USING DEEP-Q NETWORK TO SELECT CANDIDATES FROM N-BEST SPEECH RECOGNITION HYPOTHESES FOR ENHANCING DIALOGUE STATE TRACKING

Richard Tzong-Han Tsai[1,4] *Chia-Hao Chen*[1,4] *Chun-Kai Wu*[2] *Yu-Cheng Hsiao*[1]
Hung-yi Lee[3,4]

[1]Department of Computer Science and Information Engineering, National Central University,
[2]Department of Computer Science, National Tsing Hua University,
[3]National Taiwan University Electrical Engineering Department,
[4]NTU IoX Center, National Taiwan University, Taipei, Taiwan
Email: [1]thtsai@g.ncu.edu.tw

ABSTRACT

Most state-of-the-art dialogue state tracking (DST) methods infer the dialogue state based on ground-truth transcriptions of utterances. In real-world situations, utterances are transcribed by automatic speech recognition (ASR) systems, which output the n-best candidate transcriptions (hypotheses). In certain noisy environments, the best transcription is often imperfect, severely influencing DST accuracy and possibly causing the dialogue system to stall or loop. The missed or misrecognized words can often be found in the runner-up candidate transcriptions from 2 to n, which could be used to improve accuracy of DST. However, looking beyond the top-ranked ASR results poses a dilemma: going too far may introduce noise, while not going far enough may not uncover any useful information. In this paper, we propose a novel approach to automatically determine the optimal time to stop reexamining runner-up ASR transcriptions based on deep reinforcement learning. Our method outperforms the baseline system, which uses only the top-1 ASR result, by 3.1%. Then, we select the dialogue rounds with the top-10 largest word error rate (WER), our method can improve DST accuracy by 15.4%, which is five times the overall improvement rate (3.1%). This improvement was expected because our proposed method is able to select informative ASR results at any rank.

Index Terms— Automatic Speech Recognition, Dialogue State Tracking, Deep Reinforcement Learning, Deep-Q Network

1. INTRODUCTION

Dialogue state tracking (DST) is an important component of a dialogue system [1]. It keeps track of the current state in the overall architecture or flow of the dialogue. The dialogue state encodes the information needed to successfully retain and finish a dialogue, such as users' goals or requests. In the slot-filling schema, the state comprises a predefined set of variables with a predefined domain of expression for each of them. In the recent context of end-to-end machine learning dialogue systems, state tracking is an essential element of such architectures [2, 3, 4, 5].

Currently, most DST systems infer the dialogue state based on manual transcriptions of utterances [6]. In real-world situations, however, utterances are transcribed by automatic speech recognition (ASR) systems. Such systems output the n-best candidate transcriptions. In certain noisy environments, the best transcription is often imperfect; that is, some words are missed or incorrectly recognized, severely influencing DST accuracy and possibly causing the dialogue system to stall or loop.

Table 1 is a list of the top five ASR results of a users response in a human-machine dialogue selected from the official dataset of the 2^{nd} Dialogue State Tracking Challenge (DSTC), the top research challenge in the field:

Table 1: *An example of top five ASR results*

System : What kind of food would you like?		
rank	**ASR result of the user utterance**	**score**
1	which for	-0.333899
2	which four	-2.572319
3	price for	-2.795018
4	for	-3.591716
5	which **french** for	-3.626649

We can see that in the top 4 results, no food type appears. However, the rank-5 result contains the keyword **french**. According to our observations, the missed or incorrectly recognized words can often be found in the runner-up candidate transcriptions from 2 to n, which could be used to improve accuracy of DST. However, looking beyond the top-ranked ASR results poses a dilemma: going too far (n is too large) may introduce noise, while not going far enough may not uncover any useful information. In this paper, we propose a novel approach to automatically determine the optimal time

to stop reexamining runner-up ASR transcriptions based on deep reinforcement learning. Our system employs the deep-Q neural network model to decide whether that current ASR should be accepted and whether the next ASR result should be examined. The process then repeats until the model decides to stop.

2. RELATED WORK

2.1. Dialogue State Tracking

In recent years, there have been many studies related to the tracking of the dialogue state. Most commercial systems have used rule-based methods [7, 8] to update the dialogue state based on the language understanding result with the highest confidence [9]. The most widely recognized competition in this task is the Dialogue System Technology Challenge [10], formerly the Dialogue State Tracking Challenge (DSTC), which is organized by Cambridge University and Microsoft Research. The challenge has been held six times since 2013 with new themes every year. Since our research explores how to effectively select multiple ASR results in human-machine dialogues, we use the DSTC 2 dataset, which contained human-computer dialogues related to restaurant.

2.2. Reinforcement Learning

At present, reinforcement learning approaches can roughly be divided into two types: those that model the environment where the agent is situated and those that do not. The latter, such as Q learning [11] and Sarsa [12], are called model-free. On the other hand, in model-based approaches, the real world needs to be modeled and a virtual environment needs to be constructed. Another classification method is based on probability or value-based differentiation, such as Policy Gradient based on probability [13] and value-based learning such as Q learning. There are also methods combining the advantages of the two such as Actor-Critic [14] in which actor acts based on probability and critic gives values based on actions. The last classification method is online learning and offline learning. Online learning refers to learning based on one's own past experience, while off-line learning can be learned by watching others' experiences.

Q learning is one of the most representative offline learning methods, and with the progress of deep learning, there has been further research on deep-Q network [15].

Table 2: *An example of actions that are taken in a round.*

System : Would you like something in the cheap, moderate, or expensive price range?		
rank	**ASR result of the user utterance**	**Action**
1	i don't care about the price range	Accept and proceed
2	i don't about the price range	Discard and proceed
3	i don't care care about the price range	Discard and stop
4	i don't care the price range	NULL
5	i don't care about what the price range	NULL

2.3. Q-Learning

Q-learning updates the value of a state-action pair after the action has been taken in the state and an immediate reward [16] has been received. Q-learning will converge to an optimal value function under conditions of adequately visiting each state-action pair, but often needs many learning episodes to do so [11]. When an action a is taken in state s, the value of a state-action pair, is updated as

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma\, Q(s') - Q(s,a)) \quad (1)$$

where $\alpha \in [0, 1]$ is the learning rate, r is the reward, γ is the discount factor, s' is the next state, and $Q(s) = max_a Q(s,a)$. The actions are chosen by some exploration policies, such as an ϵ-greedy approach, which selects the action that maximizes the Q-value with probability $1 - \epsilon$ and a random action with probability ϵ. These policies are chosen for balancing the exploration of uncertain states and actions with the exploitation of the current policy. In a stationary environment, it is also common to decay the exploration rate (ϵ) as a policy is learned as another way to begin to deal with this tradeoff.

3. METHOD

3.1. Problem Formulation

For each user utterance, we aim to perform one round of examining all ASR results from the most to the least probable and selecting informative ones. We formulate this problem as a reinforcement learning (RL) task.

RL addresses the problem of an agent learning to act in an environment to maximize a scalar reward signal. At each time step t = 0, 1, 2, . . . , the environment provides the agent with an observation s_t, the agent responds by picking an action a_t. Then, the environment offers the next reward r_{t+1}, discount γ_{t+1} and state s_{t+1}. This interaction is formalized as a Markov Decision Process (MDP) [17, 18]. In our system, MDPs will be episodic with a constant $\gamma_{t+1} = \gamma \in [0,1]$, except on termination where $\gamma = 0$. Next, we describe the state, action, and reward in our formulation. The ASR result under examination is referred as the rank-i ASR result.

State: The current state (s_t) of our RL model contains three elements: the vector of the rank-i ASR result (of the user message), the vector of the previous system message, and the score of the rank-i ASR result. In Figure 1, we use the FastText [19] word-embedding model, which is trained on the whole Wikipedia corpus, to generate the representation vector for each word. Then, we feed vectors of the system message word by word into an long-short-term-memory (LSTM) [20] encoder to generate the representation vector v_s of the whole system message sentence. We also feed vectors of the ASR result word by word to another LSTM encoder to generate the representation vector v_a for the whole ASR result sentence.

Last, we concatenate v_s, v_a and the score of the ASR result to generate the representation of s_t.

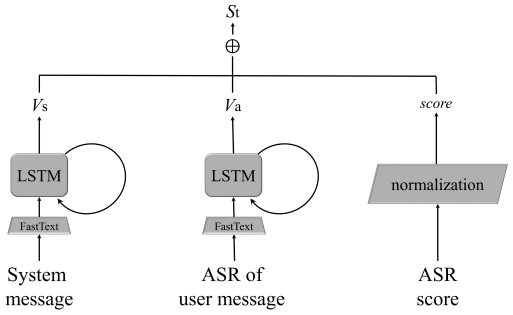

Fig. 1: *Generation of state representation*

Action: After the rank-i ASR result is processed, our RL model decides which of four actions to take next: proceed and accept, proceed and drop, stop and accept, and stop and drop. An example of each action is given below:

1. **Accept and proceed**: The rank-i ASR result is added to the reserve list and the rank-$i+1$ ASR result is read next.
2. **Accept and stop**: The rank-i ASR result is added to the reserve list and the system stops reading.
3. **Discard and proceed**: The rank-i ASR result is discarded and the rank-$i+1$ ASR result is read next.
4. **Discard and stop**: The rank-i ASR result is discarded and the system stops reading.

Table 2 shows the agent accepting the first ASR result, discarding the second, and stopping after accepting the third. Once the system has finished reading all ASR results, for each ASR result r in the reserve list, r is converted to an embedding vector and fed to the baseline DST model, which can only read one ASR result at a time.

Reward: After each action is taken, the environment should reward the agent to guide the decision model in the correct direction in Table 3. In our system, we only give non-zero rewards for the final action of each round of ASR result selection. If at least one ASR result is selected in this round, we reward the system with the DST accuracy [0-1]. If no ASR result is selected in this round, the reward is -1. Actions taken prior to the final action in a selection round are always given a reward of zero. In Table 4, we can see that the first two actions of Round 0 get rewards of 0, while only the last action gets a positive reward, 0.66.

3.2. Deep reinforcement learning and deep-Q network

In deep reinforcement learning, the various components of agents, such as policies $\pi(s,a)$ or values $Q(s,a)$, are represented with deep neural networks. The parameters of these networks are trained by gradient descent to minimize the value of the loss function. The deep-Q network (DQN) [21] integrates deep networks and reinforcement learning by using a convolutional neural network to approximate the action values for a given state s_t.

Table 3: *Action reward scheme*

Condition	Reward
If no ASR is selected and the agent stops reading	-1
if at least one ASR result is selected and the agent stops reading	DST accuracy [0-1]
Other cases	0

Table 4: *Action reward example*

Round-id	Action	Reward
0	Accept and proceed	0
0	Accept and proceed	0
0	Accept and stop	0.66
1	Discard and process	0
1	Accept and stop	0.33

At each step, the agent perceives the current state and chooses an action ε-greedily, and then adds a transition to an experience-replay memory, which keeps the last million transitions. The parameters of the neural network are optimized by using stochastic gradient descent to minimize the loss function value, as follows:

$$L = (r_{t+1} + \gamma \max_{a'} Q_{\overline{\theta}}(s_{t+1}, a') - Q_{\theta}(s_t, a_t))^2 \quad (2)$$

where t is a time step randomly picked from the replay memory. The gradient of the loss is back-propagated only into the parameters θ of the online network; the term $\overline{\theta}$ represents the parameters of a target network, a periodic copy of the online network which is not directly optimized. We employ Adam, an algorithm for first-order gradient-based optimization of stochastic objective functions, on mini-batches sampled uniformly from the experience replay. This means that in the loss function above, the time index t will be a random time index from the last million transitions, rather than the current time. We use a second network (target network) during the training procedure. This second network is used to generate the target-Q values that will be used to compute the loss function value for every action during training. The use of experience replay and target networks enables relatively stable learning of Q values.

4. EXPERIMENTS AND RESULTS

4.1. Dataset

In our experiments, we use the dataset published by DSTC 2. The data set contains a set of user-computer conversations, the first 10 ASR results for each user message, and the dialogue state label for each user message. The training set, development set, and test set contain 1612, 506, and 1117 conversations, respectively. Eight slots are defined in the DSTC 2 data set, where "area", "food" and "price range" are provided by the user. In DSTC 2, the dialog state is given as a tuple of three slot-value pairs, such as "area = north, food = Chinese, pricerange = moderate".

4.2. Experimental Settings

Since our model requires a top-performed DST system that reads one input at a time, we use Plátek et al.'s [6][1] because it outpeforms all publicly available systems. Table 5 shows the parameters we use for the DQN. The memory size is preset to 100,000. The parameter gamma, set to 0.95, is used to attenuate future awards. The word-embedding size and sentence-embedding size are both set to 300. Table 6 shows the parameters of Plátek et al.'s DST system. The batch size is set to 5 because this value achieves maximum performance in the development set. The parameter use_db_encoder is set to true, indicating that the DSTC ontology is used in our experiment.

Table 5: *Parameter settings of DQN*

Parameter	Value
memory	100000
γ	0.95
word embedding size	300
sentence embedding size	300
learning rate	0.01

Table 6: *The parameters of Plátek et al.'s DST system*

Parameter	Value
Batch size	5
encoder size	100
word embed size	100
use db encoder	150
encoder layers	1
decoder layers	1
learning rate	0.0005
epochs	50

4.3. Results and Discussion

We can observe from Table 7 that our method outperforms the baseline system, which uses only the rank-1 ASR result, by 3.1%. This improvement is expected because our method is able to select informative ASR results at any rank. We also find that integrating the top K ASR results does not improve the performance. Because the top K ASR results are noisy, simply integrating all the results without selection cannot improve the performance. In Table 8, we further analyze the performance on the top $N\%$ largest word-error-rate (WER) utterances in the test set ($N = 10, 20, 30$). It is clear that the proposed approach remarkably outperforms the baseline on the utterances with larger WER. This further verifies that the proposed approach mitigates the impact of ASR errors on DST.

We divide all errors into three types. The first type is caused by Plátek et al.'s DST system invoked by our DQN-based selection. In Table 9, we can see that the rank-1 and rank-2 result are "nor," and "North." Our DQN model accepts the rank-2 result and the state should be changed to "[area=North]," However, the called DST system incorrectly outputs "none" when the input is "North," so the state is predicted to be wrong.

The second type of error occurs when none of the top-10 ASR results contain any useful information about the dialogue state. In this case, DST accuracy will not be improved by examining runner-up ASR results.The third type occurs when our DQN model incorrectly discards the best runner-up ASR result based on its experience in the training process. In this case, the DST system cannot derive the correct dialogue state.

[1]We traced and found that the original Plátek's system used both the manual transcription and the rank-1 ASR result as the input. For fair comparison, we revised their code to use only the rank-1 ASR result. The code of Plátek's system is at https://github.com/oplatek/e2end.

Table 7: *Performance comparison with top-K ASR.*

Configuration	Test
top-1 ASR	56.9%
top-2 ASR	55.4%
top-3 ASR	54.6%
top-4 ASR	54.7%
top-5 ASR	53.3%
Our Method	**60.0**%

Table 8: *Performance analysis on utterances with top N % largest WER.*

	top-N largest WER utterances		
	$N = 10\%$	$N = 20\%$	$N = 30\%$
Baseline	47.7%	51.0%	54.2%
Our Method	**63.1**%	**57.6**%	**54.7**%

Table 9: *An example of the first error type.*

System : What part of town do you have in mind?		
rank	**ASR result of the user utterance**	**score**
1	nor	-0.521046
2	North	-1.647163

5. CONCLUSION

This paper describes a novel method of dialogue state tracking with candidate selection from the n-best speech recognition hypotheses by deep reinforcement learning. It is one of the first attempts to develop a DST method that uses only ASR results as the input instead of manual transcriptions of user utterances. In addition to the experimental results presented in the experimental section, our proposed approach offers three advantages compared to state-of-the-art tracking methods. First, we are the first to employ the DQN to select informative runner-up ASR results to improve DST. Second, we have designed a set of actions to control the DQN modules behavior. Third, our method is based on reinforcement learning, with the DST accuracy of each round being used as that rounds reward. As a result, our method does not require any extra human efforts to carry out labeling and rewarding. Compared to most DST methods, which infer the dialogue state based on manual transcriptions of utterances, our method can select informative n-best ASR results to improve DST, making application of our DST system more practicable in real-world (noisy) environments. In future work, we plan to examine not only the ASR results of the current utterance but also those of the previous round in the same dialogue.

Acknowledgment

This research was supported in part by the Ministry of Science and Technology of Taiwan (MOST 107-2633-E-002-001, MOST 108-2634-F-008 -004 and MOST 104-2221-E-008 -034 -MY3), National Taiwan University, and Intel Corporation.

6. REFERENCES

[1] Steve Young, Jost Schatzmann, Karl Weilhammer, and Hui Ye, "The hidden information state approach to dialog management," in *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*. IEEE, 2007, vol. 4, pp. IV–149.

[2] Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gasic, Lina M. Rojas Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young, "A network-based end-to-end trainable task-oriented dialogue system," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. 2017, pp. 438–449, Association for Computational Linguistics.

[3] Xuesong Yang, Yun-Nung Chen, Dilek Hakkani-Tür, Paul Crook, Xiujun Li, Jianfeng Gao, and Li Deng, "End-to-end joint learning of natural language understanding and dialogue manager," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 5690–5694.

[4] Matthew Henderson, Steve Young, and Blaise Thomson, "Deep neural network approach for the dialog state tracking challenge," in *Proceedings of the SIGDIAL 2013 Conference*, 2013, pp. 467–471.

[5] Tim Paek and Roberto Pieraccini, "Automating spoken dialogue management design using machine learning: An industry perspective," *Speech communication*, vol. 50, no. 8-9, pp. 716–729, 2008.

[6] Ondřej Plátek, Petr Bělohlávek, Vojtěch Hudeček, and Filip Jurčíček, "Recurrent neural networks for dialogue state tracking," *arXiv preprint arXiv:1606.08733*, 2016.

[7] Kai Sun, Lu Chen, Su Zhu, and Kai Yu, "A generalized rule based tracker for dialogue state tracking," in *Spoken Language Technology Workshop (SLT), 2014 IEEE*. IEEE, 2014, pp. 330–335.

[8] David Goddeau, Helen Meng, Joseph Polifroni, Stephanie Seneff, and Senis Busayapongchai, "A form-based dialogue manager for spoken language applications," in *Spoken Language, 1996. ICSLP 96. Proceedings., Fourth International Conference on*. IEEE, 1996, vol. 2, pp. 701–704.

[9] Zhuoran Wang and Oliver Lemon, "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*, 2013, pp. 423–432.

[10] Jason Williams, Antoine Raux, and Matthew Henderson, "The dialog state tracking challenge series: A review," *Dialogue & Discourse*, vol. 7, no. 3, pp. 4–33, 2016.

[11] Christopher JCH Watkins and Peter Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[12] Gavin A Rummery and Mahesan Niranjan, *On-line Q-learning using connectionist systems*, vol. 37, University of Cambridge, Department of Engineering, 1994.

[13] Jan Peters and Stefan Schaal, "Policy gradient methods for robotics," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*. IEEE, 2006, pp. 2219–2225.

[14] Jan Peters, Sethu Vijayakumar, and Stefan Schaal, "Natural actor-critic," in *European Conference on Machine Learning*. Springer, 2005, pp. 280–291.

[15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[16] Christopher John Cornish Hellaby Watkins, *Learning from delayed rewards*, Ph.D. thesis, King's College, Cambridge, 1989.

[17] Anthony Cassandra, Michael L Littman, and Nevin L Zhang, "Incremental pruning: A simple, fast, exact method for partially observable markov decision processes," in *Proceedings of the Thirteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 1997, pp. 54–61.

[18] Jason D Williams, Pascal Poupart, and Steve Young, "Factored partially observable markov decision processes for dialogue management," in *Proc. IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, 2005, pp. 76–82.

[19] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov, "Enriching word vectors with subword information," *arXiv preprint arXiv:1607.04606*, 2016.

[20] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[21] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529, 2015.