IBM's Belief Tracker: Results On Dialog State Tracking Challenge Datasets

Rudolf Kadlec, Jindřich Libovický, Jan Macek, and Jan Kleindienst
IBM Czech Republic
V Parku 4, Prague 4
Czech Republic
{rudolf_kadlec, jindrich_libovicky, jmacek2, jankle}@cz.ibm.com

Abstract

Accurate dialog state tracking is crucial for the design of an efficient spoken dialog system. Until recently, quantitative comparison of different state tracking methods was difficult. However the 2013 Dialog State Tracking Challenge (DSTC) introduced a common dataset and metrics that allow to evaluate the performance of trackers on a standardized task. In this paper we present our belief tracker based on the Hidden Information State (HIS) model with an adjusted user model component. Further, we report the results of our tracker on test3 dataset from DSTC. Our tracker is competitive with trackers submitted to DSTC, even without training it achieves the best results in L2 metrics and it performs between second and third place in accuracy. After adjusting the tracker using the provided data it outperformed the other submissions also in accuracy and yet improved in L2. Additionally we present preliminary results on another two datasets, test1 and test2, used in the DSTC. Strong performance in L2 metric means that our tracker produces well calibrated hypotheses probabilities.

1 Introduction

Spoken dialog systems need to keep a representation of the dialog state and the user goal to follow an efficient interaction path. The performance of state-of-the-art speech recognition systems varies widely with domain and environment with word accuracy rates ranging from less than 70% to 98%, which often leads to misinterpretation of the user's intention. Dialog state tracking methods need to cope with such error-prone automatic speech recognition (ASR) and spoken language understanding (SLU) outputs. Traditional dialog systems use hand-crafted rules to select from the SLU outputs based on their confidence scores. Recently, several data-driven approaches to dialog state tracking were developed as a part of end-to-end spoken dialog systems. However, specifics of these systems render comparison of dialog state tracking methods difficult.

The Dialog State Tracking Challenge (DSTC) (Williams et al., 2013) provides a shared testbed with datasets and tools for evaluation of dialog state tracking methods. It abstracts from subsystems of end-to-end spoken dialog systems focusing only on the dialog state estimation and tracking. It does so by providing datasets of ASR and SLU outputs with reference transcriptions together with annotation on the level of dialog acts.

In this paper we report initial encouraging results of our generative belief state tracker. We plan to investigate discriminative approaches in the future.

The rest of the paper continues as follows. In the next section we formally introduce the dialog tracking task together with datasets used in the DSTC. Then in Section 3 we discuss related work. Section 4 describes the belief update equations of our tracker. After that we introduce the design of our whole tracking system, especially how we trained the system in a supervised setting on the train dataset and in an unsupervised setting on the test dataset. In Section 6 we show results of our trackers, compare them to other DSTC participants, and discuss the results in the context of design choices and task characteristics.

2 DSTC Problem Definition, Datasets and Metrics

The task of the DSTC can be formally defined as computing $\mathsf{P}(g_t|\mathbf{u}_{0:t}, a_{0:t})$. That is, for each time step t of the dialog compute the probability distribution over the user's hidden goal g given a sequence of SLU hypotheses from the

Dataset	System	#	Annotated
train1a	A	1013	yes
train1b	A	1117	no
train1c	A	9502	no
train2	A	643	yes
train3	B	688	yes
test1	A	715	for eval. only
test2	A	750	for eval. only
test3	B	1020	for eval. only
test4	C	438	for eval. only

Table 1: Datasets description. The *System* column shows what dialog system was used to collect the dataset. The # column shows the number of dialogs in the dataset. The last column informs whether the ground truth annotation was provided with the dataset.

System	Dial. model	SLU scores
A	open	$\langle -\inf, 0 \rangle$
B	fixed	$\langle 0, 1 \rangle$
C	open	$\langle 0, 1 \rangle$

Table 2: Main features of the dialog managers used to collect the datasets. System A and C use open dialog structure where the user can respond with any combination of slots on any machine question. System B uses a fixed dialog structure where the user can respond only with the concept the system expects.

beginning of the dialog up to the time t denoted as $\mathbf{u}_{0:t}$ and a sequence of machine actions $a_{0:t}$. It is assumed that the goal is fixed through the dialog, unless the user is informed that the requested goal does not exist. In DSTC the user's goal consist of nine slots: *route, from.desc, from.neighborhood, from.monument, to.desc, to.neighborhood, to.monument, date, time*.

The dialog datasets in the DSTC are partitioned into five training sets and four test sets. Details and differences of the datasets are summarized in Table 1 and 2. The datasets come from dialog systems deployed by three teams denoted as A, B and C. All the training datasets were transcribed but only three of them were annotated on the level of dialog acts. The SLU confidence scores from system B are relatively well calibrated, meaning that confidences can be directly interpreted as probabilities of observing the SLU hypothesis. Confidence scores from the system A are not well calibrated as noted by several DSTC participants (Lee and Eskenazi, 2013; Kim et al., 2013).

The evaluation protocol is briefly described in Section 6. Its detailed description can be found in (Williams et al., 2012), its evaluation in (Williams et al., 2013).

In 2013, nine teams with 27 trackers participated in the challenge. The results of the best trackers will be discussed together with the results of our tracker later in Section 6.

3 Related Work

This section shortly reviews current approaches to dialog state tracking. We divide the trackers into two broad families of generative and discriminative methods.

3.1 Generative Methods

The HIS model (Young et al., 2010) introduces an approximative method of solving the belief tracking as an inference in a dynamic Bayesian network with SLU hypotheses and machine actions as observed variables and the estimate of the user's goal as a hidden variable. The HIS model was implemented several times (Williams, 2010; Gašić, 2011). Recent criticism of generative methods for belief tracking brought more attention to the discriminative methods (Williams, 2012b).

In the DSTC only few generative system participated. Kim et al. (2013) implemented the HIS model with additional discriminative rescoring, Wang and Lemon (2013) introduced a very simple model based on hand-crafted rules. Both of them scored between the second and the fourth place in the challenge.

3.2 Discriminative Methods

As was previously mentioned, the discriminative methods received more attention recently.

The overall winner of the DSTC (Lee and Eskenazi, 2013) used a maximum entropy model, which they claim to be outperformed by bringing more structure to the model by using the Conditional Random Fields (Lee, 2013). The same type of model is used also by Ren et al. (2013). Usage of Deep Neural Networks was tested by Henderson et al. (2013).

Žilka et al. (2013) compare a discriminative maximum entropy model and a generative method based on approximate inference in a Bayesian network, with the discriminative model preforming better.

4 Model

Our model is an implementation of the HIS model (Young et al., 2010). In HIS the belief state is viewed as a probability distribution over all possible user's goals. The belief state is represented by a set of so-called *partitions*, which are sets of user's goals that are indistinguishable based on actions the system observes. It means the probability mass assigned to a partition spreads to the user's goals in the partition proportionally to their's prior probabilities. The belief update is performed in two steps.

Belief refinement ensures that for each user action on the SLU n-best list and each partition all goals in the partition are either consistent with the user action or not. This step does not change the belief state, it only enables the actual belief update to be computed using the update equation (Eq. 1).

The partitions are organized in a tree structure for which it holds that a child and a parent partition are identical in some slots and complementary in the remaining ones. This is ensured by the belief refinement procedure. For each observed user action and each partition it first checks whether all of the hypotheses in the partition are either consistent with the action or not. If they are not, it splits the partition into two partitions with the parent-child relationship. The inconsistent hypotheses remain in the parent partition and the consistent ones are moved to the child. The belief of the original partition is distributed between the new ones in the ratio of their priors.

To prevent an exponential increase in the number of partitions during the dialog, a partition recombination strategy can be used that removes the less probable partition and moves their hypotheses to different partitions. We perform partition recombination at the end of each turn (Henderson and Lemon, 2008), during the recombination low probability partitions are merged with their parents exactly as suggested by Williams (2010).

For the actual belief update the following standard update equation is used:

$$\mathsf{P}_{t+1}(p) = k \cdot \mathsf{P}_t(p) \cdot \sum_{u \in \mathbf{u}} \mathsf{P}(u|\mathbf{u}) \cdot \mathsf{P}(u|p,a) \quad (1)$$

where k is a normalization constant, $\mathsf{P}_t(p)$ is belief in partition p after turn t, a is the machine action taken in turn t, $\mathbf{u}$ is a set of observed user actions, $\mathsf{P}(u|\mathbf{u})$ is the score of action u in the SLU n-best list $\mathbf{u}$. In this definition $\mathsf{P}_0(p)$ is a prior probability of partition p; the prior might be either uniform or estimated from the training data. The list $\mathbf{u}$ is extended with an unobserved action $\tilde{u}$ whose probability is:

$$\mathsf{P}(\tilde{u}|\mathbf{u}) = 1 - \sum_{u \in \mathbf{u} \setminus \{\tilde{u}\}} \mathsf{P}(u|\mathbf{u}). \quad (2)$$

$\mathsf{P}(u|p,a)$ in the update equation is the *user model*, i.e. how likely the user is to take an action u given that the last machine action was a and user's goal is represented by partition p.

In our case:

$$\mathsf{P}(u|p,a) = \frac{\Lambda(p,u,a)}{\sum_{p' \in \text{partitions}} \Lambda(p',u,a) \cdot \text{size}(p')} \quad (3)$$

where $\text{size}(p)$ is the number of possible user's goals represented by p and $\Lambda(p,u,a)$ is an indicator function that evaluates to 1 when user's action u is compatible with the goal represented by p given the last machine's action was a, otherwise Λ evaluates to 0.

Λ is defined in the following way, for every observed action $u \in \mathbf{u} \setminus \{\tilde{u}\}$:

$$\Lambda(p,u,a) = \Lambda'(p,u,a) \quad (4)$$

where Λ' is a deterministic function that encodes the meanings of user and machine actions for a given partition. The rules expressed by Λ' are for example:

$$\forall a : \Lambda'(p_{s=w}, \mathit{inform}(s=v), a) = \begin{cases} 1 & \text{if } v = w \\ 0 & \text{if } v \neq w \end{cases}$$

and

$$\Lambda'(p_{s=w}, \mathit{yes}(), \mathit{conf}(s=v)) = \begin{cases} 1 & \text{if } v = w \\ 0 & \text{if } v \neq w \end{cases}$$

where $p_{s=w}$ represents a partition where slot s has value w, $\mathit{inform}(s=v)$ is user's action assigning value v to the slot s and $\mathit{conf}(s=v)$ is machine action requiring confirmation that slot s has value v.

For an unobserved action $\tilde{u}$ we define Λ as:

$$\Lambda(p,\tilde{u},a) = \prod_{u \in \mathbf{u} \setminus \{\tilde{u}\}} (1 - \Lambda'(p,u,a)). \quad (5)$$

This definition assumes that user's unobserved action $\tilde{u}$ uniformly supports each partition not supported by any of the observed user's actions u. $\Lambda(p, \tilde{u}, a)$ evaluates to 1 if none of user's actions support given partition, otherwise it evaluates to 0. This can be viewed as an axiom of our system, alternatively we could assume that $\tilde{u}$ supports all partitions, not only those not supported by any observed action.

The key property of the update equations formulated in this way is that the probability of a partition representing a hypothesis that a user's goal was not mentioned in any of the SLU lists up to the time t does not outweigh probability of observed goals even though the prior probability of unobserved hypothesis is usually orders of magnitude higher than the probability of all observed hypotheses. However, when two goals are indistinguishable based on the SLU input then the ratio of their probabilities will be exactly the ratio of their priors.

Belief update equations are generic and independent of the internal structure of partitions. When the tracker has to be adapted to a new dialog domain with the fixed goal the application developer needs to supply only a new definition of Λ' and partition splitting mechanism adjusted according to Λ'.

4.1 Differences to the Original HIS

The key difference between our HIS implementation and previous HIS systems is in the formulation of the user model. Previous HIS-based systems (Young et al., 2010; Gašić, 2011) factorize the user model as:

$$\mathsf{P}^{orig}(u|p,a) = k \cdot \mathsf{P}(\mathcal{T}(u)|\mathcal{T}(a)) \cdot \mathcal{M}(u,p,a)$$

where $\mathsf{P}(\mathcal{T}(u)|\mathcal{T}(a))$ is a dialog act type bigram model and $\mathcal{M}$ is a deterministic item matching model that is similar to our Λ. Based on a description of the item matching model given in (Keizer et al., 2008; Young et al., 2010; Gašić, 2011) we deduce that it evaluates to a constant c_+ instead of 1 when the user action is consistent with the partition and to c_- instead of 0 otherwise. It holds that $0 \leq c_- \ll c_+ \leq 1$, e.g. $c_- = 0.1$ and $c_+ = 0.9$.

In our tracker, we omit the dialog act type model since it is not a mandatory component of the user model and it can be added later. However, the most important systematic difference between our tracker and the original HIS formulation is that instead of using a reduced user model, which would be $\mathsf{P}^{orig}(u|p,a) = \Lambda(p,u,a)$ in the original HIS, we use the formulation given in Eq. 3. The original HIS does not use a concept of partition's size (size(p') in Eq. 3) that we need for the definition of our user model.

Par.	P_t	P^{orig}_{t+1}	P^{ours}_{t+1}
p_a	1/3	1/3	1/4
p_b	1/3	1/3	1/4
p_c	1/3	1/3	1/2

Table 3: Comparison of the effects of original HIS user model and our modified user model. Initially all partitions are equally likely. After performing belief update using Eq. 1 the original model outputs probabilities in the column P^{orig}_{t+1}, the column P^{orig}_{t+1} shows results of our user model.

We will illustrate the difference between these two approaches on a minimalistic abstract example. Suppose the belief space consists of three partitions p_a, p_b and p_c, each of them having probability of 1/3 and representing one possible user's goal (i.e. size$(p_*) = 1$). There are two actions on the SLU list: $u_{a,b}$ that is consistent only with p_a and p_b (i.e. $\Lambda'(p_a, u_{a,b}, *) = 1$), and u_c that is consistent only with p_c. Both $u_{a,b}$ and u_c are equally probable, $\mathsf{P}(u_{a,b}|\mathbf{u}) = \mathsf{P}(u_c|\mathbf{u}) = 1/2$. According to one intuition p_a and p_b should *share* support given to them by action $u_{a,b}$, on the other hand p_c does not share the action u_c with any other partition. Thus after updating the probability using Eq. 1 one would expect $\mathsf{P}_{t+1}(p_c)$ to be higher than $\mathsf{P}_{t+1}(p_a)$. Now we can compare the output of our model and the original HIS side by side as shown in Table 3. The user model as formulated in the original HIS leads to a new belief state where all partitions are equally probable. However, according to our modified user model partition p_c is twice as probable than p_a or p_b. This is, we argue, closer to human intuition.

The update equation for a partition p in this simplistic example is:

$$\mathsf{P}_{t+1}(p) = k \cdot \mathsf{P}(p) \cdot \big(\mathsf{P}(u_{a,b}|\mathbf{u}) \cdot \mathsf{P}(u_{a,b}|p,*) + \mathsf{P}(u_c|\mathbf{u}) \cdot \mathsf{P}(u_c|p,*)\big).$$

For every partition the original model would output the same probability:

$$\mathsf{P}^{orig}_{t+1}(p) = k_1 \, \frac{1}{3}\left(\frac{1}{2} \cdot c_+ + \frac{1}{2} \cdot c_-\right) = \frac{1}{3}$$

However our model gives the following equation for both p_a and p_b:

$$\mathsf{P}^{our}_{t+1}(p_x) = k_2\,\frac{1}{3}\left(\frac{1}{2}\cdot\frac{1}{1+1}+\frac{1}{2}\cdot\frac{0}{1}\right)=\frac{1}{4}$$

where $x \in \{a, b\}$. The impact of $u_{a,b}$ on p_x is divided by a factor of 2 since it is shared by two partitions each representing one possible user goal. For p_c we have:

$$\mathsf{P}^{our}_{t+1}(p_c) = k_2\,\frac{1}{3}\left(\frac{1}{2}\cdot\frac{0}{1+1}+\frac{1}{2}\cdot\frac{1}{1}\right)=\frac{1}{2}.$$

This is how values in Table 3 were computed.

Another extension of the original HIS is how we handle the unobserved action. To our knowledge, the original HIS systems (Young et al., 2010; Gašić, 2011) do not deal with probability of unobserved action; Williams (2010) presents a different way of handling the unobserved action. We provide unified way how to handle unrecognized mass on the SLU list. In the original HIS model, partition p_{unobs} not supported by any of the observed actions obtains probability by $\mathcal{M}$ evaluating to c_- on each observed action. In our model, p_{unobs} receives non-zero probability due to $\Lambda(p_{unobs}, \tilde{u}, *)$ evaluating to 1 (see Eq. 5).

5 Tracker Design and its Variants

The previous section gave detailed description of the update equations of our HIS based tracker. This section presents an overall design of different implemented tracker variants. We will discuss how we use the bus route database and how we perform supervised and unsupervised prior adaptation.

5.1 Single Slot Tracking versus Joint Tracking of Multiple Slots

An advantage of a HIS-based systems is that they make it possible to track a joint probability distribution over a user's goal. This advantage is twofold. First, it enables usage of a joint prior, either learned from training data or from the bus schedule database. Second, tracking a joint distribution makes it possible to use more information from SLU hypotheses. We will illustrate this on an example. Suppose that SLU is able to extract multiple slots from one user's utterance, in our example it might be interpreted as:

```
inform(route=61,to.desc=cmu)   0.5
inform(route=60,to.desc=zoo)   0.4
```

And the machine explicitly confirms the route:

```
expl-confirm(route=61)
```

If the user's response is interpreted as:

```
negate()   0.8
affirm()   0.1
```

Then the system tracking only marginal probabilities over single slots will correctly consider route `60` as being more probable but user's negation will have no effect on marginal distribution of `to.desc`. However, a system tracking the joint distribution will now correctly rank `zoo` higher than `cmu`. The disadvantage of tracking joint hypotheses is that it requires more computational resources. A tracker tracking all slots independently with a uniform prior is denoted as $\text{IBM}^{\textit{indep}}_{\textit{uniform}}$, a tracker tracking joint hypotheses with a uniform prior as $\text{IBM}^{\textit{jointly}}_{\textit{uniform}}$.

5.2 Bus Schedule Database

Along with the dialog dataset DSTC organizers provided a database with bus schedules for routes in Pittsburgh area. We tested possibility to use relation between bus routes and bus stops that can be extracted from the database. First, we normalized bus stop names as found in the SLU hypotheses (e.g. by removing prepositions), in this way we were able to match 98 percent of bus stops found in the SLU to stops in the database.

An initial analysis of the data revealed that only around 55% of $route$, $from.desc$, $to.desc$ hypotheses annotated by human annotators as a ground truth were also found in the database. This means that either callers were often asking for non-existing combinations or the database was mismatched.

Our tracker utilizing the database tracked joint hypotheses for $route$, $from.desc$ and $to.desc$ slots and hypotheses with combinations not found in the database were penalized. The prior of a joint partition $p_{r,f,t}$, for a route r from destination f to destination t, was computed as:

$$\mathsf{P}(p_{r,f,t}) = \mathsf{P}_{uniform}\cdot DB(r,f,t)$$

Where DB is

$$DB(r,f,t) = \begin{cases} 1 & \text{if } \langle r,f,t\rangle \in database \\ \frac{1}{c} & \text{otherwise} \end{cases}$$

where parameter c is a penalty constant for hypotheses not in the database. The value of c is estimated by parameter search on the train data. This tracker will be denoted as $\text{IBM}^{\textit{jointly}}_{\textit{db}}$.

	Test set 3							
	Schedule 2				Schedule 3			
	joint acc.	avg. acc.	joint L2	avg. L2	joint acc.	avg. acc.	joint L2	avg. L2
Team 6 (Lee and Eskenazi, 2013)	**.558**	**.680**	**.801**	.597	**.589**	.823	.779	**.367**
Team 8 (unknown authors)	.424	.616	.845	**.559**	.408	.716	.878	.422
Team 9 (Kim et al., 2013)	.499	.657	.914	.710	.551	**.828**	.928	.461
Team 3 (Žilka et al., 2013)	.464	.645	.831	.669	.528	.794	**.734**	.390
1-best baseline	.448	.620	.865	.611	.492	.703	.839	.514
IBM$^{jointly}_{uniform}$	.521	.654	**.785**	.575	.557	.804	.746	**.344**
IBM$^{indep}_{uniform}$	.521	.654	**.786**	.576	.558	.806	.746	**.343**
IBM$^{jointly}_{db}$	.523	.657	**.774**	.564	.559	.806	.738	**.339**
IBM$^{indep}_{train\text{-}to\text{-}test}$	**.563**	**.680**	**.694**	**.513**	**.609**	**.828**	**.644**	**.285**
IBM$^{indep}_{unsup}$	**.573**	**.689**	**.685**	**.505**	**.611**	**.834**	**.634**	**.279**

Table 4: Results on the DSTC test set 3. Higher accuracy is better, whereas lower L2 score is better. Numbers in bold highlight performance of the best tracker in the selected metric. The first four rows show teams that performed the best in at least one of the selected metrics. For each team in each metric we show performance of the best submitted tracker. This means that numbers in one row do not have to be from a single tracker. It is an upper bound of the team's performance. The fifth row shows performance of a 1-best baseline tracker that always picks the SLU hypothesis with the top confidence. The rest are different variants of our tracker. Here the bold numbers show where our tracker performed better than the best tracker submitted to the DSTC. A light gray highlight of a cell denotes the overall best performance in online setting, a dark gray highlight denotes the best performance while tracking offline.

5.3 Priors Adaptation

We tested two variants of adjusting prior probabilities of user goals. We estimated prior probabilities as a mixture of the uniform probability and empirical distribution estimated on the training data.

In the first experiment the empirical probabilities were estimated using the annotation that was available in the training data. We tracked the slots independently because the empirical joint distribution would be too sparse to generalize on the test data. We used one prior distribution to guide the selection of *route* hypotheses Pr_{route} and one shared distribution for possible destination names Pr_{desc}. This distribution is trained on data from both *from* and *to* destinations thus gaining a more robust estimate compared to using two separate distributions for *from.desc* and *to.desc*. This tracker will be denoted as IBM$^{indep}_{train\text{-}to\text{-}test}$.

In the second experiment we used the test data without the ground truth labels to estimate the empirical prior. We first ran the tracker with the uniform prior on the testing set and we used the output hypotheses as a basis for the empirical distribution. The prior of a hypothesis is proportional to a sum of all tracker output scores for the hypothesis. This scheme is called *unsupervised prior adaptation* by Lee and Eskenazi (2013). Note that the prior was computed on the test dataset. Thus this technique is not directly applicable to a realistic setting where the belief tracker has to produce a belief for each dialog from the test set the first time it sees it. This tracker will be called IBM$^{indep}_{unsup}$.

6 Evaluation

We evaluated all our tracker variants on the DSTC test3 dataset using the protocol designed for the challenge participants. We also present initial results of the basic IBM$^{indep}_{uniform}$ and IBM$^{jointly}_{uniform}$ trackers for test1 and test2 datasets. Several quantities were measured in three different schedules, which defines, which moments of the dialog the evaluation is performed. Here we report results for schedule 2 and 3. Schedule 2 takes into account all turns when the relevant concept appeared on user's SLU list or was mentioned by the dialog system. Schedule 3 evaluates belief at the end of the dialog, i.e. at the moment when the queried information is presented to the user.

We report accuracy, which is the ratio of dialogs where the user goal was correctly estimated, and the L2 score, which is the Euclidean distance of the vector of the resulting belief from a vector having 1 for the correct hypothesis and 0s for the others. For both of these the average values over all tracked slot is reported as well as the value for the joint hypotheses. The accuracy informs us how often the correct query to the database will be made. The L2 score tells us how well-calibrated the results are, which can be important for disambiguation and for statistical policy optimization.

6.1 Method

We used one thousand partitions as the limit for the number of tracked hypotheses. For each tracker ran on the test set 3 we used only the top five SLU hypotheses.

All parameters for mixing the empirical prior probability with uniform distribution in trackers $\text{IBM}^{\textit{indep}}_{\textit{train-to-test}}$ and $\text{IBM}^{\textit{indep}}_{\textit{unsup}}$ were estimated using 3-fold cross validation scheme on the training data. The best parameter setting on the training data was then used in evaluation on the test set.

	Test set 1			
	joint acc.	avg. acc.	joint L2	avg. L2
Team 6	**.364**	**.862**	**.989**	**.278**
Team 9	.225	.789	1.154	.354
Team 2	.206	.777	1.234	.409
1-best baseline	.138	.626	1.220	.530
$\text{IBM}^{\textit{jointly}}_{\textit{uniform}}$	.332	.813	.992	.282
$\text{IBM}^{\textit{indep}}_{\textit{uniform}}$	.331	.804	1.010	.304

Table 5: Preliminary results for schedule 3 on the DSTC test set 1 of our two trackers compared to three overall well performing teams. For teams 6 and 9 see Table 4, team 2 is (Wang and Lemon, 2013). The legend of the table is the same as in Table 4.

Even though we concentrated mainly on testing the tracker on dataset 3, we also ran it on the datasets 1 and 2. For the datasets 1 and 2 we used the single best SLU hypothesis from the live system. Such hypothesis was assigned 99% probability and the remaining 1% was left for the unobserved action. For the datasets 1 and 2 a post hoc computed SLU hypotheses are available in addition to the live data. In our experiments, using the post hoc computed SLU hypotheses with normalized confidence scores yielded worse results for our tracking systems.

	Test set 2			
	joint acc.	avg. acc.	joint L2	avg. L2
Team 6	**.526**	**.854**	**.885**	**.311**
Team 9	.268	.748	1.098	.450
Team 2	.320	.764	1.148	.470
1-best baseline	.141	.487	1.185	.648
$\text{IBM}^{\textit{jointly}}_{\textit{uniform}}$	.431	.789	**.846**	.316
$\text{IBM}^{\textit{indep}}_{\textit{uniform}}$	.413	.778	**.875**	.332

Table 6: Preliminary results for schedule 3 on the DSTC test set 2. For teams see Tables 4 and 5. The legend of the table is the same as in Table 4.

6.2 Results

Results of our trackers on the DSTC dataset 3 are summarized in Table 4. Preliminary results of the trackers on datasets 1 and 2 whose confidence scores are not that well calibrated are shown in Tables 5 and 6. The running time of the trackers was on average below 0.05 seconds per turn[1]. The only exception is $\text{IBM}^{\textit{jointly}}_{\textit{db}}$ that executes plenty of database queries. Although we did not focus on the computational performance optimization most of the trackers are suitable for on-line use.

6.3 Discussion

Quantitative Comparison to DSTC Trackers. First let us discuss results of our trackers on test 3 (Table 4). Here both basic variants of the tracker $\text{IBM}^{\textit{indep}}_{\textit{uniform}}$ and $\text{IBM}^{\textit{jointly}}_{\textit{uniform}}$ perform almost identically. This is because test 3 uses fixed dialog flow as discussed in Section 2, minor differences in performance between $\text{IBM}^{\textit{indep}}_{\textit{uniform}}$ and $\text{IBM}^{\textit{jointly}}_{\textit{uniform}}$ are caused only by numerical issues. The trackers are around the third place in accuracy. In joint L2 metrics they outperform the best tracker in DSTC submitted by Team 6 (Lee and Eskenazi, 2013).

Tracker utilizing database $\text{IBM}^{\textit{jointly}}_{\textit{db}}$ does not show any significant improvement over the same tracker without database-based prior $\text{IBM}^{\textit{jointly}}_{\textit{uniform}}$. We hypothesize that this is because of the fact that people frequently asked for non-existing combinations of routes and stops, which were penalized for not being in the database, as discussed in Sec. 5.2.

Next follow the results of tracker $\text{IBM}^{\textit{indep}}_{\textit{train-to-test}}$ that learns priors for single slots on training dataset and uses them while inferring user's goal on the test set. In test set 3 priors enhanced

[1]On one core of Intel Xeon CPU E3-1230 V2, 3.30GHz, with memory limitation of 1GB.

tracker's performance in all metrics and the tracker outperformed all DSTC trackers.

Interesting results were achieved by $\text{IBM}^{\textit{indep}}_{\textit{unsup}}$ that performed even better than the $\text{IBM}^{\textit{indep}}_{\textit{train-to-test}}$. It uses a prior trained on the test set by running the tracker with a uniform prior. The tracker was run for three iterations each time using output of the previous iteration as a new prior.

After running the experiments with the top 5 SLU hypotheses, we performed an experiment that investigated influence of n-best list length on the tracker's accuracy. We evaluated five system variants that received 1, 2, 3, 4 and 5 best SLU hypotheses. The overall trend was that initially performance increased as more SLU hypotheses were provided however then performance started decreasing. The 3-best variant achieved about 1.5% increase in joint accuracy compared to the 1-best. However, when using more than 3 best hypotheses, the performance slightly decreased. For instance, $\text{IBM}^{\textit{indep}}_{\textit{uniform}}$ using 1-best hypothesis performed comparable to the 5-best configuration. Similar behavior of generative systems assuming observation independence has already been observed in different domains (Vail et al., 2007).

Based on these results we deduce two conclusions. First, strong performance of $\text{IBM}^{\textit{indep}}_{\textit{uniform}}$ 1-best system compared to the 1-best baseline system suggests that the main added value of our tracker in this domain is in the aggregation of observations from multiple time steps, not in tracking multiple hypotheses from one turn. Second, we attribute the effect of decreasing accuracy to the correlation of ASR errors from consecutive dialog turns. As noted by Williams (2012b), correlated ASR errors violate the assumption of observation independence that is assumed by HIS. Extending the user model with an auto-regressive component, that is with dependence on observations from the previous time step (i.e. $\mathsf{P}(u_t|\mathbf{u_{t-1}}, p, a)$), might help to tackle this problem in generative models (Wellekens, 1987).

To summarize the results on test set 3, even without any prior adaptation on the data our tracker is competitive with the best submissions to DSTC. After incorporating prior knowledge it outperforms all submitted trackers.

On test set 1 and test set 2 (see Tables 5 and 6) the trackers perform second in accuracy. In L2 metrics the trackers are competitive with the best tracker in DSTC submitted by Team 6 and they outperform it in one out of four cases. It is interesting that our basic strategy that ignores live SLU scores performed that strong.

However, on test 1 and test 2, which make it possible to input multiple slots in one user utterance, $\text{IBM}^{\textit{jointly}}_{\textit{uniform}}$ outperforms $\text{IBM}^{\textit{indep}}_{\textit{uniform}}$, both in accuracy and L2. We hypothesize that this is because of effect of tracking joint distributions described in Section 5.1.

Qualitative Comparison to DSTC Trackers. Compared to another HIS-based system (Kim et al., 2013) participating in the DSTC, our implementation does not suffer from the problem of assigning high probability to the hypothesis that the user goal was not observed so far. This might be due to our modified user model. Therefore our implementation does not need a final transformation of belief scores as reported by Kim et al. (2013).

Additionally, our implementation does not exhibit the *forgetting* behavior as experienced by Žilka et al. (2013). Forgetting is undesirable given the validity of assumption that the user's goal remains fixed in the whole dialog, which is the case of DSTC bus schedule domains.

7 Conclusion

Although the use of generative trackers was recently criticized by Williams (2012a), our results show that at least in some metrics (e.g. L2 metrics on dataset 3) a generative tracker can outperform the best state-of-the-art discriminative tracker (Lee and Eskenazi, 2013). Even though we agree that the discriminative approach might be more promising, it seems that in general there are conditions where generative models learn faster than discriminative models (Ng and Jordan, 2001). Thus it might be beneficial to use a generative tracker for a newly deployed dialog system with only a few training dialogs available and switch to a discriminative model once enough training data from an already running system is collected. Ensemble trackers incorporating both generative and discriminative models as used by Lee and Eskenazi (2013) might also be an interesting direction for future research.

Acknowledgment

We would like to thank Jiří Havelka for his valuable comments on a draft of this paper. This work was partially funded by the GetHomeSafe project (EU 7th Framework STREP project No. 288667).

References

Milica Gašić. 2011. *Statistical Dialogue Modelling*. PhD thesis, University of Cambridge.

James Henderson and Oliver Lemon. 2008. Mixture Model POMDPs for Efficient Handling of Uncertainty in Dialogue Management. In *Proc ACL-HLT*, pages 73–76.

Matthew Henderson, Blaise Thomson, and Steve Young. 2013. Deep neural network approach for the dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*, pages 467–471, Metz, France, August. Association for Computational Linguistics.

Simon Keizer, Milica Gašić, François Mairesse, Blaise Thomson, Kai Yu, and Steve Young. 2008. Modelling user behaviour in the his-pomdp dialogue manager. In *Spoken Language Technology Workshop, 2008. SLT 2008. IEEE*, pages 121–124. IEEE.

Daejoong Kim, Jaedeug Choi Choi, Kee-Eung Kim, Jungsu Lee, and Jinho Sohn. 2013. Engineering statistical dialog state trackers: A case study on dstc. In *Proceedings of the SIGDIAL 2013 Conference*, pages 462–466, Metz, France, August. Association for Computational Linguistics.

Sungjin Lee and Maxine Eskenazi. 2013. Recipe for building robust spoken dialog state trackers: Dialog state tracking challenge system description. In *Proceedings of the SIGDIAL 2013 Conference*, pages 414–422, Metz, France, August. Association for Computational Linguistics.

Sungjin Lee. 2013. Structured Discriminative Model For Dialog State Tracking. In *Proceedings of the SIGDIAL 2013 Conference*, pages 442–451, Metz, France, August. Association for Computational Linguistics.

Andrew Ng and Michael Jordan. 2001. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Neural Information Processing Systems*, pages 841–848.

Hang Ren, Weiqun Xu, Yan Zhang, and Yonghong Yan. 2013. Dialog state tracking using conditional random fields. In *Proceedings of the SIGDIAL 2013 Conference*, pages 457–461, Metz, France, August. Association for Computational Linguistics.

Douglas L Vail, Manuela M Veloso, and John D Lafferty. 2007. Conditional Random Fields for Activity Recognition Categories and Subject Descriptors. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent systems (AAMAS 2007)*.

Zhuoran Wang and Oliver Lemon. 2013. A simple and generic belief tracking mechanism for the dialog state tracking challenge: On the believability of observed information. In *Proceedings of the SIGDIAL 2013 Conference*, pages 423–432, Metz, France, August. Association for Computational Linguistics.

Christian Wellekens. 1987. Explicit time correlation in hidden markov models for speech recognition. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'87.*, volume 12, pages 384–386. IEEE.

Jason D Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. 2012. Dialog state tracking challenge handbook.

Jason Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. 2013. The Dialog State Tracking Challenge. In *Proceedings of the SIGDIAL 2013 Conference*, pages 404–413, Metz, France, August. Association for Computational Linguistics.

Jason D. Williams. 2010. Incremental partition recombination for efficient tracking of multiple dialog states. In *ICASSP*, pages 5382–5385.

Jason D. Williams. 2012a. Challenges and Opportunities for State Tracking in Statistical Spoken Dialog Systems: Results From Two Public Deployments. *IEEE Journal of Selected Topics in Signal Processing*, 6(8):959–970, December.

Jason D Williams. 2012b. A critical analysis of two statistical spoken dialog systems in public use. In *Spoken Language Technology Workshop (SLT), 2012 IEEE*, pages 55–60. IEEE.

Steve Young, Milica Gašić, Simon Keizer, François Mairesse, Jost Schatzmann, Blaise Thomson, and Kai Yu. 2010. The Hidden Information State model: A practical framework for POMDP-based spoken dialogue management. *Computer Speech & Language*, 24(2):150–174, April.

Lukáš Žilka, David Marek, Matěj Korvas, and Filip Jurčíček. 2013. Comparison of bayesian discriminative and generative models for dialogue state tracking. In *Proceedings of the SIGDIAL 2013 Conference*, pages 452–456, Metz, France, August. Association for Computational Linguistics.