A Deep Reinforcement Learning based Multimodal Coaching Model (DCM) for Slot Filling in Spoken Language Understanding (SLU)

Yu Wang[1], *Yilin Shen*[1], *Hongxia Jin*[1]

[1]Samsung Research America

yu.wang1@samsung.com, yilin.shen@samsung.com, hongxia.jin@samsung.com

Abstract

In this paper, a deep reinforcement learning(DRL) based multimodal coaching model (DCM) for slot filling task in SLU is proposed. The DCM takes advantage of a DRL based model as a coach of the system to learn the wrong labeled slots with/without user's feedback, hence may further improve the performance of an SLU system. This DCM model is an improved model of the deep reinforcement learning based augmented tagging model as introduced in [1], by using a better DRL model with different rewards and adding in a user's feedback modal to achieve *one-shot* learning.The performance of DCM is evaluated on two datasets: one is the benchmark ATIS corpus dataset, another is our in-house dataset with three different domains. It shows that the new system gives a better performance than the current state-of-the-art results on ATIS by using DCM. Furthermore, we build a demo application to further explain how user's input can also be used as a real-time coach to improve model's performance even more.

1. Introduction

Slot filling is one of the most important tasks in spoken language understanding (SLU), which is normally formulated as a sequence labeling problem. Some effective algorithms for this task include conditional random fields (CRFs), recurrent neural networks (RNN), or a combination of these models [2, 3, 4, 5]. Recent works also take advantage of more advanced RNN based model like sequence-to-sequence/encoder-decoder structures, which can take the attention based contextual features to further improve the model's performance [6]. The details of these works will be given in next section.

The performance of different models are normally evaluated by their F1 scores, which is mainly due to the imbalance data distribution in most slot filling tasks. Despite recent models demonstrate better performance, it becomes harder to make further improvement by using more advanced or complex network structures. The main reasons of this issue are from three aspects:

1. Most of the tags in a slot filling task are minority tags, which only counts a small percentage of the entire dataset. The reason is that tokens in word sequences are mostly labeled as 'no semantice tags', *i.e.* 'O'. Also among the misclassified tokens, we observe that most of them are with minority tags. By changing the learning model structure, it will first improve the model performance on majority tags. Sometimes a model can improve its performance on minority tags, but at a cost of degrading that on the majority tags, which can be treated as a common issue for the imbalanced data [7, 8].
2. Despite the system performance is improved by using more complicated deep learning models, the number of training parameters also increases, which is more likely to be over-fitting [9, 10, 11]. There is a bottle-neck by using more complex network structures or tuning hyper-parameters.
3. Another issue need to be noticed is that even one knows a deep learning model does not perform very well on a small portion of data with specific tags, it takes a lot of time to retrain the model using a different set of hyper-parameters or even a new model structure. The robustness of dynamic online learning for most deep learning based models are not very good [12, 13].

One possible solution for the first issue is by weighted sampling the data, *i.e.* oversampling the minority tags and undersampling the majority ones. However, this method may distort the original distribution of data, hence may degrade model's performance[14, 15]. The second issue is even harder to overcome, as over-fitting is a direct trade-off of using a more complex model, which is the direction most recent works follow. Some techniques, like regularization, dropout or max norm constraints [16, 17, 18], can resolve the over-fitting issue to some extend, but still may not work very well if the model structure becomes even more complex. For the third issue, one possible solution is to save the trained model and the wrongly labeled results, then continue to train the model using more data under the same tags as the mislabeled data. This approach, however, may adversely affect model's performance on correctly labeled data as well.

In this paper, a DRL based multimodal coaching model (DCM) for slot filling is proposed. The new model will use a DRL based slot filling model as introduced in [1], by adding in user's coaching as another modal, such that the correct tags of the wrongly labeled data can be taught by the users during training and the same mistakes won't happen again in testing. The advantage of this approach is the it is a compensatory model to make up the mistakes made by the deep learning based sequence labeling models, without changing the original model structure or re-sampling the data, hence solves the issue caused by the imbalanced data and over-fitting. Also, a new experience replay technique named as partially-fixed experience replay is proposed to achieve a faster on-line teaching without the need of retraining the entire model .

The structure of this paper is organized as following: a brief overview of our baseline RNN model is given in section 2. Section 3 will discuss the new proposed DRL based coaching system, and how it works with existing stat-of-the-art RNN based models described in section 2. In section 4, the algorithm will be tested on two datasets, one is the benchmark ATIS dataset and the other is our in-house dataset with three domains. Furthermore, a user based coaching demo is given to illustrate how our model's performance can be further enhanced by fast online training using user instructions.

2. Baseline Model

In this section, we will give a brief overview of the baseline RNN model used in our system. This model also gives the previous state-of-the-art result on ATIS dataset for slot filling task.

Slot filling task is one of the most important tasks in spo-

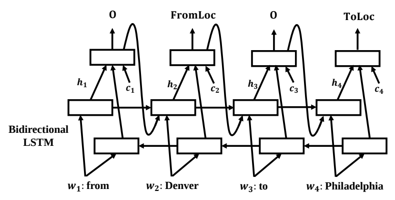


Figure 1: *Attention based Bidirectional LSTM for Slot Filling*

ken language understanding. It sequentially labels the words in an utterance using the pre-defined slot labels. The most straight-forward approach is to use a single recurrent neural network (RNN) to generate sequential tags of an utterance word by word [2]. This approach has one constraint that the number of tags generated should be the same as that of words in an utterance. One possible approach to overcome this limitation is the encoder-decoder model, in which the number of decoder's output can be different from that of the encoder's input [6].

The current state-of-the-art model which has the best performance on ATIS datatset is an attention based bidirectional RNN model as in [6]. A brief discussion about this approach will be given in this section. The model will be used as:
1. One of the baseline models for comparison in experiment section
2. The base RNN model to coarsely process the input word sequence and then generate the filtered training data for the DRL based coaching model (DCM).

The model introduced in [6] covers both of the intent detection and slot filling tasks. Considering the focus of this paper, we only discuss the slot filling part of the model. The structure of the attention based RNN model for slot filling task is as shown in Figure 1, where a bidirectional LSTM is selected as its RNN model structure. As shown in the figure, a contextual vector $c_i(\cdot)$ is defined using the attention of hidden states h_j:

$$c_i = \sum_{j=1}^{L} \alpha_{i,j} h_j \tag{1}$$

where $\alpha_{i,j}$ is the attention coefficient defined as:

$$\begin{aligned} \alpha_{i,j} &= \frac{e^{e_{i,j}}}{\sum_{k=1}^{L} e^{e_{i,k}}} \\ e_{i,k} &= \phi(s_{i-1}, h_k) \end{aligned} \tag{2}$$

where $\phi(\cdot)$ is a feed-forward neural network.
This model consists of two main properties:
1. It uses a Bi-direction LSTM (BLSTM) structure to capture the long-term dependencies in both directions.
2. The attention vector c_i gives additional contextual information, for which cannot be captured by the hidden states in BLSTM.

3. Deep Reinforcement Learning based Multimodal Coaching Model (DCM)

In this section, a novel deep reinforcement learning (DRL) based multimodal coaching model is proposed to address the issues as described earlier. The new system is built upon an existing recurrent neural network (RNN) based model f_{rnn} for slot filling. As mentioned earlier, the deep reinforcement learning based multimodal coaching model (DCM) f_{coach}, can be trained either by the given data or coached by users, in order

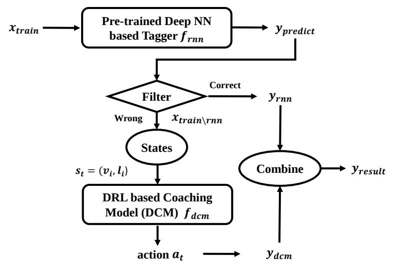


Figure 2: *Training Structure of DCM*

to generate the correct tags labeled wrongly by the f_{rnn} originally. Hence it can compensate the weakness of the original RNN based model.

3.1. Training Model Design

In this subsection, a detailed analysis of the design of the DRL based multimodal coaching model (DCM) for slot tagging is given. Following the definitions of reinforcement learning (RL) as in [19, 20, 21, 22], we will show the design of three main RL components, *i.e.* the states s_t, the actions a_t and the rewards r_t. Also a brief discussion on our DRL based training algorithm using experience replay is given. Before the discussion, the entire training structure of DCM is as shown in Figure (2), which contains several steps:
Step 1: Pre-train an RNN based tagging model f_{rnn} using the training data (x_{train}, y_{train}).
Step 2: Use the pre-trained RNN model f_{rnn} to generate the predicted outputs $y_{predict}$ and compare it with the ground truth y_{train}. The corrected tagged data pairs are labeled as (x_{rnn}, y_{rnn})
Step 3: Filter out the correctly labeled data and leave the ones with wrong labels, *i.e.* $x_{train\backslash rnn}$. Use the word tokens w_i in $x_{train\backslash rnn}$ with their correct labels l_i in $y_{train\backslash rnn}$ to generate the state s_t (as described next) for training the coaching model f_{dcm}.

The DRL based structure's design follows the model setup as in [1], which contains several elements, including states, actions, rewards and etc. A brief overview is given as following:
States (s_t): The DCM model's state s_t is as shown in Figure 3. The state is defined by each word/token w_i from its input $w_i \in x_{train\backslash rnn}$, it mainly contains two parts of information: the first part is the word level information represented by an n-gram (n is odd) averaged vector v_i, and the other part is a given label l_i of w_i, *i.e.* $s_t = [v_i, l_i]$. During the generation of training states, l_i uses all possible tags for the word/token w_i. v_i is defined as the average of the vector of word sequences from $w_{i-(n-1)/2}$ to $w_{i+(n-1)/2}$, where w_j is the center of this sequence:

$$v_i = \frac{1}{n} \sum_{j=i-(n-1)/2}^{j=i+(n-1)/2} w_j \tag{3}$$

Remarks: The reason to use an n-gram vector v_i to substitute w_i as a word level information in a state is because that we want to better capture the contextual information compared to that using a single word vector. When $n = 1$, v_i is the same as word vector w_i. Also, the average of fewer words/tokens can be used if the index of word sequences is out of the boundary of an input sentence.
Actions (a_t): The DCM model's action a_t at time step t is de-

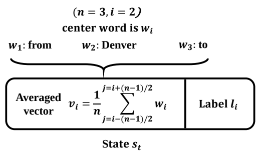


Figure 3: *State of DCM*

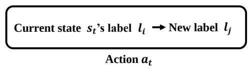


Figure 4: *Action of DCM*

fined as in Figure 4. Each action gives a transition signal such that the state will change from its current label l_i to its next label l_j by keep the same n-gram vector v_i. Simply speaking, the action set A contains all possible labels/tags for the word vector w_i or its correspond n-gram substitute v_i. At each time step t, the action a_t with the highest predicted action-reward value is chosen as:

$$a_t = \arg\max_a Q(s_t, a | a \in A, \theta_t) \tag{4}$$

where $Q(\cdot)$ is the estimated action-value function generated by the neural network in DCM model at each step, which follows its definition in DQN as in [22]. By taking the action a_t, an agent will transit from its current state to the state with a new label that is directed by a_t.

Rewards (r_t): The reward defined at a state s_t containing an n-gram vector v_i (with a center word/token w_i) will use the one-hot representation $o_{l_i^*}$ of w_i's true label l_i^*, the one-hot vector o_{l_i} of the label l_i in current state s_t, and the predicted probability p_i for the word w_i using f_{rnn} as:

$$r_t = \begin{cases} 1 & \text{if } |o_{l_i^*} - p_i||_2 > ||o_{l_i} - o_{l_i^*}||_2 \\ -1 & \text{otherwise} \end{cases} \tag{5}$$

where $||\cdot||_2$ is the $\mathcal{L}_2$ norm.

The reward is defined by comparing the distance from f_{rnn} predicted label to $w_i's$ true label and that from current state's label to its true label. The insight behind this definition is that a positive reward should be assigned to a state in which its label is more closer to the true label compared with the f_{rnn}'s predicted one. The reward function is one of the key factors to get further performance improvement using our new augmented tagger.

Remarks:

1. It is worth noticing that p_i should be generated by f_{rnn} using the same word sequence as preparing v_i. Similarly, l_i^* is the true label of w_i in the same sentence of preparing p_i and v_i.

2. In our setup, we use multiple cascaded modals (including a RNN model and a DRL model) to improve the system performance. This idea of using multiple modals'/models' collective information has been widely used in system identification [23, 24, 25, 26], reinforcement learning [27, 28, 29] and also deep neural networks/deep learning (in NLP)[30, 31]. It has been shown as an effective approach to boost the model's performance both theoretically and empirically.

Training DCM using Experience Replay: One training technique we borrowed from [22, 32, 33] is the experience replay used in DQN. It improves the convergence issue in neural network-estimator based DRL by storing the state s_t visited before, action performed a_t, state's reward r_t and the next state s_{t+1} after performing the action a_t in an experience tuple (s_t,r_t, a_t, s_{t+1}). This tuple is then pushed into the experience replay memory queue M. Whenever an action is performed then a new state is arrived, the past experience tuple will be pushed into the replay memory queue M following First-in First-out (FIFO). At each training iteration, a random tuple is selected from M and the loss function value is calculated based on the s_t, r_t, a_t and s_{t+1} in the tuple.

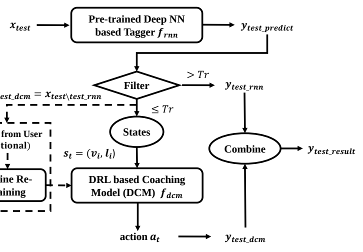


Figure 5: *Coaching Model Inference*

3.2. Model Inference

As shown in Figure 5, the inference part of DCM is a bit different from training DCM. Since there is no ground truth during inference, in order to filter those data that may be labeled wrongly by f_{rnn}, a threshold value T_r is defined. All the tokens with their predicted tags' probabilities p_i below T_r are used as the inference input of DCM, *i.e.* $w_i \in x_{test\backslash test\_rnn}$. The outputs of DCM are the actions that can transfer the tokens from their current states to the states with their target label l_i^*.

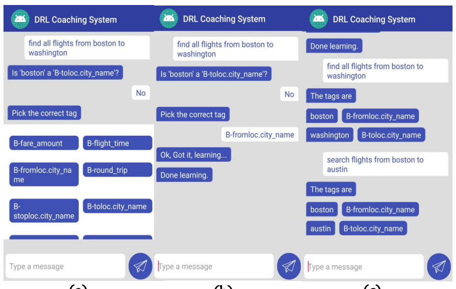
3.3. Coaching the System by Users

Beside the DRL based modal in DCM can improve a conventional RNN tagging model by using the wrongly labeled data, another important modal of DCM is the user coaching modal, in which the system can also be further coached by users in an online manner. During the inference, the model allows a user to select the correct labels of the words from its input, *i.e* $x_{test\backslash test\_rnn}$, and then retrain the model, which is as shown in the dashed box in Figure 5. The retraining step is the same as that in section 3.1, except two aspects:

1. A new state $s_{user} = [v_{user}, l_{user}]$ is generated based on user input, and put together with the selected words's current state $s_t = [v_{user}, l_i]$ as the experience replay tuple (s_t,r_t, a_t, s_{user}).
2. During experience replay, the model fixes the tuple(s) (s_t,r_t, a_t, s_{user}) containing user instructions in its replay memory M, and only push in/pop out other states. Since the tuple(s) storing user instruction(s) is/are always in the memory, it has a higher chance to be selected and further trained. In practice, it gives us a much faster online training performance, hence *one-shot* learning can be achieved.

4. Experiment

The experiment consists of two set-ups: one set-up uses DCM only to coach the system without user's feedback, and the other set-up takes the user's instruction and retraining the DCM model during the inference. The performance comparison is based on their F1 scores.



(a)
find all flights from boston to washington
Is 'boston' a 'B-toloc.city_name'?
No
Pick the correct tag
B-fare_amount
B-flight_time
B-fromloc.city_name
B-round_trip
B-stoploc.city_name
B-toloc.city_name

(b)
find all flights from boston to washington
Is 'boston' a 'B-toloc.city_name'?
No
Pick the correct tag
B-fromloc.city_name
Ok, Got it, learning...
Done learning.

(c)
Done learning.
find all flights from boston to washington
The tags are
boston B-fromloc.city_name
washington B-toloc.city_name
search flights from boston to austin
The tags are
boston B-fromloc.city_name
austin B-toloc.city_name

Figure 6: *DCM with user coaching*

4.1. Data Sets

Two datasets are used in the experiments, one is the the public ATIS dataset [34] containing utterances of flight reservations, and the other is our self-collected dataset in three different domains: Food, Home and Movie. The ATIS dataset used in this paper follows the same format as in [6, 3, 35, 4], which contains 4978 utterance in training set and 893 utterance in test set, the total number of slot tags is 127. For our self-collected dataset, it contains three domains: food, home and movie.There are 15 semantic tags in food domain, 16 semantic tags in home domain, 14 semantic tags in movie domain.

4.2. Training Setup

The neural network structure of the DRL in our DCM is chosen as LSTM, the number of LSTM states is chosen as 200. The averaged word vector in a reinforcement learning state is chosen as a trigram, *i.e.* n=3. The discount factor is chosen as $\gamma = 0.9$ and the threshold T_r for inference is set as 0.9. The size of word embedding is 128, which are initialized randomly at the beginning of the experiment. The pre-trained RNN tagging model f_{rnn} is chosen as the attention based BLSTM structure as illustrated in section 2, with a set-up follows [6]: the number of state in is 128, the drop out rate is 0.5 and batch size is 16. The models are trained individually on single Nvidia M40 GPU.

4.3. Performance of DCM with/without user coaching

The experiment is conducted using two different set-ups: one is a self-learning DCM model only based on the filtered states generated by f_{rnn} and the threshold T_r, the other set-up takes the user/coach's input such that *one-shot* learning can be achieved. For the second scenario, in order to interact with users, a coaching app is also developed on an andriod mobile as shown in Figure 6. During the inference, a small number of k sentences (here is chosen as $k = 5$) with tags below the threshold T_r are selected from the entire data input of DCM, and to be corrected by the user, as shown in Figure 6(a) and 6(b). Then the DCM on our server is retrained using the corrected user inputs for 30 seconds. Figure 6(c) shows the outputs using the same and a similar input after DCM is retrained, which gives the correct output tags correspondingly.

To future evaluate the DCM model in a more quantitative manner, the test result on ATIS by using DCM with/without user's coaching is as shown in Table 1, by comparing it with the other existing algorithms. The number of sentences for user coaching is selected as $k = \{5, 20\}$. Also, the performance of the model using self-collected data in three other domains is as shown in Table 2. It can be observed that the DCM structure without user coaching outperform the current state-of-the-art method by 0.77% on ATIS and 0.5% to 0.8% on our own three domains. The reason behind it can be simply explained: the new DCM structure is built upon the current best tagging model and future improves it by learning from its wrongly labeled slots. It coaches f_{rnn} whenever it fails or gives low quality predictions. The performance of the models with user coaching is even better, giving 2.26% higher F1 scores than the current state-of-the-art result on ATIS dataset, and 1.7% to 4.1% better F1 score on our own three domains datasets. Though the comparison between DCM with user coaching and other models is not very fair since extra information from user side is used, considering the short retraining time needed (30 seconds in our set-up), however, the user based approach has great potential for online fast teaching and learning, which can be extended to general labeling tasks.

Table 1: *Performance of Different Models on ATIS Dataset*

Model	F1 Score
Recursive NN [2]	93.96%
RNN with Label Sampling [3]	94.89%
Hybrid RNN [35]	95.06%
RNN-EM [36]	95.25%
CNN CRF [4]	95.35%
Encoder-labeler Deep LSTM [5]	95.66%
Attention Encoder-Decoder NN [6]	95.87%
Attention BSLTM [6]	95.98%
DCM without user coaching	**96.75%**
DCM with user coaching ($k = 5$)	**97.26%**
DCM with user coaching ($k = 20$)	**98.24%**

Table 2: *Performance of Different Models on Self-Collected Dataset*

Domain	Model	Size	F1 Score
Movie	Attention BLSTM	979	92.1%
	DCM without user coaching	979	92.7%
	DCM with user coaching ($k = 5$)	979	94.8%
	DCM with user coaching ($k = 20$)	979	**95.2%**
Food	Attention BLSTM	983	92.3%
	DCM without user's coaching	983	93.8%
	DCM with user coaching ($k = 5$)	983	95.8%
	DCM with user coaching ($k = 20$)	983	**96.4%**
Home	Attention BLSTM	689	96.5%
	DCM without user coaching	689	97.3%
	DCM with user coaching ($k = 5$)	689	97.6%
	DCM with user coaching ($k = 20$)	689	**98.2%**

5. Conclusion

In this paper, a new DRL based coaching model with/without user's feedback for slot filling is designed. The system uses DCM to coach the entire tagging model when "unsatisfied results" (below a threshold T_r) are generated. Furthermore, the model is designed in a manner that users can also teach the system based on their knowledge, which can be learned by the system in *one-shot*. The results generated by the DCM with/without user's input outperform the state-of-the-art models on public ATIS dataset and our own in-house dataset. More importantly, as shown in the demo, the DCM has great potential as a fast on-line coaching model for general labeling tasks.

6. References

[1] Y. Wang, A. Patel, and H. Jin, "A new concept of deep reinforcement learning based augmented general sequence tagging system," in *Proceedings of COLING 2018, the 27th International Conference on Computational Linguistics: Technical Pa-*

pers, 2018.

[2] D. Guo, G. Tur, W.-t. Yih, and G. Zweig, "Joint semantic utterance classification and slot filling with recursive neural networks," in *Spoken Language Technology Workshop (SLT), 2014 IEEE*. IEEE, 2014, pp. 554–559.

[3] B. Liu and I. Lane, "Recurrent neural network structured output prediction for spoken language understanding," in *Proc. NIPS Workshop on Machine Learning for Spoken Language Understanding and Interactions*, 2015.

[4] P. Xu and R. Sarikaya, "Convolutional neural network based triangular crf for joint intent detection and slot filling," in *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*. IEEE, 2013, pp. 78–83.

[5] G. Kurata, B. Xiang, B. Zhou, and M. Yu, "Leveraging sentence-level information with encoder lstm for semantic slot filling," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016, pp. 2077–2083.

[6] B. Liu and I. Lane, "Attention-based recurrent neural network models for joint intent detection and slot filling," *arXiv preprint arXiv:1609.01454*, 2016.

[7] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on knowledge and data engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[8] N. V. Chawla, N. Japkowicz, and A. Kotcz, "Special issue on learning from imbalanced data sets," *ACM Sigkdd Explorations Newsletter*, vol. 6, no. 1, pp. 1–6, 2004.

[9] M. A. Babyak, "What you see may not be what you get: a brief, nontechnical introduction to overfitting in regression-type models," *Psychosomatic medicine*, vol. 66, no. 3, pp. 411–421, 2004.

[10] D. M. Hawkins, "The problem of overfitting," *Journal of chemical information and computer sciences*, vol. 44, no. 1, pp. 1–12, 2004.

[11] I. V. Tetko, D. J. Livingstone, and A. I. Luik, "Neural network studies. 1. comparison of overfitting and overtraining," *Journal of chemical information and computer sciences*, vol. 35, no. 5, pp. 826–833, 1995.

[12] D. J. Montana and L. Davis, "Training feedforward neural networks using genetic algorithms." in *IJCAI*, vol. 89, 1989, pp. 762–767.

[13] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Security and Privacy (SP), 2017 IEEE Symposium on*. IEEE, 2017, pp. 39–57.

[14] Y. Sun, M. S. Kamel, and Y. Wang, "Boosting for learning multiple classes with imbalanced class distribution," in *Data Mining, 2006. ICDM'06. Sixth International Conference on*. IEEE, 2006, pp. 592–602.

[15] Y. Sun, M. S. Kamel, A. K. Wong, and Y. Wang, "Cost-sensitive boosting for classification of imbalanced data," *Pattern Recognition*, vol. 40, no. 12, pp. 3358–3378, 2007.

[16] B. Schölkopf and A. J. Smola, *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[17] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[18] N. Srebro and A. Shraibman, "Rank, trace-norm and max-norm," in *International Conference on Computational Learning Theory*. Springer, 2005, pp. 545–560.

[19] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," in *Reinforcement Learning*. Springer, 1992, pp. 5–32.

[20] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[21] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.

[22] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[23] K. S. Narendra, Y. Wang, and W. Chen, "Stability, robustness, and performance issues in second level adaptation," in *American Control Conference (ACC), 2014*. IEEE, 2014, pp. 2377–2382.

[24] ——, "Extension of second level adaptation using multiple models to siso systems," in *American Control Conference (ACC), 2015*. IEEE, 2015, pp. 171–176.

[25] Y. Wang, "Adaptive control and learning using multiple models," Ph.D. dissertation, Yale University, 2017.

[26] K. S. Narendra, Y. Wang, and W. Chen, "The rationale for second level adaptation," *arXiv preprint arXiv:1510.04989*, 2015.

[27] K. S. Narendra, S. Mukhopadyhay, and Y. Wang, "Improving the speed of response of learning algorithms using multiple models," *arXiv preprint arXiv:1510.05034*, 2015.

[28] K. S. Narendra, Y. Wang, and S. Mukhopadhay, "Fast reinforcement learning using multiple models," in *Decision and Control (CDC), 2016 IEEE 55th Conference on*. IEEE, 2016, pp. 7183–7188.

[29] Y. Wang and H. Jin, "A boosting-based deep neural networks algorithm for reinforcement learning," in *Proceedings of the 2018 American Control Conference*, 2018.

[30] Y. Wang, "A new concept using lstm neural networks for dynamic system identification," in *American Control Conference (ACC), 2017*. IEEE, 2017, pp. 5324–5329.

[31] Y. Wang, Y. Shen, and H. Jin, "A bi-model based rnn semantic frame parsing model for intent detection and slot filling," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2*, vol. 2, 2018, pp. 309–314.

[32] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[33] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[34] C. T. Hemphill, J. J. Godfrey, G. R. Doddington *et al.*, "The atis spoken language systems pilot corpus," in *Proceedings of the DARPA speech and natural language workshop*, 1990, pp. 96–101.

[35] G. Mesnil, Y. Dauphin, K. Yao, Y. Bengio, L. Deng, D. Hakkani-Tur, X. He, L. Heck, G. Tur, D. Yu *et al.*, "Using recurrent neural networks for slot filling in spoken language understanding," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 23, no. 3, pp. 530–539, 2015.

[36] B. Peng and K. Yao, "Recurrent neural networks with external memory for language understanding," *arXiv preprint arXiv:1506.00195*, 2015.