Exploiting Machine-Transcribed Dialog Corpus to Improve Multiple Dialog States Tracking Methods

Sungjin Lee[1,2] and **Maxine Eskenazi**[1]
[1]Language Technologies Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania
[2]Computer Science and Engineering, Pohang University of Science and Technology, South Korea
{sungjin.lee, max}@cs.cmu.edu[1], junion@postech.ac.kr[2]

Abstract

This paper proposes the use of unsupervised approaches to improve components of partition-based belief tracking systems. The proposed method adopts a dynamic Bayesian network to learn the user action model directly from a machine-transcribed dialog corpus. It also addresses confidence score calibration to improve the observation model in a unsupervised manner using dialog-level grounding information. To verify the effectiveness of the proposed method, we applied it to the Let's Go domain (Raux et al., 2005). Overall system performance for several comparative models were measured. The results show that the proposed method can learn an effective user action model without human intervention. In addition, the calibrated confidence score was verified by demonstrating the positive influence on the user action model learning process and on overall system performance.

1 Introduction

With present Automatic Speech Recognition (ASR) and Spoken Language Understanding (SLU) errors, it is impossible to directly observe the true user goal and action. It is crucial, therefore, to efficiently infer this true state from erroneous observations over multiple dialog turns. The Partially Observable Markov Decision Process (POMDP) framework has offered a well-founded theory for this purpose (Henderson et al., 2008; Thomson and Young, 2010a; Williams and Young, 2007; Young et al., 2010). Several approximate methods have also emerged to tackle the vast complexity of representing and maintaining belief states, e.g., partition-based approaches (Gasic and Young, 2011; Williams, 2010; Young et al., 2010) and Bayesian network (BN)-based methods (Raux and Ma, 2011; Thomson and Young, 2010a). The partition-based approaches attempt to group user goals into a small number of partitions and split a partition only when a distinction is required by observations. This property endows it with the high scalability that is suitable for fairly complex domains. However, the parameter learning procedures for the partition-based methods is still limited to hand-crafting or the use of a simple maximum likelihood estimation (Keizer et al., 2008; Roy et al., 2000; Thomson and Young, 2010a; Williams, 2008). In contrast, several unsupervised methods which do not require human transcription and annotation have been recently proposed to learn BN-based models (Jurcicek et al., 2010; Syed and Williams, 2008; Thomson et al., 2010b). In this paper we describe an unsupervised process that can be applied to the partition-based methods. We adopt a dynamic Bayesian network to learn the user action model which defines the likelihood of user actions for a given context. In addition, we propose a simple confidence score calibration method to improve the observation model which represents the probability of an observation given the true user action.

This paper is structured as follows. Section 2 describes previous research and the novelty of our approach. Section 3 and Section 4 elaborate on our proposed unsupervised approach. Section 5 explains the experimental setup. Section 6 presents and discusses the results. Finally, Section 7 concludes with a brief summary and suggestions for future research.

2 Background and Related Work

In order to reduce the complexity of the belief states over the POMDP states, the following factorization of the belief state has been commonly applied to the belief update procedure (Williams et al., 2005):

$$\begin{aligned} & b(\mathbf{g}_t, \mathbf{u}_t, \mathbf{h}_t) \\ & \propto \underbrace{p(\mathbf{o}_t|\mathbf{u}_t)}_{\text{observation model}} \sum_{\mathbf{h}_{t-1}} \underbrace{p(\mathbf{h}_t|\mathbf{h}_{t-1}, \mathbf{u}_t, \mathbf{s}_t)}_{\text{dialog history model}} \\ & \quad \underbrace{p(\mathbf{u}_t|\mathbf{g}_t, \mathbf{s}_t, \mathbf{h}_{t-1})}_{\text{user action model}} \sum_{\mathbf{g}_{t-1}} \underbrace{p(\mathbf{g}_t|\mathbf{g}_{t-1}, \mathbf{s}_{t-1})}_{\text{user goal model}} \\ & \quad \sum_{\mathbf{u}_{t-1}} b(\mathbf{g}_{t-1}, \mathbf{u}_{t-1}, \mathbf{h}_{t-1}) \end{aligned} \tag{1}$$

where $\mathbf{g}_t, \mathbf{s}_t, \mathbf{u}_t, \mathbf{h}_t, \mathbf{o}_t$ represents the user goal, the system action, the user action, the dialog history, and the observed user action for each time slice, respectively. The user goal model describes how the user goal evolves. In the partition-based approaches, this model is further approximated by assuming that the user does not change their mind during the dialog (Young et al., 2010):

$$\sum_{\mathbf{g}_{t-1}} p(\mathbf{g}_t|\mathbf{g}_{t-1}, \mathbf{s}_{t-1}) = p(\mathbf{p}_t|\mathbf{p}_{t-1}) \tag{2}$$

where $\mathbf{p}_t$ is a partition from the current turn. The dialog history model indicates how the dialog history changes and can be set deterministically by simple discourse rules, for example:

$$\begin{aligned} & p(\mathbf{h}_t = \textit{Informed}|\mathbf{h}_{t-1}, \mathbf{u}_t, \mathbf{s}_t) = \\ & \quad \begin{cases} 1 & \text{if } \mathbf{h}_{t-1} = \textit{Informed} \text{ or } \mathbf{u}_t = \textit{Inform}(\cdot), \\ 0 & \text{otherwise.} \end{cases} \end{aligned} \tag{3}$$

The user action model defines how likely user actions are. By employing partitions, this can be approximated by the bigram model of system and user action at the predicate level, and the matching function (Keizer et al., 2008):

$$\begin{aligned} & p(\mathbf{u}_t|\mathbf{g}_t, \mathbf{s}_t, \mathbf{h}_{t-1}) \\ & \quad \propto p(\mathcal{T}(\mathbf{u}_t)|\mathcal{T}(\mathbf{s}_t)) \cdot \mathcal{M}(\mathbf{u}_t, \mathbf{p}_t, \mathbf{s}_t) \end{aligned} \tag{4}$$

where $\mathcal{T}(\cdot)$ denotes the predicate of the action and $\mathcal{M}(\cdot)$ indicates whether or not the user action matches the partition and system action. However, it turned out that the bigram user action model did not provide an additional gain over the improvement achieved by the matching function according to (Keizer et al., 2008). This might indicate that it is necessary to incorporate more historical information. To make use of historical information in an unsupervised manner, the *Expectation Maximization* algorithm was adopted to obtain maximum likelihood estimates (Syed and Williams, 2008). But these methods still require a small amount of transcribed data to learn the observation confusability, and they suffer from overfitting as a general property of maximum likelihood. To address this problem, we propose a Bayesian learning method, which requires no transcribed data.

The observation model represents the probability of an observation given the true user action. The observation model is usually approximated with the confidence score computed from the ASR and SLU results:

$$p(\mathbf{o}_t|\mathbf{u}_t) \approx p(\mathbf{u}_t|\mathbf{o}_t) \tag{5}$$

It is therefore of vital importance that we obtain the most accurate confidence score as possible. We propose an efficient method that can improve the confidence score by calibrating it using grounding information.

3 User Action Model

To learn the user action model, a dynamic Bayesian network is adopted with several conditional independence assumptions similar to Equation 1. This gives rise to the graphical structure shown in Figure 1. As mentioned in Section 2, the user action model deals with actions at the predicate level[1]. This abstract-level handling enables the user action model to employ exact inference algorithms such as the *junction tree* algorithm (Lauritzen and Spiegelhalter, 1988) for more efficient reasoning over the graphical structure.

[1]To keep the notation uncluttered, we will omit $\mathcal{T}(\cdot)$.

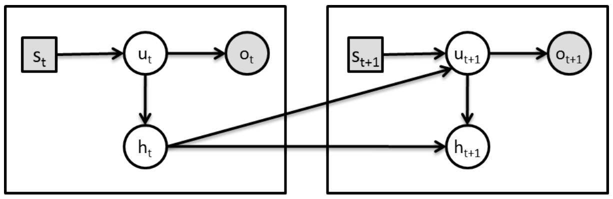

Figure 1: The graphical structure of the dynamic Bayesian network for the user action model. The shaded items are observable and the transparent ones are latent.

The joint distribution for this model is given by

$$\begin{aligned} &p(\mathbf{S},\mathbf{H},\mathbf{U},\mathbf{O}|\boldsymbol{\Theta}) \\ &\quad = p(\mathbf{h}_0|\boldsymbol{\pi})\prod_t p(\mathbf{u}_t|\mathbf{s}_t,\mathbf{h}_{t-1},\boldsymbol{\phi}) \\ &\quad\quad \cdot p(\mathbf{h}_t|\mathbf{h}_{t-1},\mathbf{u}_t,\boldsymbol{\eta})p(\mathbf{o}_t|\mathbf{u}_t,\boldsymbol{\zeta}) \end{aligned} \tag{6}$$

where a capital letter stands for the set of corresponding random variables, e.g., $\mathbf{U} = \{\mathbf{u}_1,\ldots,\mathbf{u}_N\}$, and $\boldsymbol{\Theta} = \{\boldsymbol{\pi},\boldsymbol{\phi},\boldsymbol{\eta},\boldsymbol{\zeta}\}$ denotes the set of parameters governing the model[2].

Unlike previous research which learns $\boldsymbol{\zeta}$ using maximum likelihood estimation, we use a deterministic function that yields a fraction of an observed confidence score in accordance with the degree of agreement between $\mathbf{u}_t$ and $\mathbf{o}_t$:

$$p(\mathbf{o}_t|\mathbf{u}_t) = CS(\mathbf{o}_t)\cdot\left(\frac{|\mathbf{o}_t\cap\mathbf{u}_t|}{|\mathbf{o}_t\cup\mathbf{u}_t|}\right)+\epsilon \tag{7}$$

where $CS(\cdot)$ returns the confidence score of the associated observation. As mentioned above, $\boldsymbol{\pi}$ and $\boldsymbol{\eta}$ are deterministically set by simple discourse rules (Equation 3). This only leaves the user action model $\boldsymbol{\phi}$ to be learned. In a Bayesian model, any unknown parameter is given a prior distribution and is absorbed into the set of latent variables, thus it is not feasible to directly evaluate the posterior distribution of the latent variables and the expectations with respect to this distribution. Therefore a deterministic approximation, called *mean field* theory (Parisi, 1988), is applied.

In *mean field* theory, the family of posterior distributions of the latent variables is assumed to be partitioned into disjoint groups:

$$q(\mathbf{Z}) = \prod_{i=1}^{M} q_i(\mathbf{Z}_i) \tag{8}$$

where $\mathbf{Z} = \{\mathbf{z}_1,\ldots,\mathbf{z}_N\}$ denotes all latent variables including parameters and $\mathbf{Z}_i$ is a disjoint group. Amongst all distributions $q(\mathbf{Z})$ having the form of Equation 8, we then seek the member of this family for which the divergence from the true posterior distribution is minimized. To achieve this, the following optimization with respect to each of the $q_i(\mathbf{Z}_i)$ factors is to be performed in turn (Bishop, 2006):

$$\ln q_j^*(\mathbf{Z}_j) = E_{i\neq j}\big[\ln(\mathbf{X},\mathbf{Z})\big] + const \tag{9}$$

where $\mathbf{X} = \{\mathbf{x}_1,\ldots,\mathbf{x}_N\}$ denotes all observed variables and $E_{i\neq j}$ means an expectation with respect to the q distributions over all groups $\mathbf{Z}_i$ for $i \neq j$.

Now we apply the *mean field* theory to the user model. Before doing so, we need to introduce the prior over the parameter $\boldsymbol{\phi}$ which is a product of *Dirichlet* distributions[3].

$$\begin{aligned} p(\boldsymbol{\phi}) &= \prod_{\mathbf{k}} Dir(\boldsymbol{\phi}_{\mathbf{k}}|\boldsymbol{\alpha}^0_{\mathbf{k}}) \\ &= \prod_{\mathbf{k}} C(\boldsymbol{\alpha}^0_{\mathbf{k}})\prod_l \phi_{\mathbf{k},l}^{\alpha^0_{\mathbf{k}}-1} \end{aligned} \tag{10}$$

where $\mathbf{k}$ represents the joint configuration of all of the parents and $C(\boldsymbol{\alpha}^0_{\mathbf{k}})$ is the normalization constant for the *Dirichlet* distribution. Note that for symmetry we have chosen the same parameter $\alpha^0_{\mathbf{k}}$ for each of the components.

Next we approximate the posterior distribution, $q(\mathbf{H},\mathbf{U},\boldsymbol{\phi})$ using a factorized form, $q(\mathbf{H},\mathbf{U})q(\boldsymbol{\phi})$. Then we first apply Equation 9 to find an expression for the optimal factor $q^*(\boldsymbol{\phi})$:

[2]Here, a uniform prior distribution is assigned on $\mathbf{S}$

[3]Note that priors over parameters for deterministic distributions (e.i., $\boldsymbol{\pi}$, $\boldsymbol{\eta}$,and $\boldsymbol{\zeta}$) are not necessary.

$$\begin{aligned}\ln q^*(\boldsymbol{\phi}) &= E_{\mathbf{H},\mathbf{U}}\big[\ln p(\mathbf{S},\mathbf{H},\mathbf{U},\mathbf{O},\boldsymbol{\Theta})\big] + \mathit{const} \\ &= E_{\mathbf{H},\mathbf{U}}\Big[\sum_t \ln p(\mathbf{u}_t|\mathbf{s}_t,\mathbf{h}_{t-1},\boldsymbol{\phi})\Big] \\ &\quad + \ln p(\boldsymbol{\phi}) + \mathit{const} \\ &= \sum_t \sum_{i,j,k}\Big(E_{\mathbf{H},\mathbf{U}}\big[\boldsymbol{\delta}_{i,j,k}\big]\ln \phi_{i,j,k}\Big) \\ &\quad + \sum_{i,j,k}(\alpha^o_{i,j,k}-1)\ln\phi_{i,j,k} + \mathit{const} \\ &= \sum_{i,j,k}\Big(\Big(E_{\mathbf{H},\mathbf{U}}[n_{i,j,k}] + (\alpha^o_{i,j,k}-1)\Big) \\ &\quad \cdot \ln\phi_{i,j,k}\Big) + \mathit{const}\end{aligned} \tag{11}$$

where $\delta(\cdot,\cdot)$ denotes *Kronecker* delta and $\boldsymbol{\delta}_{i,j,k}$ denotes $\delta(\mathbf{s}_t,i)\delta(\mathbf{h}_{t-1},j)\ \delta(\mathbf{u}_t,k)$. $n_{i,j,k}$ is the number of times where , $\mathbf{s}_t = i, \mathbf{h}_{t-1} = j$, and $\mathbf{u}_t = k$. This leads to a product of *Dirichlet* distributions by taking the exponential of both sides of the equation:

$$\begin{aligned} q^*(\boldsymbol{\phi}) &= \prod_{i,j} Dir(\boldsymbol{\phi}_{i,j}|\boldsymbol{\alpha}_{i,j}), \\ \alpha_{i,j,k} &= \alpha^0_{i,j,k} + E_{\mathbf{H},\mathbf{U}}[n_{i,j,k}]\end{aligned} \tag{12}$$

To evaluate the quantity $E_{\mathbf{H},\mathbf{U}}[n_{i,j,k}]$, Equation 9 needs to be applied once again to obtain an optimal approximation of the posterior distribution $q^*(\mathbf{H},\mathbf{U})$.

$$\begin{aligned}\ln q^*(\mathbf{H},\mathbf{U}) &= E_{\boldsymbol{\phi}}\big[\ln p(\mathbf{S},\mathbf{H},\mathbf{U},\mathbf{O},\boldsymbol{\Theta})\big] + \mathit{const} \\ &= E_{\boldsymbol{\phi}}\Big[\sum_t \ln p(\mathbf{u}_t|\mathbf{s}_t,\mathbf{h}_{t-1},\boldsymbol{\phi}) \\ &\quad + \ln p(\mathbf{h}_t|\mathbf{h}_{t-1},\mathbf{u}_t) \\ &\quad + \ln p(\mathbf{o}_t|\mathbf{u}_t)\Big] + \mathit{const} \\ &= \sum_t \Big(E_{\boldsymbol{\phi}}\big[\ln p(\mathbf{u}_t|\mathbf{s}_t,\mathbf{h}_{t-1},\boldsymbol{\phi})\big] \\ &\quad + \ln p(\mathbf{h}_t|\mathbf{h}_{t-1},\mathbf{u}_t) \\ &\quad + \ln p(\mathbf{o}_t|\mathbf{u}_t)\Big) + \mathit{const}\end{aligned} \tag{13}$$

where $E_{\boldsymbol{\phi}}\big[\ln p(\mathbf{u}_t|\mathbf{s}_t,\mathbf{h}_{t-1},\boldsymbol{\phi})\big]$ can be obtained using Equation 12 and properties of the *Dirichlet* distribution:

$$\begin{aligned} & E_{\boldsymbol{\phi}}\big[\ln p(\mathbf{u}_t|\mathbf{s}_t,\mathbf{h}_{t-1},\boldsymbol{\phi})\big] \\ &\quad = \sum_{i,j,k}\boldsymbol{\delta}_{i,j,k}E_{\boldsymbol{\phi}}\big[\ln\phi_{i,j,k}\big] \\ &\quad = \sum_{i,j,k}\boldsymbol{\delta}_{i,j,k}(\psi(\alpha_{i,j,k}) - \psi(\hat{\boldsymbol{\alpha}}_{i,j}))\end{aligned} \tag{14}$$

where $\psi(\cdot)$ is the digamma function with $\hat{\boldsymbol{\alpha}}_{i,j} = \sum_k \alpha_{i,j,k}$. Because computing $E_{\mathbf{H},\mathbf{U}}[n_{i,j,k}]$ is equivalent to summing each of the marginal posterior probabilities $q^*(\mathbf{h}_{t-1},\mathbf{u}_t)$ with the same configuration of conditioning variables, this can be done efficiently by using the *junction tree* algorithm. Note that the expression on the right-hand side for both $q^*(\boldsymbol{\phi})$ and $q^*(\mathbf{H},\mathbf{U})$ depends on expectations computed with respect to the other factors. We will therefore seek a consistent solution by cycling through the factors and replacing each in turn with a revised estimate.

4 Confidence Score Calibration

As shown in Section 2, we can obtain a better observation model by improving confidence score accuracy. Since the confidence score is usually computed using the ASR and SLU results, it can be enhanced by adding dialog-level information. Basically, the confidence score represents how likely it is that the recognized input is correct. This means that a well-calibrated confidence score should satisfy that property such that:

$$p(\mathbf{u}_t = a|\mathbf{o}_t = a) \simeq \frac{\sum_k \delta(\mathbf{u}_k,a)\delta(\mathbf{o}_k,a)}{\sum_k \delta(\mathbf{o}_k,a)} \tag{15}$$

However, the empirical distribution on the right side of this equation often does not well match the confidence score measure on the left side. If a large corpus with highly accurate annotation was used, a straightforward remedy for this problem would be to construct a mapping function from the given confidence score measure to the empirical distribution. This leads us to propose an unsupervised method that estimates the empirical distribution and constructs the mapping function which is fast enough to run in real time. Note that we will not construct

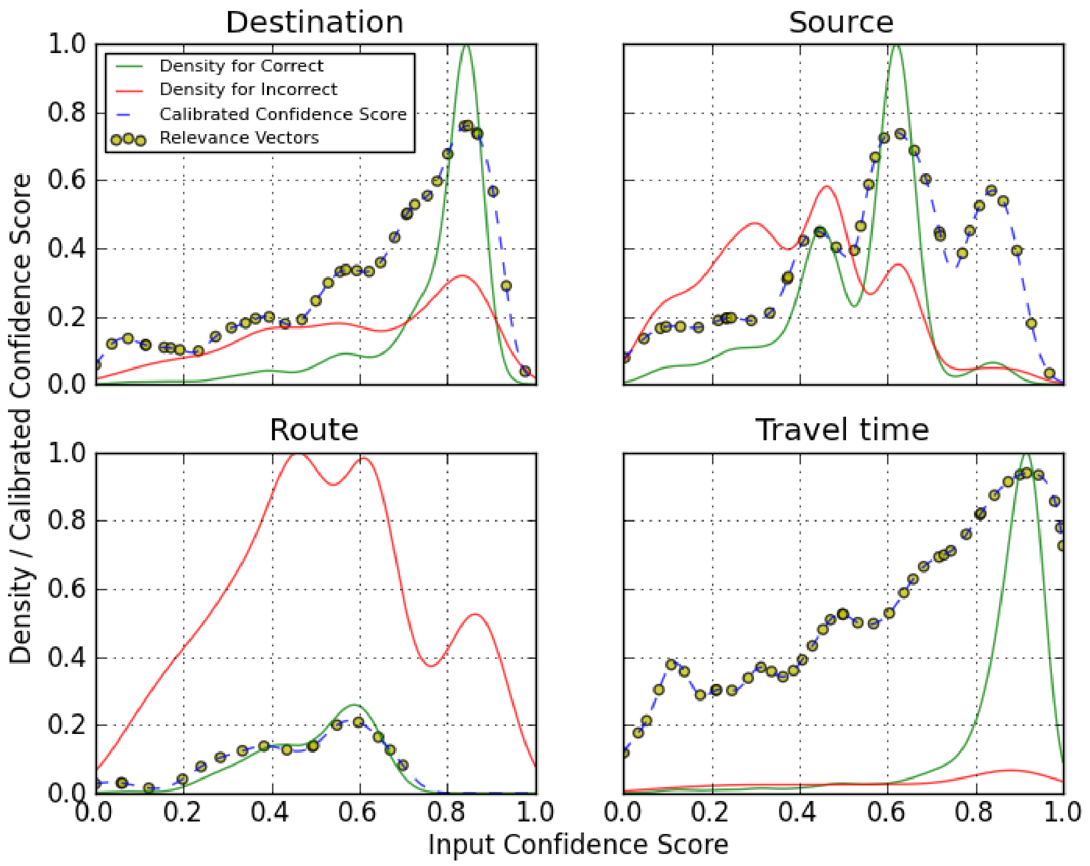

Figure 2: Illustrations of confidence score calibration for the representative concepts in the Let's Go domain

a mapping function for each instance, but rather for each concept, since the former could cause severe data sparseness. In order to estimate the empirical distribution in an unsupervised manner, we exploit grounding information[4] as true labels. We first parse dialog logs to look for the grounding information that the users have provided. Each time we encounter grounding information that includes the constraints used in the backend queries, this is added to the list. If two actions contradict each other, the later action overwrites the earlier one. Then, for each observation in the data, we determine its correctness by comparing it with the grounding information. Next, we gather two sets of confidence scores with respect to correctness, on which we apply a Gaussian kernel-based density estimation. After that, we scale the two estimated densities by their total number of elements to see how the ratio of correct ones over the sum of correct and incorrect ones varies according to the confidence score. The ratio computed above will be the calibrated score:

$$c' = \frac{d_c(c)}{d_c(c) + d_{inc}(c)} \tag{16}$$

where c' indicates the calibrated confidence score and c is the input confidence score. $d_c(\cdot)$ denotes the scaled density for the correct set and $d_{inc}(\cdot)$ is the scaled density for the incorrect set.

Note that this approach tends to yield a more conservative confidence score since correct user actions can exist, even though they may not match the grounding information. Finally, in order to efficiently obtain the calibrated score for a given confidence score, we employ the *sparse Bayesian regression* (Tipping, 2001) with the Gaussian kernel. By

[4]Specifically, we used explicitly confirmed information by the system for this study

virtue of the sparse representation, we only need to consider a few so-called *relevance vectors* to compute the score:

$$y(\mathbf{x}) = \sum_{\mathbf{x}_n \in RV} w_n k(\mathbf{x}, \mathbf{x}_n) + b \qquad (17)$$

where RV denotes the set of relevance vectors, $|RV| \ll |\{\mathbf{x}_n\}|$. $k(\cdot,\cdot)$ represents a kernel function and b is a bias parameter. Figure 2 shows the aforementioned process for several representative concepts in the Let's Go domain.

5 Experimental Setup

To verify the proposed method, three months of data from the Let's Go domain were used to train the user action model and the observation model. The training data consists of 2,718 dialogs and 23,044 turns in total. To evaluate the user action model, we compared overall system performance with three different configurations: 1) the uniform distribution, 2) the user action model without historical information[5] which is comparable to the bigram model of (Keizer et al., 2008), 3) the user action model with historical information included. For system performance evaluation, we used a user simulator (Lee and Eskenazi, 2012) which provides a large number of dialogs with statistically similar conditions. Also, the simulated user enables us to examine how performance changes over a variety of error levels. This simulated user supports four error levels and each model was evaluated by generating 2,000 dialogs at each error level. System performance was measured in terms of average dialog success rate. A dialog is considered to be successful if the system provides the bus schedule information that satisfies the user goal.

To measure the effectiveness of the calibration method, we conducted two experiments. First, we applied the calibration method to parameter learning for the user action model by using the calibrated confidence score in Equation 7. We compared the log-likelihood of two models, one with calibration and the other without calibration. Second, we compared overall system performance with four different settings: 1) the user action model with historical information and the observation model with calibration, 2) the user action model with historical information and the observation model without calibration, 3) the user action model without historical information and the observation model with calibration, 4) the user action model without historical information and the observation model without calibration.

[5]This model was constructed by marginalizing out the historical variables.

6 Results

The effect of parameter learning of the user action model on average dialog success rate is shown in Figure 3. While, in the previous study, the bigram model unexpectedly did not show a significant effect, our result here indicates that our comparable model, i.e. the model with historical information excluded, significantly outperformed the baseline uniform model. The difference could be attributed to the fact that the previous study did not take transcription errors into consideration, whereas our approach handles the problem by treating the true user action as hidden. However, we cannot directly compare this result with the previous study since the target domains are different. The model with historical information included also consistently surpassed the uniform model. Interestingly, there is a noticeable trend: the model without historical information performs better as the error level increases. This result may indicate that the simpler model is more robust

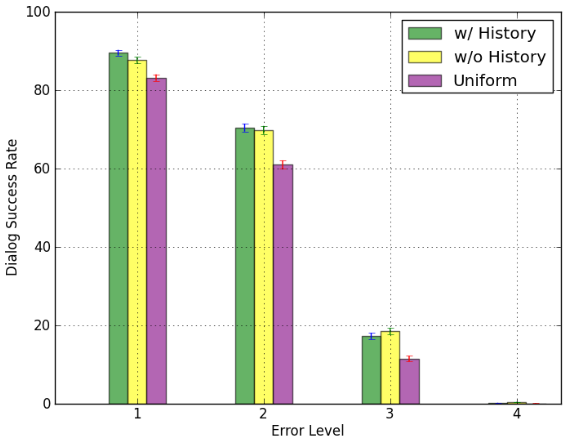

Figure 3: The effect of parameter learning of each user action model on overall system performance. The error bar represents standard error.

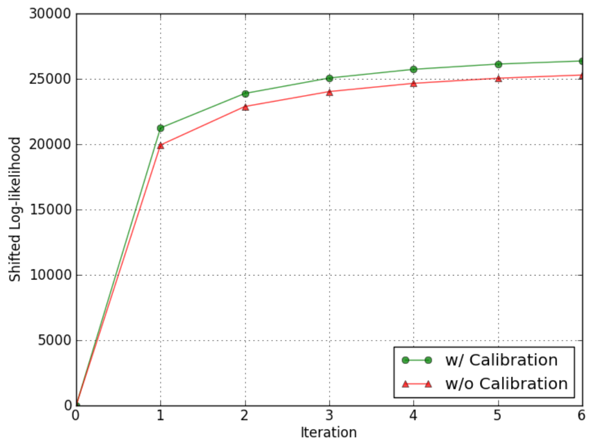

Figure 4: The effect of confidence score calibration on the log-likelihood of the user action model during the training process.

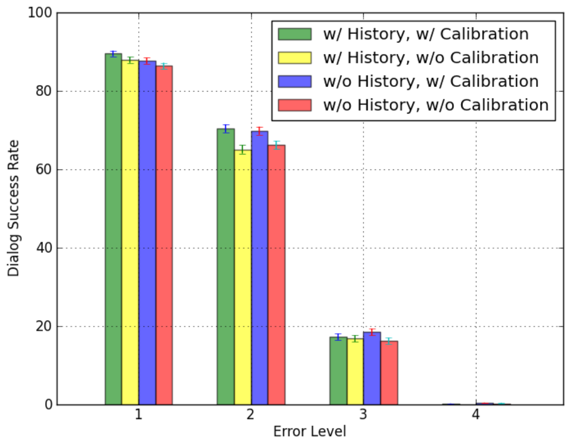

Figure 5: The effect of confidence score calibration for the observation model on overall system performance. The error bar shows standard error.

to error. Although average dialog success rates became almost zero at error level four, this result is a natural consequence of the fact that the majority of the dialogs in this corpus are failed dialogs.

Figure 4 shows the effect of confidence score calibration on the log-likelihood of the user action model during the training process. To take into account the fact that different confidence scores result in different log-likelihoods regardless of the quality of the confidence score, we shifted both log-likelihoods to zero at the beginning. This modification more clearly shows how the quality of the confidence score influences the log-likelihood maximization process. The result shows that the calibrated confidence score gives greater log-likelihood gains, which implies that the user action model can better describe the distribution of the data.

The effect of confidence score calibration for the observation model on average dialog success rate is presented in Figure 5. For both the user action model with historical information included and excluded, the application of the confidence score calibration consistently improved overall system performance. This result implies the possibility of automatically improving confidence scores in a modularized manner without introducing a dependence on the underlying methods of ASR and SLU.

7 Conclusion

In this paper, we have presented novel unsupervised approaches for learning the user action model and improving the observation model that constitute the partition-based belief tracking method. Our proposed method can learn a user action model directly from a machine-transcribed spoken dialog corpus. The enhanced system performance shows the effectiveness of the learned model in spite of the lack of human intervention. Also, we have addressed confidence score calibration in a unsupervised fashion using dialog-level grounding information. The proposed method was verified by showing the positive influence on the user action model learning process and the overall system performance evaluation. This method may take us a step closer to being able to automatically update our models while the system is live. Although the proposed method does not deal with N-best ASR results, the extension to support N-best results will be one of our future directions, as soon as the Let's Go system uses N-best ASR results.

Acknowledgments

This work was supported by the second Brain Korea 21 project.

References

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

M. Gasic and S. Young, 2011. Effective handling of dialogue state in the hidden information state POMDP-based dialogue manager. *ACM Transactions on Speech and Language Processing*, 7(3).

J. Henderson, O. Lemon, K. Georgila, 2008. Hybrid Reinforcement / Supervised Learning of Dialogue Policies from Fixed Datasets. *Computational Linguistics*, 34(4):487-511.

F. Jurcicek, B. Thomson and S. Young, 2011. Natural Actor and Belief Critic: Reinforcement algorithm for learning parameters of dialogue systems modelled as POMDPs. *ACM Transactions on Speech and Language Processing*, 7(3).

S. Keizer, M. Gasic, F. Mairesse, B. Thomson, K. Yu, S. Young, 2008. Modelling User Behaviour in the HIS-POMDP Dialogue Manager. *In Proceedings of SLT*.

S. Lauritzen and D. J. Spiegelhalter, 1988. Local Computation and Probabilities on Graphical Structures and their Applications to Expert Systems. *Journal of Royal Statistical Society*, 50(2):157–224.

S. Lee and M. Eskenazi, 2012. An Unsupervised Approach to User Simulation: toward Self-Improving Dialog Systems. *In Proceedings of SIGDIAL*. http://infinitive.lti.cs.cmu.edu:9090.

G. Parisi, 1988. *Statistical Field Theory*. Addison-Wesley.

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

A. Raux and Y. Ma, 2011. Efficient Probabilistic Tracking of User Goal and Dialog History for Spoken Dialog Systems. *In Proceedings of Interspeech*.

N. Roy, J. Pineau, and S. Thrun, 2000. Spoken dialogue management using probabilistic reasoning. *In Proceedings of ACL*.

U. Syed and J. Williams, 2008. Using automatically transcribed dialogs to learn user models in a spoken dialog system. *In Proceedings of ACL*.

B. Thomson and S. Young, 2010. Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems. *Computer Speech & Language*, 24(4):562-588.

B. Thomson, F. Jurccek, M. Gasic, S. Keizer, F. Mairesse, K. Yu, S. Young, 2010. Parameter learning for POMDP spoken dialogue models. *In Proceedings of SLT*.

M. Tipping, 2001. Sparse Bayesian Learning and the Relevance Vector Machine. *Journal of Machine Learning Research*, 1:211–244.

J. Williams, P. Poupart, and S. Young, 2005. Factored Partially Observable Markov Decision Processes for Dialogue Management. *In Proceedings of Knowledge and Reasoning in Practical Dialogue Systems*.

J. Williams and S. Young, 2007. Partially observable Markov decision processes for spoken dialog systems. *Computer Speech & Language*, 21(2):393-422.

J. Williams, 2008. Exploiting the ASR N-best by tracking multiple dialog state hypotheses. *In Proceedings of Interspeech*.

J. Williams, 2010. Incremental partition recombination for efficient tracking of multiple dialog states. *In Proceedings of ICASSP*.

S. Young, M. Gasic, S. Keizer, F. Mairesse, J. Schatzmann, B. Thomson and K. Yu, 2010. The Hidden Information State Model: a practical framework for POMDP-based spoken dialogue management. *Computer Speech and Language*, 24(2):150–174.