _ and *; *default categories* are those with patterns having wildcard symbols * or _. The wildcard symbols match any input but can differ in their alphabetical order. For example, given input ‘hello robot’, if ALICE does not find a category with exact matching atomic pattern, then it will try to find a category with a default pattern; The third type, *recursive categories* are those with templates having <srai> and <sr> tags, which refer to simply recursive artificial intelligence and symbolic reduction. Recursive categories have many applications: symbolic reduction that reduces complex grammatical forms to simpler ones; divide and conquer that splits an input into two or more subparts, and combines the responses to each; and dealing with synonyms by mapping different ways of saying the same thing to the same reply.

The knowledge bases of almost all chatbots are edited manually which restricts users to specific languages and domains. We developed a Java program to read a text from a machine readable text (corpus) and convert it to AIML format. The chatbot-training-program was built to be general, the generality in this respect implies, no restrictions on specific language, domain, or structure. Different languages were tested: English, Arabic, Afrikaans, French, and Spanish. We also trained with a range of different corpus genres and structures, including: dialogue, monologue, and structured text found in the Qur’an, and FAQ websites.

The chatbot-training-program is composed of four phases as follows:
- Reading module which reads the dialogue text from the basic corpus and inserts it into a list.
- Text reprocessing module, where all corpus and linguistic annotations such as overlapping, fillers and others are filtered.
- Converter module, where the pre-processed text is passed to the converter to consider the first turn as a pattern and the

second as a template. All punctuation is removed from the patterns, and the patterns are transformed to upper case.
- Producing the AIML files by copying the generated categories from the list to the AIML file.

An example of a sequence of two utterances from an English spoken corpus is:

```
<u who=F72PS002>
<s n="32"><w ITJ>Hello<c PUN>.
</u>
<u who=PS000>
<s n="33"><w ITJ>Hello <w NP0>Donald<c PUN>.
</u>
```

After the reading and the text processing phase, the text becomes:

```
F72PS002: Hello
PS000: Hello Donald
```

The corresponding AIML atomic category that is generated from the converter modules looks like:

```
<category>
<pattern>HELLO</pattern>
<template>Hello Donald</template>
</category>
```

As a result different prototypes were developed, in each prototype, different machine-learning techniques were used and a new chatbot was tested. The machine learning techniques ranged from a primitive simple technique like single word matching to more complicated ones like matching the least frequent words. Building atomic categories and comparing the input with all atomic patterns to find a match is an instance based learning technique. However, the learning approach does not stop at this level, but it improved the matching process by using the most significant words (least frequent word). This increases the ability of finding a nearest match by extending the knowledge base which is used during the matching process. Three prototypes will be discussed in this paper as listed below:

- The KGA prototype that is trained by a corpus of spoken Afrikaans. In this prototype two learning approaches were adopted. The first word and the most significant word (least frequent word) approach;
- The Qur'an prototype that is trained by the holy book of Islam (Qur'an): where in addition to the first word approach, two significant word approaches (least frequent words) were used, and the system was adapted to deal with the Arabic language and the non-conversational nature of Qur'an as shown in section 5;
- The FAQchat prototype that is used in the FAQ of the School of Computing at University of Leeds. The same learning techniques were used, where the question represents the pattern and the answer represents the template. Instead of chatting for just 10 minutes as suggested by the Loebner Prize, we advocate alternative evaluation methods more attuned to and appropriate to practical information systems applications. Our evaluation takes account of open-ended trials by real users, rather than artificial 10-minute trials as illustrated in the following sections.

The aim of the different evaluations methodologies is as follows:

- Evaluate the success of the learning techniques in giving answers, based on dialogue efficiency, quality and users' satisfaction applied on the KGA.
- Evaluate the ability to use the chatbot as a tool to access an information source, and a useful application for this, which was applied on the Qur'an corpus.
- Evaluate the ability of using the chatbot as an information retrieval system by comparing it with a search engine, which was applied on FAQchat.

4 Evaluation of the KGA prototype

We developed two versions of the ALICE that speaks Afrikaans language, Afrikaana that speaks only Afrikaans and AVRA that speaks English and Afrikaans; this was inspired by our observation that the Korpus Gesproke Afrikaans actually includes some English, as Afrikaans speakers are generally bilingual and "code-switch" comfortably. We mounted prototypes of the chatbots on websites using Pandorabot service[5], and encouraged

[5] http://www.pandorabots.com/pandora

open-ended testing and feedback from remote users in South Africa; this allowed us to refine the system more effectively.

We adopted three evaluation metrics:

- Dialogue efficiency in terms of matching type.
- Dialogue quality metrics based on response type.
- Users' satisfaction assessment based on an open-ended request for feedback.

4.1 Dialogue efficiency metric

We measured the efficiency of 4 sample dialogues in terms of atomic match, first word match, most significant match, and no match. We wanted to measure the efficiency of the adopted learning mechanisms to see if they increase the ability to find answers to general user input as shown in table 1.

Matching Type	D1	D2	D3	D4
Atomic	1	3	6	3
First word	9	15	23	4
Most significant	13	2	19	9
No match	0	1	3	1
Number of turns	23	21	51	17

Table 1. Response type frequency

The frequency of each type in each dialogue generated between the user and the Afrikaans chatbot was calculated; in Figure 2, these absolute frequencies are normalised to relative probabilities.

No significant test was applied, this approach to evaluation via dialogue efficiency metrics illustrates that the first word and the most significant approach increase the ability to generate answers to users and let the conversation continue.

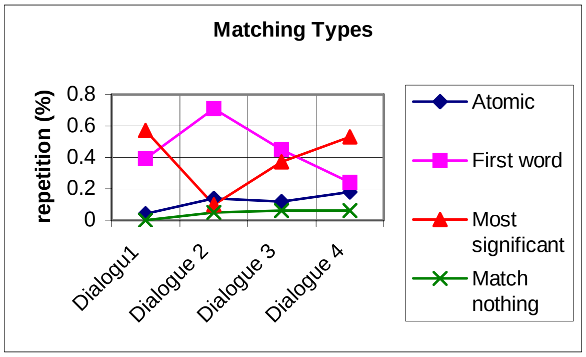


Figure 2. Dialogue efficiency: Response Type Relative Frequencies

4.2 Dialogue quality metric

In order to measure the quality of each response, we wanted to classify responses according to an independent human evaluation of "reasonableness": reasonable reply, weird but understandable, or nonsensical reply. We gave the transcript to an Afrikaans-speaking teacher and asked her to mark each response according to these classes. The number of turns in each dialogue and the frequencies of each response type were estimated. Figure 3 shows the frequencies normalised to relative probabilities of each of the three categories for each sample dialogue. For this evaluator, it seems that "nonsensical" responses are more likely than reasonable or understandable but weird answers.

4.3 Users' satisfaction

The first prototypes were based only on literal pattern matching against corpus utterances: we had not implemented the first word approach and least-frequent word approach to add "wildcard" default categories. Our Afrikaans-speaking evaluators found these first prototypes disappointing and frustrating: it turned out that few of their attempts at conversation found exact matches in the training corpus, so Afrikaana replied with a default "ja" most of the time. However, expanding the AIML pattern matching using the first-word and least-frequent-word approaches yielded more favorable feedback. Our evaluators found the conversations less repetitive and more interesting. We measure user satisfaction based on this kind of informal user feed back.

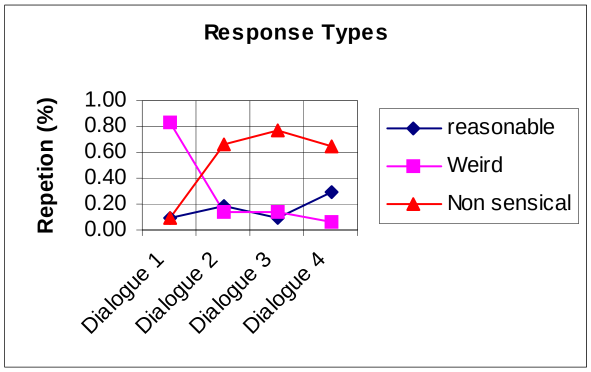


Figure 3. The quality of the Dialogue: Response type relative probabilities

5 Evaluation of the Qur'an prototype

In this prototype a parallel corpus of English/Arabic of the holy book of Islam was used, the aim of the Qur'an prototype is to explore the problem of using the Arabic language and of using a text which is not conversational in its nature like the Qur'an. The Qur'an is composed of 114 soora (chapters), and each soora is composed of different number of verses. The same learning technique as the KGA prototype were applied, where in this case if an input was a whole verse, the response will be the next verse of the same soora; or if an input was a question or a statement, the output will be all verses which seems appropriate based on the significant word. To measure the quality of the answers of the Qur'an chatbot version, the following approach was applied:

1. Random sentences from Islamic sites were selected and used as inputs of the English/Arabic version of the Qur'an.
2. The resulting transcripts which have 67 turns were given to 5 Muslims and 6 non-Muslims students, who were asked to label each turn in terms of:

- Related (R), in case the answer was correct and in the same topic as the input.
- Partially related (PR), in case the answer was not correct, but in the same topic.
- Not related (NR), in case the answer was not correct and in a different topic.

Proportions of each label and each class of users (Muslims and non-Muslims) were calculated as the total number over number of users times number of turns. Four out of the 67 turns returned no answers, therefore actually 63 turns were used as presented in figure 4.

In the transcripts used, more than half of the results were not related to their inputs. A small difference can be noticed between Muslims and non-Muslims proportions. Approximately one half of answers in the sample were not related from non-Muslims' point of view, whereas this figure is 58% from the Muslims' perspective. Explanation for this includes:

- The different interpretation of the answers. The Qur'an uses traditional Arabic language, which is sometimes difficult to understand without knowing the meaning of some words, and the historical story behind each verse.
- The English translation of the Qur'an is not enough to judge if the verse is related or not, especially given that non-Muslims do not have the background knowledge of the Qur'an.

Using chatting to access the Qur'an looks like the use of a standard Qur'an search tool. In fact it is totally different; a searching tool usually matches words not statements. For example, if the input is: "How shall I pray?" using chatting: the robot will give you all ayyas where the word "pray" is found because it is the most significant word. However, using a search tool[6] will not give you any match. If the input was just the word "pray", using chatting will give you the same answer as the previous, and the searching tool will provide all ayyas that have "pray" as a string or substring, so words such as: "praying, prayed, etc." will match.

Another important difference is that in the search tool there is a link between any word and the document it is in, but in the chatting system there is a link just for the most significant words, so if it happened that the input statement involves a significant word(s), a match will be found, otherwise the chatbot answer will be: "I have no answer for that".

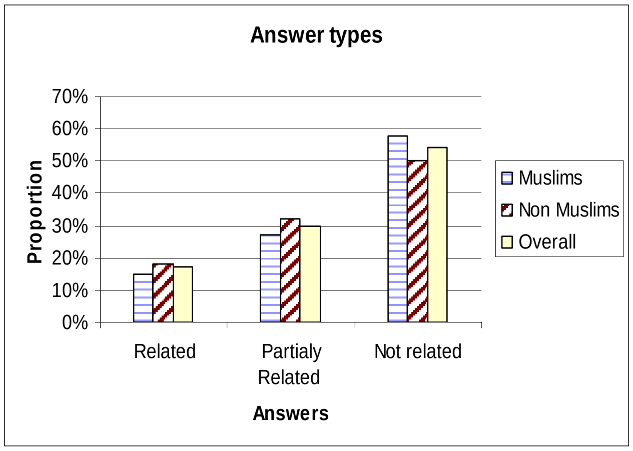


Figure4. The Qur'an proportion of each answer type denoted by users

6 Evaluation of the FAQchat prototype

To evaluate FAQchat, an interface was built, which has a box to accept the user input, and a button to send this to the system. The outcomes ap-

[6] http://www.islamicity.com/QuranSearch/

pear in two columns: one holds the FAQchat answers, and the other holds the Google answers after filtering Google to the FAQ database only. Google allows search to be restricted to a given URL, but this still yields all matches from the whole SoC website (http://www.comp.leeds.ac.uk) so a Perl script was required to exclude matches not from the FAQ sub-pages.

An evaluation sheet was prepared which contains 15 information-seeking tasks or questions on a range of different topics related to the FAQ database. The tasks were suggested by a range of users including SoC staff and research students to cover the three possibilities where the FAQchat could find a direct answer, links to more than one possible answer, and where the FAQchat could not find any answer. In order not to restrict users to these tasks, and not to be biased to specific topics, the evaluation sheet included spaces for users to try 5 additional tasks or questions of their own choosing. Users were free to decide exactly what input-string to give to FAQchat to find an answer: they were not required to type questions verbatim; users were free to try more than once: if no appropriate answer was found; users could reformulate the query.

The evaluation sheet was distributed among 21 members of the staff and students. Users were asked to try using the system, and state whether they were able to find answers using the FAQchat responses, or using the Google responses; and which of the two they preferred and why.

Twenty-one users tried the system; nine members of the staff and the rest were postgraduates. The analysis was tackled in two directions: the preference and the number of matches found per question and per user.

6.1 Number of matches per question

The number of evaluators who managed to find answers by FAQchat and Google was counted, for each question.

Results in table 2 shows that 68% overall of our sample of users managed to find answers using the FAQchat while 46% found it by Google. Since there is no specific format to ask the question, there are cases where some users could find answers while others could not. The success in finding answers is based on the way the questions were presented to FAQchat.

Users /Tool	Mean of users finding answers		Proportion of finding answers	
	FAQchat	Google	FAQchat	Google
Staff	5.53	3.87	61%	43%
Student	8.8	5.87	73%	49%
Overall	14.3	9.73	68%	46%

Table 2: Proportion of users finding answers

Of the overall sample, the staff outcome shows that 61% were able to find answers by FAQchat where 73% of students managed to do so; students were more successful than staff.

6.2 The preferred tool per each question

For each question, users were asked to state which tool they preferred to use to find the answer. The proportion of users who preferred each tool was calculated. Results in figure 5 shows that 51% of the staff, 41% of the students, and 47% overall preferred using FAQchat against 11% who preferred the Google.

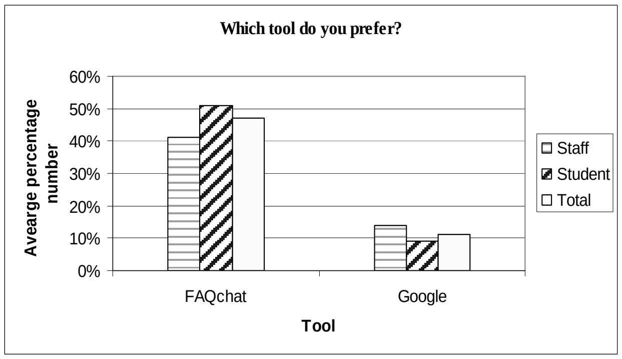


Figure5. Proportion of preferred tool

6.3 Number of matches and preference found per user

The number of answers each user had found was counted. The proportions found were the same. The evaluation sheet ended with an open section inviting general feedback. The following is a summary of the feedback we obtained:

- Both staff and students preferred using the FAQchat for two main reasons:

1. The ability to give direct answers sometimes while Google only gives links.
2. The number of links returned by the FAQchat is less than those returned by Google for some questions, which saves time browsing/searching.

- Users who preferred Google justified their preference for two reasons:

1. Prior familiarity with using Google.
2. FAQchat seemed harder to steer with carefully chosen keywords, but more often did well on the first try. This happens because FAQchat gives answers if the keyword matches a significant word. The same will occur if you reformulate the question and the FAQchat matches the same word. However Google may give different answers in this case.

To test reliability of these results, the t=Test were applied, the outcomes ensure the previous results.

7 Conclusion

The Loebner Prize Competition has been used to evaluate the ability of chatbots to fool people that they are speaking to humans. Comparing the dialogues generated from ALICE, which won the Loebner Prize with real human dialogues, shows that ALICE tries to use explicit dialogue-act linguistic expressions more than usual to re enforce the impression that users are speaking to human.

Our general conclusion is that we should NOT adopt an evaluation methodology just because a standard has been established, such as the Loebner Prize evaluation methodology adopted by most chatbot developers. Instead, evaluation should be adapted to the application and to user needs. If the chatbot is meant to be adapted to provide a specific service for users, then the best evaluation is based on whether it achieves that service or task

References

Abu Shawar B and Atwell E. 2003. Using dialogue corpora to retrain a chatbot system. In *Proceedings of the Corpus Linguistics 2003 conference*, Lancaster University, UK, pp681-690.

Batacharia, B., Levy, D., Catizone R., Krotov A. and Wilks, Y. 1999. CONVERSE: a conversational companion. In Wilks, Y. (ed.), *Machine Conversations*. Kluwer, Boston/Drdrecht/London, pp. 205-215.

Colby, K. 1999a. Comments on human-computer conversation**.** In Wilks, Y. (ed.), *Machine Conversations*. Kluwer, Boston/Drdrecht/London, pp. 5-8.

Colby, K. 1999b. Human-computer conversation in a cognitive therapy program. In Wilks, Y. (ed.), *Machine Conversations*. Kluwer, Boston/Drdrecht/London, pp. 9-19.

Epstein R. 1992. Can Machines Think?. *AI magazine*, Vol 13, No. 2, pp80-95

Garner R. 1994. The idea of RED, [Online], http://www.alma.gq.nu/docs/ideafred_garner.htm

Hirschman L. 1995. The Roles of language processing in a spoken language interface. In Voice Communication Between Humans and Machines, D. Roe and J. Wilpon (Eds), National Academy Press Washinton, DC, pp217-237.

Hutchens, J. 1996. *How to pass the Turing test by cheating*. [Onlin], http://ciips.ee.uwa.edu.au/Papers/, 1996

Hutchens, **T., Alder, M.** 1998. Introducing MegaHAL. [Online], http://cnts.uia.ac.be/conll98/pdf/271274hu.pdf

Loebner H. 1994. In Response to lessons from a restricted Turing Test. [Online], http://www.loebner.net/Prizef/In-response.html

Maier E, Mast M, and LuperFoy S. 1996. Overview. In Elisabeth Maier, Marion Mast, and Susan LuperFoy (Eds), *Dialogue Processing in Spoken Language Systems*, , Springer, Berlin, pp1-13.

McTear M. 2002. Spoken dialogue technology: enabling the conversational user interface. *ACM Computing Surveys.* Vol. 34, No. 1, pp. 90-169.

Shieber S. 1994. Lessons from a Restricted Turing Test. *Communications of the Association for Computing Machinery*, Vol 37, No. 6, pp70-78

Turing A. 1950. Computing Machinery and intelligence. *Mind* 59, 236, 433-460.

Weizenbaum, J. 1966. ELIZA-A computer program for the study of natural language communication between man and machine. *Communications of the ACM.* Vol. 10, No. 8, pp. 36-45.

Whalen T. 2003. My experience with 1994 Loebner competition, [Online], http://hps.elte.hu/~gk/Loebner/story94.htm

Panel on Spoken Dialog Corpus Composition and Annotation for Research

Organizers: *Giuseppe DiFabbrizio, Dilek Hakkani-Tür, Oliver Lemon, Mazin Gilbert, Alex Rudnicky*

The goal of this forum is to provide researchers from various institutes with the opportunity to comment on a proposed NSF-sponsored data collection plan for a spoken dialog corpus. The corpus is to be used for research in speech recognition, spoken language understanding, dialog management, machine learning, and language generation. Currently, there exists a corpus with over 600 dialog interactions, collected from users using the Discoh system (from the IEEE SLT 2006 workshop) and the Conquest system (from ICSLP 2006) to obtain general information about conference services. These systems were created as part of a joint collaboration between CMU, ATT, Edinburgh and ICSI.

The workshop panel will host a number of invited researchers who have received a subset of the corpus and annotation. An open discussion will be held to obtain comments from workshop participants to help finalize the annotation guidelines. If you would like to provide feedback with respect to the annotation plan and receive a sample of the dialog interactions then please send an email to info@discoh.org

Author Index