Still Talking to Machines (Cognitively Speaking)

Steve Young

Cambridge University Engineering Department
sjy@eng.cam.ac.uk

Abstract

This overview article reviews the structure of a fully statistical spoken dialogue system (SDS), using as illustration, various systems and components built at Cambridge over the last few years. Most of the components in an SDS are essentially classifiers which can be trained using supervised learning. However, the dialogue management component must track the state of the dialogue and optimise a reward accumulated over time. This requires techniques for statistical inference and policy optimisation using reinforcement learning. The potential advantages of a fully statistical SDS are the ability to train from data without hand-crafting, increased robustness to environmental noise and user uncertainty, and the ability to adapt and learn on-line.
Index Terms: spoken dialogue systems, reinforcement learning, speech understanding, speech synthesis, natural language generation

1. Introduction

At Interspeech 2002 in Denver, I suggested that it should be possible to build a complete spoken dialogue system in which every component was based on a statistical model with parameters estimated from data[1]. The potential advantages of such a system would include lower development cost, increased robustness to noise and the ability to learn on-line so that performance would continue to improve over time.

Eight years later, fully statistical systems are now being built in the laboratory and their potential demonstrated. This talk will review the basic principles of statistical dialogue systems and discuss the major lessons learnt so far. This will be illustrated primarily by describing a number of systems and components developed at Cambridge but related work at other institutions will be referenced where relevant. The focus will be on dialogue management and in particular the representation of dialogue state, approaches to belief monitoring, parameter estimation and policy optimisation. Probabilistic components for speech recognition, semantic decoding, natural language generation and synthesis will also be briefly mentioned.

2. Architecture of a Statistical SDS

The basic architecture of a statistical spoken dialogue system is shown in Fig. 1. The user speaks producing a noisy acoustic signal y which is converted by a speech recogniser into a sequence of words w. A semantic decoder then converts the word sequence w into an abstract representation of the user's intended dialogue act v. The dialogue manager maintains an internal state s representing the inferred user goal g, the hypothesised user dialogue act u and any relevant dialogue history h[2]. Based on the dialogue state s, the dialogue manager selects an action a in the form of a system dialogue act. This dialogue act is converted into an output message m by a natural language generator and finally into speech x by a synthesiser. The user responds to x with another input y and the cycle repeats.

The modular decomposition shown in Fig. 1 is both conceptually and practically convenient. Each module can be designed, constructed and optimised individually and then integrated to form a working system. However, any deterministic system built this way would be sub-optimal for two reasons. Firstly, if each module is entirely independent, then it will be forced to take early decisions without having access to all of the necessary contextual evidence. Secondly, local optimisation of individual modules does not guarantee global optimality. Ensuring that all of the components are based on probabilistic models provides a potential solution to both of these problems.

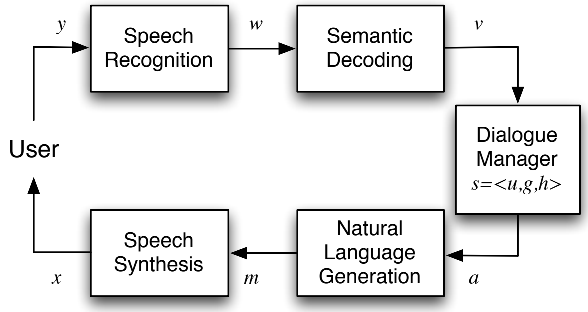

Figure 1: *Architecture of a Spoken Dialogue System*

In a fully statistical system, the signal at each interface is a distribution and not the single best hypothesis. Typically, this distribution is approximated by an N-best list [3] or some form of lattice [4]. Thus on the input side, the output w of the speech recogniser with parameters λ_{asr} is the distribution $p(w|y, \lambda_{asr})$ where y is the noisy speech input from the user. The semantic decoder then combines this distribution with its own model $p(v|w, \lambda_{sd})$ to give

$$p(v|y, \lambda_{asr}, \lambda_{sd}) = \sum_{w} p(v|w, \lambda_{sd})p(w|y, \lambda_{asr}) \quad (1)$$

On the output side, similar principles can be applied except that the output distribution must be sampled to select an exemplar utterance to present to the user. The NLG component with parameters λ_{nlg} converts input dialogue acts a into a sequence of phrases m which are then converted by a synthesiser with parameters λ_{ss} into output speech x. Thus,

$$p(x|a, \lambda_{ss}, \lambda_{nlg}) = \sum_{m} p(x|m, \lambda_{ss})p(m|a, \lambda_{nlg}) \quad (2)$$

Typically, the sample value of x is chosen to maximise $p(x|a)$ but alternatives could be used if variability in the output was required. At first sight this sampled-distribution approach might seem a little odd since it would surely be simpler to just generate a single utterance m and convert it to speech. However, this would prevent the generation process from taking account of the ability of the synthesiser to produce different forms of the same underlying message. The probabilistic framework allows the NLG and synthesis to be jointly optimised and this will become increasingly important as the speech output from SDS is required to be much more sensitive to the linguistic and emotional context[5].

Both the input and output components of an SDS implement statistical mappings which can be trained from data using supervised learning. The dialogue manager component is rather different since its role is to select an appropriate response to each user input with the long-term goal of satisfying the user's requirements. Furthermore it must do this under conditions of great uncertainty since users will often be unclear about their requirements and the speech recogniser will further confound matters by making errors. For example in the tourist information system described below, the user's goal will typically be a venue satisfying certain requirements (e.g. a cheap italian restaurant on the north side of town). These requirements will need to be conveyed to the system over a number of dialogue turns. Hence, the dialogue manager will have many choices at each turn including confirming the given requirements, asking for further requirements or offering a venue. Sometimes there will be no venue that matches and the system must then negotiate with the user to relax the requirements until an appropriate venue can be found.

Conventional SDS systems implement the dialogue manager using deterministic decision logic. Essentially each input is treated as a command which is decoded by a set of handcrafted decision rules to generate a response. To assist the decision making process, the system must maintain a dialogue state which records the current best estimate of the user's goal and the dialogue history. The problem with this conventional approach is that it is expensive to produce, inflexible and fragile.

In [6] it is argued that any human-computer interface, but especially an SDS, must be able to support four key features which are characteristic of cognitive behaviour:

- able to support reasoning and inference - necessary to interpret noisy inputs in context to robustly resolve ambiguities and minimize errors;
- able to plan under uncertainty - the communicative goals must be defined objectively and strategies then optimized to meet the objectives as efficiently as possible;
- able to adapt on-line to changing circumstances. Contexts and environments change and behaviour must adapt to maintain an acceptable level of performance;
- able to learn from experience. In addition to short-term adaptation, an SDS should be capable of learning from its own interactions over the long term. The more it is used, the smarter it should become.

Interfaces which have these four essential properties are referred to as *cognitive user interfaces* and the key to building such an interface is dialogue management based on the partially observable Markov decision process (POMDP)[7].

A POMDP-based dialogue manager has two essential components. Firstly, a POMDP maintains a distribution $b(s)$ over all possible dialogue states called the *belief state*. User inputs are not treated as commands, instead they are noisy observations. Bayesian inference is then used to update the belief state at each dialogue turn given the noisy observation $o = p(v)$[1] and the previous system action a

$$\begin{aligned} b'(s') &= P(s'|o', a, b) \\ &= k \cdot P(o'|s', a) \sum_s P(s'|a, s) b(s). \end{aligned} \tag{3}$$

where $k = 1/P(o'|a, b)$ is a normalisation constant [8]. This belief update process is referred to in the POMDP literature as belief monitoring.

Secondly, a policy $\{\pi : b \rightarrow a\}$ provides a mapping from belief states b to actions a. The policy is designed to maximise the cumulative sum R of rewards $r(b, a)$ obtained at each dialogue turn. The reward function $r(b, a)$ therefore serves as an objective metric of performance and should be chosen to reflect the design criteria of the application. Typically for spoken dialogues it will consist of a small negative reward at each turn, a large positive reward for successfully completing the dialogue and a large negative reward for failure. POMDP optimisation then involves finding the policy which maximises the expected cumulative reward R over the course of a dialogue[2].

To elaborate on how the above architecture can be realised in practice, the next few sections describe the components of a fully statistical SDS built at Cambridge in the Tourist Information domain.

3. The CamInfo System

The CamInfo system is based on the architecture shown in Fig. 1. All of the components are statistical and all of the component interfaces are implemented as N-best list approximations of the output distributions. The system is server based and it uses sipgate[3] to make it accessible via the telephone or via a web interface which also displays an interactive map. CamInfo can help the user locate places to stay, places to eat, places to drink, places to see and a variety of other entities such as shops, sports facilities, transport and general amenities. Typical requests might be "I want a cheap restaurant in Cherry Hinton", "I'd like a hotel near the river", "Is there anywhere to get a drink near Trinity College?".

3.1. Ontology

The information in the CamInfo database is organised according to an ontology as shown in Fig. 2. As can be seen, it is quite large with several layers of hierarchy. The top level defines a generic venue and the layer below that defines some general classes of venue such as places-to-eat, places-to-stay, places-to-drink, places-to-see, university-venues, sports-venues, etc. As an example, Fig. 3 shows the main elements of the ontology relating to places-to-eat. On system startup, CamInfo compiles this ontology to create the internal representation of a user goal (see 3.4 below).

3.2. Dialogue Acts

As noted in section 2, CamInfo uses a common abstract representation called a *dialogue act* for both inputs and outputs to the dialogue manager. The definition and form of dialogue acts are critical since they affect the design of all processing below the word level i.e. semantic decoding, dialogue management and natural language generation.

[1] In practice this will be an N-best list of user dialogue acts.
[2] The cumulative reward is also called the *return*.
[3] See http://www.sipgate.co.uk

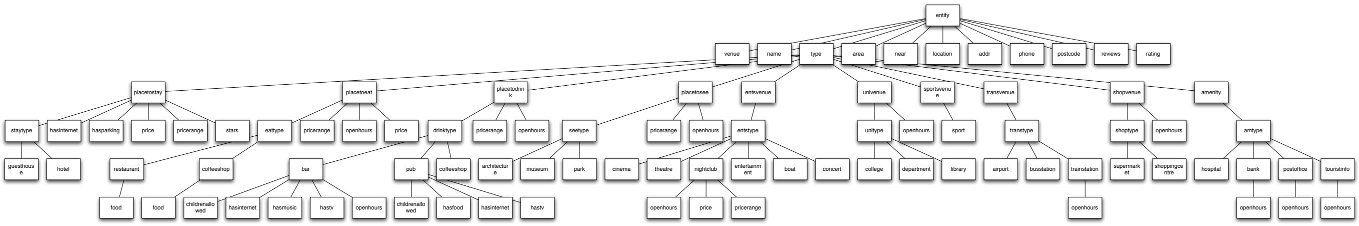

Figure 2: *Full Ontology of the CamInfo System*

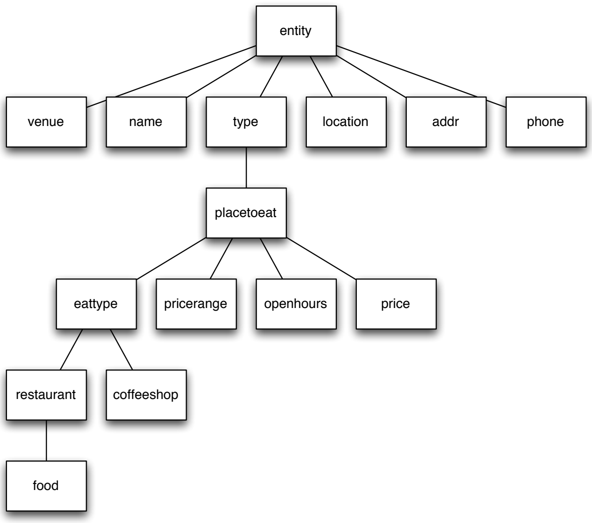

Figure 3: *Ontology of Places to Eat*

Act	Description
hello(a=x,b=y,...)	open a dialog and give info a=x, b=y, ...
inform(a=x,b=y,...)	give information a=x, b=y, ...
request(a, b=x,...)	request value for a given b=x ...
reqalts(a=x,..)	request alternative with a=x,...
confirm(a=x,b=y,..)	explicitly confirm a=x,b=y,..
confreq(a=x,.., d)	implicitly confirm a=x,.. and request d
select(a=x,a=y)	select either a=x or a=y
affirm(a=x, b=y,...)	affirm and give further info a=x, b=y, ...
negate(a=x)	negate and give corrected value a=x
deny(a=x)	deny that a=x
bye()	close a dialogue

Table 1: The principal dialogue acts used by all Cambridge dialogue systems.

Dialogue acts in all of the Cambridge dialogue systems[4] take the form *acttype(*$a_1{=}v_1, a_2{=}v_2, ..$*)* where *acttype* denotes the type of dialogue act and the arguments are *act items* consisting of attribute-value pairs[5]. Attributes refer to the key information nodes in the user goal tree g of the dialogue state which is compiled from the ontology described in 3.1. The values of an attribute are either sub-nodes as in "type=placetoeat" or atomic values as in "food=chinese". In some contexts, the value can be omitted, for example, where the intention is to query the value of an attribute. Note that the dialogue act types are application domain independent whereas the attributes and their values are derived from the application ontology. The most common acts are listed in Table 1 and a simple dialogue illustrating their use is shown in Table 2. A full description of the dialogue act set is given in [9].

3.3. Speech Understanding

The speech understanding function in CamInfo is implemented by combining a HTK-based speech recogniser with a statistical semantic decoder(see Fig. 1). The speech recogniser converts the acoustic speech signal y into a 10-best list of recognised word sequences $\{w_i\}$ and the semantic decoder then converts each w_i into one or more user dialogue acts. Duplicate dialogue acts are merged to form a single N-best list of user dialogue acts and their corresponding posterior probabilities $\{v_j, p_j\}$. Hence, the actual number of dialogue acts passed to the dialogue manager varies depending on the speaker and noise level.

Speech recognition in the CamInfo system is provided by ATK [10] which is a real-time multi-threaded recogniser compatible with acoustic and language model resources built using HTK [11]. The CamInfo recogniser has a vocabulary of around 3000 words. It uses state-clustered word-internal triphones trained on 40 hours of audio data to produce 4091 distinct states each with 4 Gaussians per state. The speech waveform is coded using PLP coefficients plus their 1st, 2nd and 3rd derivatives mapped to a 39-dimensional vector using HLDA[12]. The recogniser runs with a fixed trigram language model trained on around 400k words of in-domain data smoothed with about 80M words of general corpus data[13]. There is no grammar switching so the user can say anything at any time.

Semantic decoding requires a sequence of words to be converted into a dialogue act plus some number of attribute-value pairs. Early attempts to solve this problem assumed that it was sufficient to tag each word with a semantic label and hence simple HMM-taggers could be used[14]. However, this fails with utterances which have embedded phrases where some form of hierarchical structure must be supported in order to analyse the utterance correctly.

A review of various approaches to statistical semantic decoding is given in [15, 16]. However, a critical issue for configuring a decoder to a specific spoken dialogue application is the cost of producing training data in the form of matched pairs $< w, v >$ of word string and corresponding dialogue act. This cost is greatly reduced if the need for an explicit alignment between w and v can be avoided. For the CamInfo sys-

[4]In addition to the tourist information system described here, we have also built systems for Appointments Scheduling and the Pittsburgh Let's Go bus information service.

[5]Attributes are referred to as *slots* in some dialogue systems.

	Utterance	Dialogue Act
U:	Hi, I am looking for somewhere to eat.	hello(task=find,type=restaurant)
S:	You are looking for a restaurant. What type of food do you like?	confreq(type=restaurant,food)
U:	I'd like an Italian somewhere near Trinity College.	inform(food=Italian,near="Trinity")
S:	Strada is a nice Italian restaurant near Trinity.	inform(name="Strada", type=restaurant, food=Italian, near"Trinity")
U:	Is it reasonably priced?	confirm(pricerange=moderate)
S:	Yes, Strada is in the moderate price range.	affirm(name="Strada", pricerange=moderate)
U:	What is the phone number?	request(phone)
S:	The number of Strada is 385456.	inform(name="Strada", phone="385456")
U:	Ok, thank you goodbye.	bye()

Table 2: An example dialogue and its representation at the dialogue act level.

tem, we have experimented with three different decoders, all of which can be trained on unaligned data: the hidden vector state decoder[17, 18], a transformation-based decoder [19] and an SVM-based decoder[20].

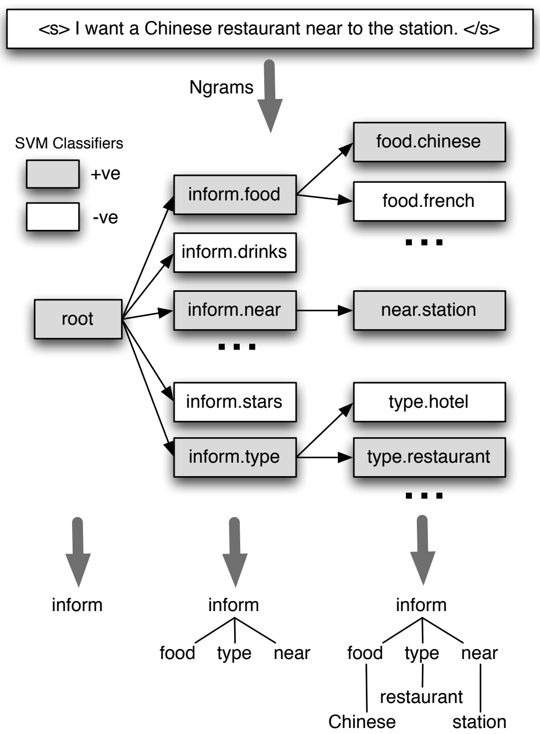

Figure 4: *Semantic parse tree generation using SVM classifiers. The input utterance is converted to N-gram features and input to a bank of SVM classifiers which detect parent/offspring nodes in the semantic parse tree.*

Due to its simplicity, high performance and very low run-time compute cost, we now mostly use the SVM-based decoder. This is based on the very simple idea that a semantic parse tree such as the one shown in Fig. 4 can be constructed from a set of tuples $< n_1, n_2, n_3, ... >$ where node n_i is the parent of node n_{i+1}. Using the N-grams of w as classification features, a binary SVM is trained for every possible tuple to determine whether or not w has a semantic parse containing that tuple. All that is then required to decode a previously unseen utterance w is to apply all of the SVM classifiers to w and collect the set of all tuples that are predicted to be in w's semantic tree. These tuples are then glued together to form the required tree. The choice of tuple size is a trade-off between classifier accuracy and fidelity of the tree reconstruction. For the CamInfo domain, a tuple size of 2 appears to be adequate. Of course, there is potentially a very large number of SVMs to apply but lexicalising terminal nodes reduces that number, and each classification is very fast being a simple scalar product. By searching for higher level tuples first, the search for lower level tuples can be limited to those which are consistent with those already found. Finally, the SVM margins can be used to compute a confidence score for the overall parse. As shown in [20], this simple SVM decoder can achieve a dialogue act accuracy and an F-measure on attribute-value pairs of 95% on clean data, out-performing a hand-crafted parser.

3.4. Dialogue Management

As noted in section 2, the CamInfo dialogue state s is decomposed into three factors $s = \{g, u, h\}$ where g represents the user's goal, u represents the user input act and h represents the dialogue history [2]. All of these are highly complex. Furthermore, since speech recognition error rates are typically high, there is significant uncertainty in the decoded user utterances u and this propagates uncertainty into g and h. In addition, the system action space must cover every possible system response so policies must map from complex and uncertain dialog states into a large space of possible actions. All of these factors combine to make implementation of a POMDP-based dialogue system a significant challenge.

The first step towards building a practical system is to make belief monitoring tractable by simplifying the representation of the state distribution. For example, suppose that in the tourist information domain the user's goal consisted of four discrete values: type, location, price, food. Exact belief monitoring would require that the full joint distribution over $P(type, location, price, food)$ be maintained; but even a modest number of types, locations, price-points and food types would quickly render the full joint probability impossibly large. One of the simplest ways of dealing with this is to use an M-best approximation whereby the probability of all state values are ranked and pruned to retain only the M most likely states. For instance, the ranked list for the tourist example might be as follows

P(hotel, east, cheap, none)	=	0.65
P(hotel, west, cheap, none)	=	0.21
P(restaurant, east, cheap, italian)	=	0.08
P(bar, east, cheap, none)	=	0.04
P(hotel, east, expensive, none)	=	0.01
...		

where all combinations other than those shown have such low probabilities that they are not worth maintaining. This idea is developed further in the Hidden Information State (HIS) dialogue manager described below.

A second approach to belief state approximation is to factor the joint distribution by making some independence assumptions. For example, from knowledge of the domain it might be argued that the kind of food and the price of a venue depends only on its type, and the type and location are independent, then

$$P(type, location, price, food) \approx P(price|type)P(food|type)P(type)P(food) \quad (4)$$

This results in a Bayesian network representation for the dialogue state and forms the basis for the Bayesian Update of Dialogue State (BUDS) dialogue manager also described below.

In order that belief monitoring can accurately track the dialogue state as it evolves, it is important to minimise the approximations made in the representation of the state space. However, policy implementation and optimisation do not require such a detailed representation of the state. For example, when the user asks for a restaurant in a particular area of town, the belief state must record all of the possible areas that the user might have said. However, to generate an appropriate response it is only necessary for the system to decide between a few high level options such as to confirm the last user input, ask for further information or suggest a venue. Thus the second step towards building a practical POMDP-based system is to introduce the notion of a *summary belief space*. At each input turn, the so-called *master belief space* is updated and then mapped into summary space where the appropriate next *summary action* is determined. The summary action is then mapped back into master space and converted into a system response. Since the policy is maintained and optimised in summary space, tractability issues are much reduced.

The HIS and BUDS dialogue managers exemplify the two alternative approaches to belief state approximation described in step one above and they both exploit a form of master-summary space mapping. They have both been implemented within the CamInfo system in order to understand the impact of the differing approximations they make. They are now briefly described.

3.4.1. The Hidden Information State Dialogue Manager

If the factorisation of the dialogue state, $s = \{u, g, h\}$ is plugged into the belief update equation (3) and some reasonable independence assumptions are made, it is straightforward to show that

$$b'(u', g', h') = k \cdot \underbrace{P(o'|u')}_{\text{observation model}} \cdot \underbrace{P(u'|g', a)}_{\text{user action model}} \cdot \sum_{g,h} \underbrace{P(g'|g, a)}_{\text{user goal model}} \cdot \underbrace{P(h'|g', u', h, a)}_{\text{dialogue history model}} \cdot b(g, h) \quad (5)$$

where the primes indicate the next time step [21]. As shown by under-braces, the belief update equation for a spoken dialogue system involves four distinct probability models. The user goal model and the history model represent the dynamics of the underlying Markov decision process. In the HIS system, it is assumed that the user's goal changes changes only rarely so the probability of change is low. It is also assumed that the history model can be replaced by a deterministic finite state grounding model. More interesting are the observation and user action models. The observation model encodes the error characteristics of the speech understanding system and the user action model encodes the probability that the user would speak u' given the goal g' and the last system prompt a. Since the observation is an N-best list of hypotheses, the user action model effectively allows this list to be re-ranked according to the context. Thus the user action model provides a context sensitive filter which is extremely effective at reducing errors, especially in high noise conditions [22].

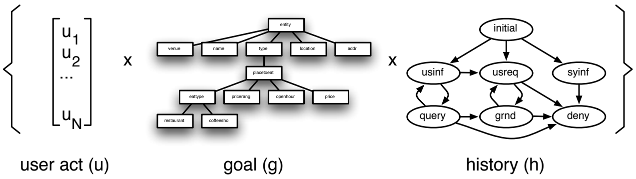

Figure 5: The HIS Belief Space. Each combination of user dialogue act, goal partition, and set of history states constitutes a single partition of states in belief space. The history states record grounding and query status information (see [23] for details)).

To simplify belief monitoring further, the HIS system groups user goal states into equivalence classes called partitions. At the start of a dialogue all goal states are in a single partition. As new evidence is received via the input user acts, the partitions are expanded to preserve the different possible goals. This expansion follows a set of ontology rules derived from the database and it is tree-structured such that when a partition is split by new information $x = a$, the new partition explicitly records the fact that $x = a$ and the existing partition is updated to record the fact that $x = \bar{a}$ [24]. Since all of the goals in a partition are indistinguishable based on the evidence so far, beliefs can be updated at the partition level rather than at the individual state level and this significantly reduces the computational load. The goal g in (5) now denotes a partition and since by definition, g is the parent of g', the sum over g is no longer required.

The resulting HIS state space is illustrated schematically in Fig. 5. Each partition of HIS states consists of a hypothesised last user act, a user goal partition and a set of grounding information. The latter constitutes the dialog history such that every node in the partition tree has a grounding state which changes according to a finite state transition network. Thus overall, the HIS state space consists of all possible hypothesised user acts, and all possible partitions combined with all possible combinations of grounding states. The probability of every partition of states in this set is evaluated, rank ordered and pruned. The system typically maintains 300 to 3000 active partitions and together these constitute its belief state b.

Master-summary space mapping and policy representation in the HIS system are shown in Fig. 6. The summary space consists of a fixed-length vector of features such as the prob-

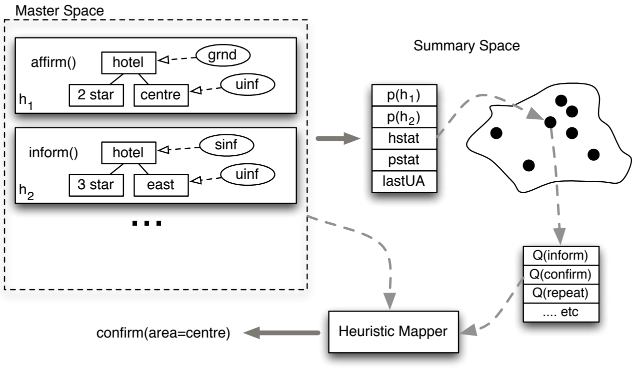

Figure 6: HIS Master-Summary state mapping. The summary space consists of a fixed-length vector of features which is mapped into a discrete grid point in summary space by a vector quantizer. Each grid point has an associated array of Q-values and associated actions. The action with the highest Q value which can be mapped back into master space is selected (see [25] for details).

ability of the top state in master space, the probability of the next-to-top state, various discrete variables representing status information, and the last user act. This summary vector is then mapped into a fixed grid point in summary space by a vector quantizer [26, 27]. Each grid point has a set of associated summary system actions ranked according to their Q values[6]. Initially the action with the highest Q value is selected and mapped back to master space by an inverse mapping function which in most cases assumes that the topic of the system action concerns the most likely partition in master belief space. In some cases, the selected summary action cannot be mapped back to master space in which case the action with the next highest Q value is tried [25]. This is illustrated in Fig. 6 where the highest ranked action is to inform the user about a venue, but since no venue has yet been determined, the 2nd choice confirm action is selected and mapped back to master space by heuristically selecting a specific attribute to confirm from the top-ranked hypothesis.

The vector quantization of the continuous summary space converts the HIS POMDP into a simple discrete Markov Decision Process (MDP) for which many optimization algorithms exist. In fact, the standard HIS system uses the Monte Carlo Control algorithm in conjunction with a user simulator to estimate the optimal action set via on-line reinforcement learning [28] (see section 4 below).

3.4.2. The Bayesian Update of Dialog State System

The HIS system described in the previous section illustrates how the M-best approach to belief space approximation coupled with a master-summary state mapping can lead to a tractable real-world dialog system. However, it suffers from two major problems. Firstly, the M-best approximation makes it difficult to represent a state transition matrix and hence, the HIS system assumes that the probability that the user's goal will change during the course of a dialogue is very low. Secondly, the probability models in the HIS system are hybrids of deterministic rules and statistical models which are difficult to train automatically from data.

[6] $Q(s, a)$ is the reward expected from taking action a when in state s

As explained in the introduction to this subsection, an alternative approach is to use a Bayesian network representation for the dialogue state. This form of approximation retains the ability to properly represent system dynamics and to use fully parametric models but at the cost of ignoring much of the conditional dependency inherent in real world domains.

The BUDS system uses the same factorization of the dialogue state $s = \{u, g, h\}$ as the HIS system, but it further factorizes each component into concepts[29]. For example, in the tourist information domain, the user might be interested in the concepts *location*, *price*, *food*, *room-rate*, *music*, etc. Most of these concepts will be dependent on the type of venue involved (restaurant, bar, hotel, etc) but otherwise they can be deemed to be independent. This results in a dynamic Bayesian network structure of the form shown in Fig. 7 which shows the venue *type* and one dependent concept *food*. Note that in the actual CamInfo system there are about 20 concepts in total. Each concept c has three principle nodes: a goal node g_c with values ranging over the user's possible choices; a user act node u_c denoting the type of the last user act or *null* if the last user act did not mention this concept; and a history node h_c taking the values of a simple grounding model such as (*initial, mentioned, grounded*). All u_c nodes depend on a single node representing the full user act and this in turn depends on the observation, which like the HIS system is normally an N-best list of hypothesized user dialogue acts. System dynamics are represented by adding dependencies on the equivalent node in the previous time slot. In the BUDS system, both the goal nodes and the history nodes depend on their previous values.

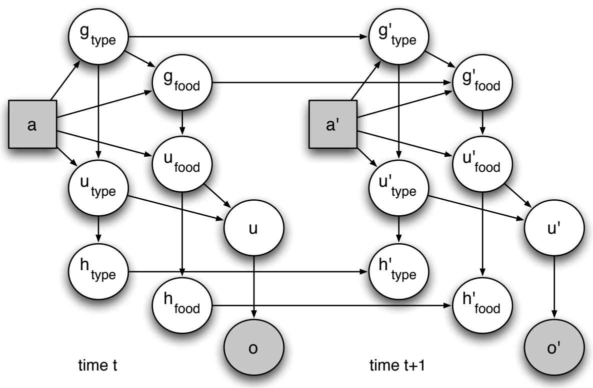

Figure 7: Bayesian Update of Dialogue State (BUDS). BUDS uses a dynamic Bayesian network in which the dialogue state is decomposed into *slots* representing features such as kind of *food*, *price*, *location*. Each slot has goal g, an associated user dialogue act u, and history information h. The slots are mostly independent except that all slots depend on the venue *type*.

The BUDS system represents all of the relevant dialogue state information in a single Bayesian network structure. Belief monitoring can therefore utilize any of the existing algorithms for approximate inference in a Bayesian network. In the BUDS system, loopy belief propagation (LBP) is used [30]. However, to make this run in real-time, various optimizations are essential. For example, goal nodes may range over a large set of values whereas only a very few of these values will ever be mentioned in a single dialogue. Standard LBP can thus be made much faster by partitioning the values in each slot similar to the way states are partitioned in the HIS system. This typically reduces the effective cardinality of each slot down to 2 or 3 and this has a dramatic effect on computation times. Another very effective optimization is to assume that the probability of user goal changing is constant. This reduces the cardinality of a user goal transition matrix with n possible values from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$ [31].

Distributing the belief space over a large number of factors requires a different approach to policy representation since direct mapping of each master state into summary space is no longer possible. In the BUDS system, the policy is represented by a softmax function with parameters $\boldsymbol{\theta}$ as follows

$$\pi(a|b,\boldsymbol{\theta}) = \frac{e^{\boldsymbol{\theta}.\boldsymbol{\phi}_a(b)}}{\sum_{a'} e^{\boldsymbol{\theta}.\boldsymbol{\phi}'_a(b)}} \tag{6}$$

where $\boldsymbol{\phi}_a(b)$ is a *basis function* for action a. These basis functions can be separated into components to allow the separate effects of each concept goal node in the Bayesian network to contribute to the overall policy

$$\boldsymbol{\phi}_a(b)^\top = [\boldsymbol{\phi}_{a,1}(b)^\top, \ldots, \boldsymbol{\phi}_{a,G}(b)^\top, \boldsymbol{\phi}_{a,*}(b)^\top]. \tag{7}$$

where the indices $1..G$ range over the relevant goal nodes and the final term $\boldsymbol{\phi}_{a,*}(b)^\top$ allows global information to be incorporated such as the number of database entries matching the most likely user goal. In effect, the mapping from master to summary space is performed implicitly via function approximation. The resulting parametric policy can be optimized by maximizing the expected reward and in the BUDS system, the Natural Actor Critic algorithm has been found to be the very effective for this [32].

3.5. Response Generation

The process of generating a spoken output from CamInfo is divided into two components. Firstly, the system dialogue act output by the dialogue manager is converted to a list of candidate word strings by a Natural Language Generation (NLG) component and then secondly, the word string is converted to speech by a HMM-based speech synthesiser. Attributes and values in the dialogue act which are new to the dialogue or are the focus of a confirmation are marked for emphasis and these emphasis marks are carried over to the word or phrase describing that attribute.

Most existing approaches to statistical natural language generation (NLG) use statistics to simplify or refine a hand-crafted generator, for example, by overgenerating to ignore coding agreement rules and then filtering the output using N-gram language models[33]. In contrast, the CamInfo NLG component, has no hand-crafted generation components[34]. Its statistical model is inspired by the Hidden Vector State semantic decoder which represents semantic parse trees as a sequence of stacks[17]. The basic idea is illustrated in Fig. 8. The input system dialogue act a is mapped deterministically to a set of *attribute stacks* S_a. These stacks are then augmented by a set of *filler stacks* S_f and arranged in a sequence S which is in turn mapped to a sequence of phrases constituting the output message $m = m_1 \ldots m_T$. The sequencing of the stacks is determined by two probabilistic models: a stack sequence model $P(S|S_a)$ and a phrase realisation model $P(m|S)$. Hence,

$$\begin{aligned} P(m|a) &= P(m|S_a) \\ &= \sum_{S \in Seq(S_a)} P(m|S)P(S|S_a) \end{aligned} \tag{8}$$

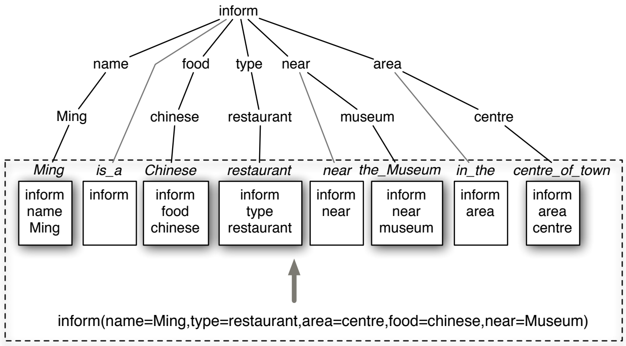

Figure 8: Generation via semantic stacks. The process is illustrated within the dotted box. Each attribute in the dialogue act inform(name= ..) maps deterministically to an *attribute* stack (shown as shadowed boxes). These are augmented by a number of filler stacks to form a stack sequence which maps probabilistically to the surface realisation. The semantic parse above the box illustrates the equivalence of the stack sequence to the semantic parse of the realised utterance.

where $Seq(S_a)$ is the set of all stack sequences which contain all of the attribute stacks S_a. In principle, the distribution $P(m|a)$ could be computed and then sampled. In practice a two step process is used to reduce the computational complexity. Firstly, the sequence S^* is found which maximises (8) and then the sequence of phrases m^* is found which maximises $P(m|S^*)$.

Factored language models are used for both probability models[35]. The stack sequence model is

$$P(S|S_a) = \begin{cases} \prod_{t=1}^{T} P(s_t|s_{t-1}, s_{t-2}, s_\tau) & \text{if } S \in Seq(S_a) \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

where s_τ is denotes the most recent attribute stack. The realisation model is

$$P(m|S) = \prod_{t=1}^{T} P(m_t|m_{t-1}, s_{t-1}, s_t, s_{t+1}) \tag{10}$$

Obtaining training data for the NLG component of a limited domain dialogue system is problematic since there are usually no sources of existing data and collecting data from WoZ-type experiments would be tedious and expensive. An alternative is to use a web-based work distribution system such as Amazon's Mechanical Turk (MTurk)[7]. For CamInfo, the dialogue manager was run with the user simulator used for training to generate a large random set of system dialogue acts. These were then canonicalised and sampled to provided a corpus for annotation. In a two step process, MTurk annotators were firstly asked to provide a simple English utterance matching an abstract description of each dialogue act. In a second step, annotators were asked to align the attributes and values in the dialogue act with the words in the utterance (see [34] for details).

Speech synthesis in the CamInfo system uses a HMM-based approach which is now widely available via the HTS HMM Synthesis Toolkit [36]. Since it is parametric, one of the key benefits of HMM-based synthesis is that it offers the potential to provide speech with intonation, voice quality and

[7]See https://www.mturk.com

Speaker	Utterance	Dialogue act
Sys:	Thank you for calling the Cambridge Information system. How may I help you?	
User:	HI I'M LOOKING FOR AN INDIAN RESTAURANT	hello(=restaurant,food=Indian,task=find)
Sys:	Let me confirm. You are looking for an indian restaurant.	confirm(type=restaurant,food=Indian)
User:	YES	affirm()
Sys:	Blue Spice is a great restaurant and it serves indian food.	inform(name="Blue Spice",type=restaurant,food=Indian)
User:	ERM I WANT SOMETHING CHERRY HINTON	inform(area=cherryhinton)
Sys:	Spice Merchants is a great restaurant in the Cherry Hinton part of town and it serves indian food.	inform(name="Spice Merchants", type=restaurant,area=cherryhinton,food=Indian)
User:	YOU UH WHAT'S THE ADDRESS	request(addr)
Sys:	Spice Merchants is on 60 Trumpington Road.	inform(name="Spice Merchants",addr="60 Trumpington Road")
User:	THE ANYTHING ELSE	reqalts(=dontcare)
Sys:	Sitar Tandoori, which serves indian food, is a great restaurant in the Cherry Hinton part of town.	inform(name="Sitar Tandoori",type=restaurant, area=cherryhinton,food=Indian)
User:	DO YOU HAVE THE ADDRESS	request(addr)
Sys:	Sitar Tandoori is on 43 High Street, Cherry Hinton.	inform(name="Sitar Tandoori",addr="43 High Street,Cherry Hinton")
User:	ARE THERE ANY CHINESE RESTAURANTS	inform(food=Chinese,=restaurant)
Sys:	The Hotpot Restaurant is an outstanding restaurant in the riverside part of town. It serves chinese food.	inform(name="Hotpot Restaurant",type=restaurant, area=riverside,food=Chinese)
User:	NO I WANT CHERRY HINTON	negate(area=cherryhinton)
Sys:	HK Fusion serves chinese food, it's an outstanding restaurant in the Cherry Hinton part of town.	inform(name="HK Fusion",type=restaurant, area=cherryhinton,food=Chinese)
User:	WHAT'S THE ADDRESS	request(addr)
Sys:	HK Fusion is located on 21 Burleigh Street.	inform(name="HK Fusion",addr="21 Burleigh Street")
User:	THE PHONE NUMBER	request(phone)
Sys:	The phone number of HK Fusion is 01223 355909.	inform(name="HK Fusion",phone="01223 355909")
User:	OK THANK YOU GOODBYE	bye()
Sys:	Thank you goodbye	bye()

Figure 9: Example dialogue extracted from CamInfo log files (using the HIS dialogue manager).

affect which varies to match the dialogue context. To facilitate more natural pitch contour generation, the CamInfo synthesiser replaces the standard multi-space distribution HMM with an alternative which assumes that F0 is continuous throughout an utterance but it is generated during unvoiced regions by a globally tied distribution (GTD)[37]. This GTD-based model has the advantage that F0 and its derivatives can be modelled by a single stream and this leads to more accurate trajectories. Furthermore, since the GTD HMM is directly compatible with the HMMs used in HTK-based speech recognisers, all of the tools developed for recognition can be used in synthesis.

As mentioned above, within a limited-domain dialogue system, one of the most interesting features of HMM-based synthesis is the possibility of modifying the intonation to signal discourse information. One obvious example of this is the use of word emphasis to acknowledge a correction. For example, if the user has asked for a cheap, French restaurant and the system has responded with an expensive French restaurant, the user might say "No, I want a cheap French restaurant!". The response of the system should then be "Oh, you want a *cheap* French restaurant." The ability to put emphasis on "cheap" provides important feedback and helps ground that particular user requirement. In HMM-based synthesis, an emphasis capability can be implemented by adding a word emphasis feature to the decision tree used to cluster the context-dependent phones. Unfortunately, if the word emphasis feature is simply added to the existing feature list, the relative rarity of emphasised words in the training data will result in the feature appearing very low down in the tree reducing its effectiveness during synthesis. To overcome this, the decision trees can be factorised and built in multiple passes ensuring that emphasis features are applied first [38]. The result is a system which can indicate the focus of an utterance very effectively by simply setting the word emphasis feature for the phones of the word to be emphasised.

3.6. Evaluation

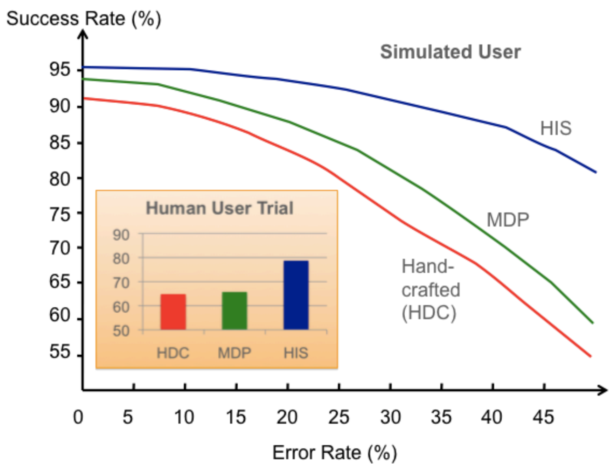

Figure 10: HIS performance as a function of the input error rate. The main graph shows % success rate as a function of semantic error rate using a simulated user. The inset shows the comparable results for a human user trial held in noisy conditions for which the average error rate was 25%.

The CamInfo system described above is the basis of continuous development. Figure 10 shows the performance of the HIS system in 2007 compared to an MDP-based dialogue manager which only maintains a single most likely dialogue state and a version of the HIS system which has multiple states but only a hand-crafted policy. The main graph shows the average success rate as a function of the input semantic error rate using a user simulator, where a dialogue is said to be successful if it returns a venue to the user which satisfies her constraints and any information requested about that venue such as the address or telephone number. As can be seen, the HIS system is substantially more robust at high error rates. The inset summarizes the aggregate results of a user trial conducted in February 2008 in which 36 subjects undertook a variety of tasks in noisy conditions [23]. Again the HIS system is clearly more robust. Since that time a variety of improvements have been made to the system and a further trial will be held in late 2010 as part of the EU Classic Project. To give a more direct impression of the current CamInfo system in operation, Fig. 9 shows a typical dialogue extracted from the CamInfo system log files.

Like the HIS system, the BUDS system has been tested both with a user simulator and in human trials. Performance results are typically similar to the HIS results.

4. Optimisation Issues

The CamInfo system is an example of a large statistical system consisting of a number of components configured ideally to optimise a single global reward function. In practice, each component is typically optimised locally rather than globally and moreover the local objective function may differ from the global one. When the HIS-based CamInfo system was first built, the dialogue manager probability models and the user simulator were hand-crafted. The dialogue policy was optimised using the basic Monte Carlo Control algorithm and it required around 100,000 dialogues to converge. Since that time, various improvements have been made both to model parameter and policy optimisation.

Improved policy optimisation requires better generalisation across states so that when a good action is discovered for some state, other similar states benefit also. In the HIS system, a simple way to achieve this is by smoothing over states in summary space, for example by interpolating Q values over the k-Nearest Neighbours[39]. A more powerful and more general approach is to use function approximation to directly model the value function. Both Kalman filters [40] and Gaussian Processes [41, 42] have been shown to be effective and the number of training cycles required to estimate good policies can be reduced by an order of magnitude or more.

In terms of dialogue manager parameter estimation, a major advantage of the BUDS system is that the network can be extended to include the model parameters themselves along with appropriate Dirichlet priors. If loopy belief propagation is then replaced by expectation propagation [43], the system can learn the model parameters and adapt on-line from data[44].

One limitation of this approach to parameter estimation is that the objective function differs from the policy reward function. More recently it has been demonstrated that the natural gradient methods used to train the BUDS policy can also be used to optimise the BUDS model parameters even though they are not directly differentiable[45]. This is made possible by learning differentiable parameters of their priors in place of trying to learn the parameters directly. This is interesting because as well as providing the ability to optimise the dialogue model parameters wrt to the reward function, the method can be extended in principle to optimise any parameter in the entire dialogue system including non-differentiable parameters.

5. Conclusions

The previous sections have described the CamInfo system including its two interchangeable Dialogue Managers. The system is demonstrably robust to environmental noise. As indicated in section 3.6 and the many further evaluations described in the references, the explicit representation of uncertainty provided by the POMDP framework provides a natural resilience to noise. Furthermore, this resilience requires no developer effort to achieve. In conventional systems, a large part of the design and tuning effort revolves around error recovery dialogues. POMDP-based systems do not require any of this. As soon as a misunderstanding is detected, the probability of the current top hypothesis is reduced and another hypothesis rises to the top. The result is a natural dialogue flow even at high error rates.

Fully statistical systems are also cost effective to develop. The major cost is in data collection but this is an activity which does not require skilled developers. Indeed, we have found that a combination of web-based collection using Amazon's Mechanical Turk and bootstrapping can generate the needed data at modest cost. Furthermore, the architecture is portable. For example, a complete BUDS-based CamInfo system was ported to the CMU Let's Go bus information domain in just a few weeks.

This paper started by recalling the definition of a cognitive user interface. The CamInfo system meets the first two requirements of supporting reasoning and inference, and being able to plan under uncertainty. The remaining two requirements concern adaptation. Work on short-term adaptation is already under way and the recent developments in fast reinforcement learning based on Kalman filtering and Gaussian Processes suggest that this will be possible. Long term adaptation and learning is rather more problematic since in the most general case this will require dynamically modifying the ontology to introduce new concepts, and then learning how these new concepts relate to existing ones. This would require not only on-the-fly modifications to the internal user goal models but consequent updating of the input/output components that reference to them. So whilst progress has been made, there is clearly still plenty more to do.

6. Acknowledgement

The work described in this paper represents the combined efforts of many people. In particular, I would like to acknowledge the contribution of past and present members of the Cambridge Dialogue Systems Group: Milica Gašić, Filip Jurčíček, Simon Keizer, Fabrice Lefevre, François Mairesse, Jost Schatzmann, Konrad Scheffler, Matt Stuttle, Blaise Thomson, Karl Weilhammer, Jason Williams, Hui Ye, and Kai Yu. Some of the research mentioned in this article was funded by the UK EPSRC under grant agreement EP/F013930/1 and by the EU FP7 Programme under grant agreement 216594 (CLASSiC project: `www.classic-project.org`).

7. References

[1] S. Young, "Talking to Machines (Statistically Speaking)," in *Int Conf Spoken Language Processing*, Denver, Colorado, 2002.

[2] J. Williams, P. Poupart, and S. Young, "Factored Partially Observable Markov Decision Processes for Dialogue Management," in *4th Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, Edinburgh, 2005.

[3] M. Ostendorf, A. Kannan, S. Austin, O. Kimball, R. Schwartz, and J. Rohlicek, "Integration of Diverse Recognition Methodologies through Reevaluation of N-Best Sentence Hypotheses," in *Proc DARPA Speech and Natural Language Workshop*, Morgan Kaufmann Publishers, 1991, pp. 83–87.

[4] F. Richardson, M. Ostendorf, and J. Rohlicek, "Lattice-Based Search Strategies for Large Vocabulary Recognition," in *Proc ICASSP*, vol. 1, Detroit, 1995, pp. 576–579.

[5] C. Boidin, V. Rieser, L. van der Plas, O. Lemon, and J. Chevelu, "Predicting how it sounds: Re-ranking dialogue prompts based

on TTS quality for adaptive Spoken Dialogue Systems," in *Interspeech 2009*, Brighton, UK, 2009.

[6] S. Young, "Cognitive User Interfaces," *Signal Processing Magazine*, vol. 27, no. 3, pp. 128–140, 2010.

[7] E. Sondik, "The Optimal Control of Partially Observable Markov Decision Processes," PhD, Stanford University, 1971.

[8] L. Kaelbling, M. Littman, and A. Cassandra, "Planning and Acting in Partially Observable Stochastic Domains," *Artificial Intelligence*, vol. 101, pp. 99–134, 1998.

[9] S. Young, "CUED Standard Dialogue Acts," Cambridge University Engineering Department, Report, 14th October 2007 2007. [Online]. Available: http://mi.eng.cam.ac.uk/research/dialogue/LocalDocs/dastd.pdf

[10] ——, "ATK: An Application Toolkit for HTK," 2005. [Online]. Available: http://mi.eng.cam.ac.uk/research/dialogue/atk_home

[11] S. Young, G. Evermann, M. Gales, T. Hain, D. Kershaw, X. Liu, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland, *The HTK Book Version 3.4*. Cambridge, England: Cambridge University, 2006.

[12] M. Gales and S. Young, "The Application of Hidden Markov Models in Speech Recognition," *Foundations and Trends in Signal Processing*, vol. 1, no. 3, pp. 195–304, 2007.

[13] K. Weilhammer, M. Stuttle, and S. Young, "Bootstrapping Language Models for Dialogue Systems," in *ICSLP 2006*, Pittsburgh, PA, 2006.

[14] R. Pieraccini and E. Levin, "Stochastic Representation of Semantic Structure for Speech Understanding," *Speech Communication*, vol. 11, no. 2, pp. 283–288, 1992.

[15] Y. Wang, L. Deng, and A. Acero, "Spoken Language Understanding An Introduction to the Statistical Framework,," *Signal Processing Magazine*, vol. 22, no. 5, pp. 16–31, 2005.

[16] R. De Mori, F. Bechet, D. Hakkani-Tur, M. McTear, G. Riccardi, and G. Tur, "Spoken language understanding," *IEEE Signal Processing Magazine*, vol. 25, no. 3, pp. 50–58, 2008.

[17] Y. He and S. Young, "Semantic Processing using the Hidden Vector State Model," *Computer Speech and Language*, vol. 19, no. 1, pp. 85–106, 2005.

[18] ——, "Spoken Language Understanding using the Hidden Vector State Model," *Speech Communication*, vol. 48, no. 3-4, pp. 262–275, 2006.

[19] F. Jurcicek, M. Gasic, S. Keizer, F. Mairesse, B. Thomson, K. Yu, and S. Young, "Transformation-based Learning for Semantic Parsing," in *Interspeech 2009*, Brighton, UK, 2009.

[20] F. Mairesse, M. Gasic, F. Jurcicek, S. Keizer, B. Thomson, K. Yu, and S. Young, "Spoken Language Understanding from Unaligned Data using Discriminative Classification Models," in *Int Conf Acoustics Speech and Signal Processing (ICASSP)*, Taipei, Taiwan, 2009.

[21] J. Williams and S. Young, "Partially Observable Markov Decision Processes for Spoken Dialog Systems," *Computer Speech and Language*, vol. 21, no. 2, pp. 393–422, 2007.

[22] S. Keizer, M. Gašić, F. Mairesse, B. Thomson, K. Yu, and S. Young, "Modelling User Behaviour in the HIS-POMDP Dialogue Manager," in *IEEE SLT Workshop*, Goa, India, 2008.

[23] S. Young, M. Gasic, S. Keizer, F. Mairesse, J. Schatzmann, B. Thomson, and K. Yu, "The Hidden Information State Model: a practical framework for POMDP-based spoken dialogue management," *Computer Speech and Language*, vol. 24, no. 2, pp. 150–174, 2010.

[24] M. Gasic and S. Young, "Effective Handling of Dialogue State in HIS POMDP Dialogue Manager for Complex Structure Domains," Cambridge University, Tech. Rep. CUED/F-INFENG/TR.650, 2010.

[25] M. Gasic, F. Lefevre, F. Jurcicek, S. Keizer, F. Mairesse, B. Thomson, K. Yu, and S. Young, "Back-off Action Selection in Summary Space-Based POMDP Dialogue Systems," in *ASRU 2009*, Merano, Italy, 2009.

[26] W. Lovejoy, "Computationally feasible bounds for partially observed Markov decision processes," *Operations Research*, vol. 39, pp. 162–175, 1991.

[27] R. Brafman, "A Heuristic Variable Grid Solution Method for POMDPs," in *AAAI*, Cambridge, MA, 1997.

[28] M. Gašić, S. Keizer, B. Thomson, F. Mairesse, J. Schatzmann, K. Yu, and S. Young, "Training and evaluation of the HIS-POMDP dialogue system in noise," in *Proc. 9th SIGdial*, Columbus, Ohio, 2008.

[29] B. Thomson, J. Schatzmann, and S. Young, "Bayesian Update of Dialogue State for Robust Dialogue Systems," in *Int Conf Acoustics Speech and Signal Processing ICASSP*, Las Vegas, 2008.

[30] C. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

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

[32] J. Peters and S. Schaal, "Natural Actor-Critic," *Neurocomputing*, vol. 71, no. 7-9, pp. 1180–1190, 2008.

[33] I. Langkilde and K. Knight, "Generation that Exploits Corpus-Based Statistical Knowledge," in *Proc 17th Int Conf on Computational Linguistics (COLING-ACL 1998)*, Montreal, Canada, 1998.

[34] F. Mairesse, M. Gasic, F. Jurcicek, S. Keizer, J. Prombonas, B. Thomson, K. Yu, and S. Young, "Phrase-based Statistical Language Generation using Graphical Models and Active Learning," in *ACL 2010*, Uppsala, Sweden, 2010.

[35] J. Bilmes and K. Kirchhoff, "Factored Language Models and Generalised Parallel Backoff," in *HLT/NAACL 2003*, Edmonton, Canada, 2003.

[36] T. Yoshimura, Y. Tokuda, T. Masuko, T. Kobayashi, and T. Kitamura, "Simultaneous modeling of spectrum, pitch and duration in HMM-based speech synthesis," in *Eurospeech 99*, 1999.

[37] K. Yu, T. Toda, M. Gasic, S. Keizer, F. Mairesse, B. Thomson, and S. Young, "Probabilistic Modelling of F0 in Unvoiced Regions in HMM-based Speech Synthesis," in *Int Conf Acoustics Speech and Signal Processing (ICASSP)*, Taipei, Taiwan, 2009.

[38] K. Yu, F. Mairesse, and S. Young, "Word-level Emphasis Modelling in HMM-based Speech Synthesis," in *Proc ICASSP*, Dallas, TX, 2010.

[39] F. Lefevre, M. Gasic, F. Jurcicek, S. Keizer, F. Mairesse, B. Thomson, K. Yu, and S. Young, "k-Nearest Neighbor Monte-Carlo Control Algorithm for POMDP-based Dialogue Systems," in *SigDial 2009*, London, UK, 2009.

[40] M. Geist, O. Pietquin, and G. Fricout, "Kalman temporal differences: the deterministic case," in *IEEE International Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, Nashville, TN, USA, 2009, p. 185192.

[41] E. Engel, S. Mannor, and R. Meir, "Reinforcement learning with Gaussian processes," in *ICML 2005*, Bonn, Germany, 2005.

[42] M. Gasic, F. Jurcicek, S. Keizer, F. Mairesse, B. Thomson, K. Yu, and S. Young, "Gaussian Processes for Fast Policy Optimisation of a POMDP Dialogue Manager for a Real-world Task," in *Submitted to SigDial 2010*, Tokyo, Japan, 2010a.

[43] T. Minka, "Expectation Propagation for Approximate Bayesian Inference," in *Proc 17th Conf in Uncertainty in Artificial Intelligence*. Seattle: Morgan-Kaufmann, 2001, pp. 362–369.

[44] B. Thomson, "Statistical methods for spoken dialogue management," PhD, Cambridge University, 2010.

[45] F. Jurcicek, B. Thomson, S. Keizer, F. Mairesse, M. Gasic, K. Yu, and S. Young, "Natural Belief-Critic: a reinforcement algorithm for parameter estimation in statistical spoken dialogue systems," in *Submitted to Interspeech 2010*, Tokyo, Japan, 2010.