A Concise Introduction to Reinforcement Learning

Research · February 2018

DOI: 10.13140/RG.2.2.31027.53285

CITATIONS
8

READS
7,202

1 author:

Ahmad Hammoudeh
None
21 PUBLICATIONS **142** CITATIONS

SEE PROFILE

A Concise Introduction to Reinforcement Learning

Ahmad Hammoudeh
RLR[1], Data Science programme[2]
AIC[1], Princess Suamaya University for Technology[2]
Amman, Jordan
at.hammoudeh@gmail.com

***Abstract*— This paper aims to introduce, review, and summarize several works and research papers on Reinforcement Learning.**

Reinforcement learning is an area of Artificial Intelligence; it has emerged as an effective tool towards building artificially intelligent systems and solving sequential decision making problems. Reinforcement Learning has achieved many impressive breakthroughs in the recent years and it was able to surpass human level in many fields; it is able to play and win various games. Historically, reinforcement learning was efficient in solving some control system problems. Nowadays, it has a growing range of applications.

This work includes an introduction to reinforcement learning which demonstrates the intuition behind Reinforcement Learning in addition to the main concepts. After that, the remarkable successes of reinforcement learning are highlighted. Consequently, methods and the details for solving reinforcement learning problems are summarized. Thenceforth, data from a wide collection of works in various reinforcement learning applications were reviewed. Finally, the prospects and the challenges of reinforcement learning are discussed.

Keywords*— *Reinforcement Learning; Artificial Intelligence; Machine learning

I. INTRODUCTION

Artificial Intelligence (AI) turned out to be a hot topic in the recent years. Many articles, books, and movies were produced with raised questions like "can machines think?", "can AI exceed human intelligence?", "will machines replace humans?", "how dangerous is AI?", "what distinguishes human from AI?", in addition to enslavement problem [1]. Researchers and scientific committees are not away from these questions; Alan M. Turing discussed some of these questions [2] and hence Turing test was formulated to test machine's ability of showing intelligent behavior indistinguishable from that of a human [3]. However, some of these questions remain debatable between the most powerful CEOs (i.e. Mark Zuckerberg and Elon Musk) as well as the leading AI researchers. Discussing such questions entails comprehensive understanding of reinforcement learning (RL). I refer the reader to Stuart J. Russell work in [1].

The upraise of Artificial Intelligence is associated with Deep Learning achievements in the recent years. Deep Learning is basically a set of multiple layers of neural networks connected to each other. While Deep learning algorithms are the same as what was used in the late 1980's [4], deep learning progress is driven by the development of computational power and the tremendous increase of both generated and collected data [5]. Shifting from CPU (Central Processing Unit) to GPU (Graphics Processing Unit) [6] and later to TPU (Tensor Processing Unit) [7] accelerated processing speed and opened the door for more successes. However, computational capabilities are bounded by Moore's law [8] which may slow down building strong AI systems [9].

Reinforcement learning is learning through interaction with an environment by taking different actions and experiencing many failures and successes while trying to maximize the received rewards. The agent is not told which action to take. Reinforcement learning is similar to natural learning processes where a teacher or a supervisor is not available and learning process evolves with trial and error, different from supervised learning, in which an agent needs to be told what the correct action is for every position it encounters [1], [9].

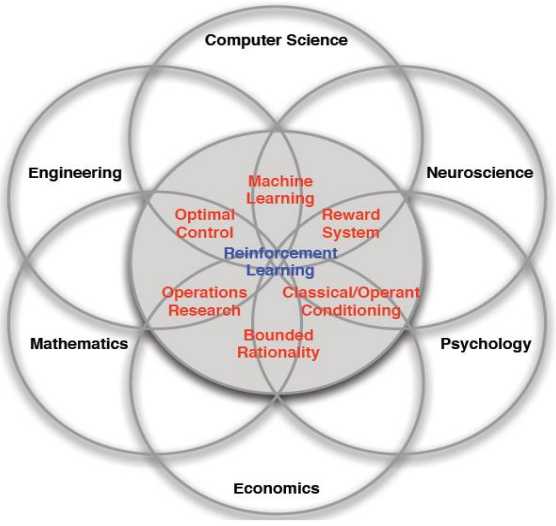

Fig. 1 Reinforcement Learning faces [10]

Reinforcement Learning overlaps with multiple fields: computer science, engineering, neuroscience, mathematics, psychology, and economics. Figure 1 demonstrates these intersections [10] .

Reinforcement learning is different from other branches of machine learning both supervised learning and unsupervised learning.

Supervised learning is the most widely studied and researched branch of machine learning. In supervised learning the machine learns from a training set of labeled data that are provided by an external teacher or supervisor who determines the correct actions that the system should take for each example. The system's task is to generalize its responses to act correctly in cases that are not available in the training examples. The performance of the supervised learning system enhances by increasing the number of the training examples. Some examples of supervised learning problems include: classification, object detection, image captioning, regression, and labeling. Although this type of learning is important, it is not adequate for interactive environments since it is unrealistic to obtain labeled data that are both representative and correct. In interactive environments, learning will be more efficient if the system can learn from its own experience. [9]

Unsupervised learning is oriented about finding structure hidden in a set of unlabeled data. Some examples of unsupervised learning include: Clustering, feature learning, dimensionality reduction, and density estimation. Even though reinforcement learning may seem as a kind of unsupervised learning since it does not learn from labeled data, it is different; reinforcement learning is trying to maximize the rewards rather than finding hidden structure [9].

Reinforcement learning is considered as a third paradigm of machine learning, along the side of unsupervised learning and supervised learning. However, the door is open for other paradigms [9].

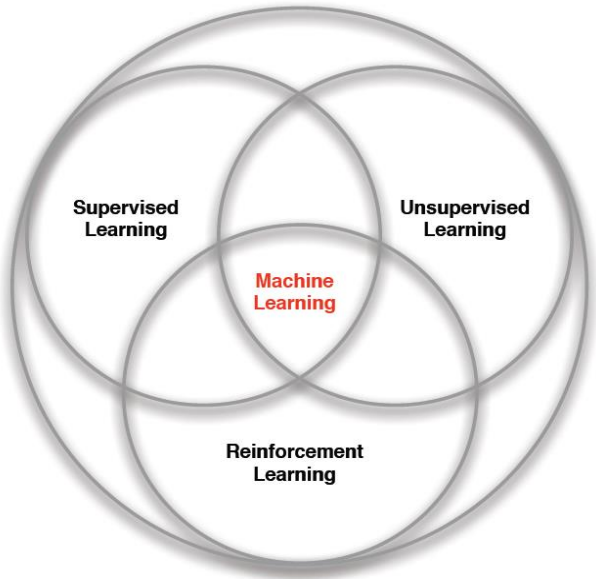

Fig. 2 Machine Learning branches [10]

The rest of this section will discuss the standard model and the basic components of reinforcement learning system. In the next section some remarkable reinforcement learning successes are highlighted. Then the idea of Markov Decision Process and Bellman optimality equations which represent the core for formulating reinforcement learning problems are discussed. After that, some algorithms for solving reinforcement problem are reviewed; starting by the general approaches such as standard tabular methods and approximate solution methods, followed by Monte Carlo method and Temporal Difference method and ending by policy based methods and deep Q-network method. In the end, some reinforcement learning applications are highlighted.

A. Standard Model

The main components of reinforcement learning system are: policy, reward signal, value function and model [9], [10].

- The policy (π) is the way that the agent (something that perceives and acts in an environment [1]) will behave under certain circumstances. Simply the policy maps states into actions. It can be a lookup table, a function, or it may involve a search process. Finding the optimal policy is the core goal of reinforcement learning process [9], [10].
- The reward signal (R) indicates how good and bad is an event and it defines the goal of the problem where the agent objective is to maximize the total received reward. Accordingly, the reward is the main factor for updating the policy. Reward may be immediate or delayed, for delayed signals the agent need to determine which actions are more relevant to a delayed reward [9], [10].
- The value function is a prediction of the total future rewards, it is used to evaluate the states and select between actions accordingly [9], [10]

The state-value function $V(s)$ is the expected return when starting from a state s [9], [10].

$$V(s) = \mathbb{E}(G_t | S_t = s) \quad (1)$$

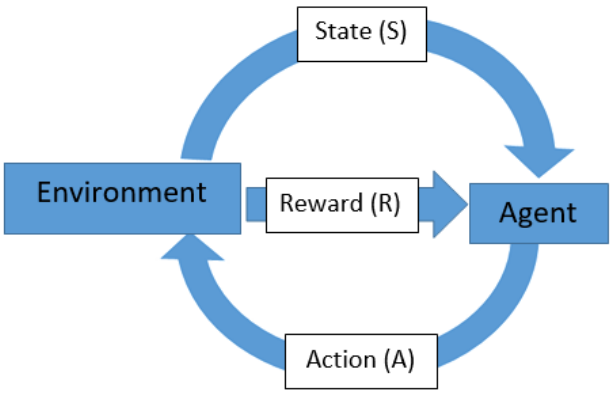

Fig. 3 Reinforcement Learning standard diagram

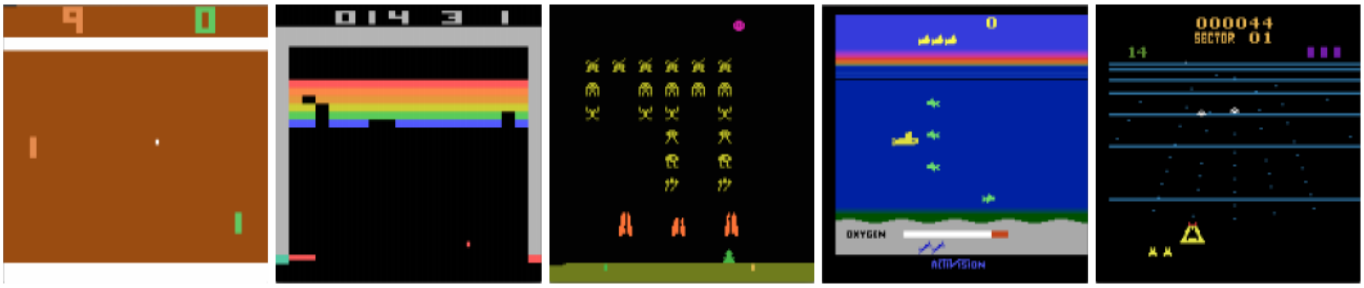

Fig. 4 Screen shots from five Atari 2600 Games: (Left-to-right) Pong, Breakout, Space Invaders, Seaquest, Beam Rider [11]

Where the return Gt is the total rewards R from time-step t. it is the sum of the immediate reward and discounted future reward [9], [10] .

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{m=0}^{\infty} \gamma^m R_{t+m+1} \quad (2)$$

The discount γ represents the degradation factor of the future rewards when they are evaluated at present, γ ranges between 0 and 1. However, the use of discount is sometimes controversial.

The action-value function q(s,a) is the expected return when starting from a state s and taking an action a. equation 3 shows the mathematical formulation [9], [10].

$$q(s,a) = \mathbb{E}(G_t \mid S_t = s, A_t = a) = \mathbb{E}\left(R_{t+1} + \sum_{m=1}^{\infty} \gamma^m R_{t+m+1} \mid S_t = s, A_t = a\right) \quad (3)$$

- The model of the environment allows predictions to be made about the behavior of the environment. However, Model is an optional element of reinforcement learning; methods that use models and planning are called model-based methods. On the other side is model free methods where the agent does not have a model for the environment. model-free methods are explicitly trial-and error learners [9], [10].

II. Reinforcement Learning Remarkable Successes

Reinforcement learning is not a new idea. however, the rise of reinforcement learning and the most achievements are recent.

Deepmind achieved human level on Atari games by a combination of reinforcement learning and deep learning, it learned to play 49 different games from self-play and the score of the game using the same algorithm without tuning to a particular game. It mapped raw screen pixels to a Deep Q Network. DeepMind published details of a program that can play Atari at a professional level in 2013 [11]. Later in the beginning of 2014, Google acquired DeepMind [12].

Gerald Tesauro from IBM developed a program that plays backgammon using reinforcement learning. Gerald's program was able to surpass human players [13]. However, scaling this success to more complicated games proved to be difficult until 2016 when DeepMind trained AlphaGo program using reinforcement learning and succeeded in defeating Go's world champion, Lee Sedol. Go is an ancient Chinese game which had been much harder for computers to master [14].

One of the noticeable achievements in the field of reinforcement learning is AlphaZero that achieved a superhuman level in more than one game: Chess, Shogi (Japanese chess), and Go in 24 hours of self-play. The breakthrough is that it reuses the same hyperparameters for all games without game-specific tuning [15] .

In the field of control; Andrew Ng and peter Abbeel succeeded in flying a helicopter automatically in 2006. They had a pilot flying the helicopter to help find a model of helicopter dynamics and a reward function. Then they used reinforcement learning to find an optimized controller for the resulting model and reward function [16], [17].

Fig. 5 Reinforcement learning was implemented to make strategic decisions in Jeopardy! (IBM's Watson 2011) [18]

A. Hammouden, "A Concise Introduction to Reinforcement Learning," 2018.

III. MARKOV DECISION PROCESS (MDP)

The problem of Markov Decision Process consists of sequential decisions in which an action (A) has to be taken in each state (S) that is visited by the agent. Markov decision process can be represented as a sequence of states, actions, and rewards as shown in the following line [10]

$$\ldots\ldots S_t, A_t, R_t, S_{t+1}, A_{t+1}, R_{t+1}, S_{t+2}, A_{t+2}, R_{t+2}, \ldots\ldots\ldots$$

The main property of Markov process is that the future is independent of the past given the present. Equation 4 formulates Markov state as that the probability of the next state S_{t+1} depends only on the current state S_t regardless the past states $S_{t-1}, S_{t-2}, S_2, S_1$ [10]

$$Prob(S_{t+1}|S_t) = Prob(S_{t+1}|S_t, S_{t-1}, \ldots\ldots S_2, S_1) \quad (4)$$

A. *Bellman optimality equation*

MDPs are well studied in control theory, and their bases go back to the pioneering work of Richard Bellman. Bellman's main contribution was to show that solving MDP using dynamic programming (DP) reduces the computational burdens effectively [19], [20].

The goal of an agent is to take actions that maximizes the total rewards, basically this an optimality problem where the reinforcement learning agent tries taking actions that lead to maximum rewards which can be represented by the action value function q(s, α) [9], [10]

The optimal action value function $q_*(s,a)$ is the maximum action-value function over all policies

$$q_*(s,a) = \max_\pi (q_\pi(s,a)) \quad (5)$$

Bellman optimality equation (equation 7) takes advantage of the recursive form of the action value function q(s, α) which can be rewritten in the recursive form as below (equation 6) [9], [10]

$$q(s,a) = R(s,a) + \gamma\, q(s',a') \quad (6)$$

Bellman optimality equation:

$$q_*(s,a) = R(s,a) + \gamma \sum_{s'} \mathrm{T}(s,a,s') \max_{a'} (q_\pi(s',a')) \quad (7)$$

Where:

R(s,α) is the immediate reward given the current state s and the current action a

$$R(s,a) = \mathbb{E}(R_{t+1} \mid S_t = s, A_t = a) \quad (8)$$

$q(s',a')$ is the action value function given the state s' (next possible state) and the action a' (next possible action)

$\mathrm{T}(s,a,s')$ is the probability of a transition to state s' given the current state s and the current action a

$$\mathrm{T}(s,a,s') = \mathbb{E}(S_{t+1} = s' \mid S_t = s, A_t = a) \quad (9)$$

Bellman Optimality Equation is non-linear and in general no closed form solution is available. However, many iterative solution methods have been reported such as value iteration, policy iteration, Q-learning [21], and SARSA [22] .

Classical methods of DP such as policy iteration and value iteration may break down when it comes to large-scale and complex MDPs; there are two barriers that may hinder MDP's scalability 1) the curse of modeling which is the difficulty of computing the transition probabilities and 2) the curse of dimensionality which raises when manipulating the elements of MDP becomes challenging. However, adaptive functions approximations [23] and learning-based methods [24] act well in finding near-optimal solution for large-scale MDPs.

IV. REINFORCEMENT LEARNING ALGORITHMS

Optimal policy for reinforcement learning can be found using exact solution methods or approximate solution methods (function approximation).

A. *Tabular methods*

Q-learning is a simple reinforcement learning algorithm that learns long-term optimal behavior without any model of the environment [25]. The idea behind Q-learning algorithm is that the current value of our estimate of Q can improve our estimated solution (bootstrapping) [21].

$$\Delta Q(S_t,A_t) = \alpha\delta = \alpha\,(R_{t+1} + \gamma \max_a (Q(S_{t+1},a) - Q(S_t,A_t)) \quad (10)$$

Where α is the learning rate and δ is the temporal difference (TD) error

Q-learning is off-policy; it evaluates one policy (target policy) while following another policy (behavior policy). However, finding an optimal action value function q_* under arbitrary behavior policy can be achieved using policy iteration. Q converges to the optimal action value function $q_*(s,a)$, and its greedy policy converges to an optimal policy under appropriate choice of the learning rate over time [21].

A. Hammouden, "A Concise Introduction to Reinforcement Learning," 2018.

Policy iteration requires policy improvement and policy evaluation. The later enhances the action value function estimate by minimizing temporal difference errors (TD) in experienced paths by following the policy. As the estimate develops, the policy can normally be improved by selecting greedy actions according to latest value function Q. Instead of performing previous steps separately (similar to traditional policy iteration), the process can be accelerated by allowing for interleaved steps as in generalized policy iteration [26].

B. *Approximate solution methods*

Tabular methods represent exact value functions and exact policies in tables. As the scale and the complexity of the environment increases, the required computational power dramatically increases. However, Approximations is a powerful concept that absorbs the hidden states problem and scalability problem [9].

An action value function is represented as a parametrized function approximator $\hat{q}(s, a, \theta)$ with parameter θ. the approximator can be a linear weighting of features or a deep neural network with weights as parameter θ. [10]

Approximated function parameter θ can be updated using the semi-gradient SARSA update shown in equation 11

$$\Delta \theta_t = \alpha \left(R_{t+1} + \gamma \hat{q}(S_{t+1}, A_{t+1}, \theta_t) - \hat{q}(S_t, A_t, \theta_t)\right) \frac{\partial \hat{q}(S_t, A_t, \theta_t)}{\partial \theta_t} \quad (11)$$

SARSA is an on-policy method where the behavior policy is the same as the target policy; it evaluates and follows a single policy. SARSA algorithm approximates $\hat{q}(s, a, \theta)$ and therefore the policy should not be totally greedy; it should be near greedy ($\varepsilon - greedy$) where the policy acts greedy most of the time but there is low probability ε of choosing random actions. [22]

In general, on policy methods perform better than off-policy but they find worse policies. An off policy method finds better policy (target policy) but it does not follow it therefore the performance is worse [9].

Given on-policy methods with linear function approximator, many published works reported guaranteed convergence for prediction problems [27]–[29] and guaranteed non divergence for control problems [30].

Function approximation can work well with off-policy methods such as Q-learning; a semi gradient Q-learning update for the parameters of approximate function is shown in equation 12 [25]

$$\Delta \theta_t = \alpha \left(R_{t+1} + \gamma \max_a \hat{q}(S_{t+1}, a, \theta_t) - \hat{q}(S_t, A_t, \theta_t)\right) \frac{\partial \hat{q}(S_t, A_t, \theta_t)}{\partial \theta_t} \quad (12)$$

However, Q-learning with function approximation may encounter the problem of instability; the main cause of instability is not learning or sampling, since dynamic programming suffers from divergence with function approximation; neither greedification, exploration, nor control is the main cause, since policy evaluation alone can produce instability, furthermore the complexity of function approximation is not the root cause, since linear function approximation can diverge [9].

When combining function approximation, bootstrapping, and off-policy, the risk of instability and divergence arises [28]. However, choosing two out of three is challenging; function approximation is significant for generalization and scalability, bootstrapping is important for computational and data efficiency, and off-policy learning helps in finding a better policy by freeing behavior policy from target policy [9].

Reported attempts for achieving stability and surviving the deadly triad (combining function approximation, bootstrapping, and off-policy) suggest that Experience Replay [31] and more stable targets like Double Q-learning [32] can help. The use of least-squares methods such as off-policy LSTD(λ) survives the triad easily [33], [34]. However, their computational costs scale with the square of the number of parameters. The true-gradient methods such as Gradient-TD [35], [36] and proximal-gradient-TD [37] work towards robust convergence properties of stochastic gradient descent.

C. *Monte Carlo Method and Temporal-Difference learning method*

Both Monte Carlo Method and Temporal-Difference learning method are model free with no knowledge of MDP transitions or rewards. they learn directly from episodes of experience, i.e. the MC return can be estimated by averaging the return from multiple rollouts. In the contrary from Monte Carlo which learns from complete episodes by sampling; Temporal-Difference learns from incomplete episodes by sampling and bootstrapping. However, it is possible to get the best of both Monte Carlo method and TD learning as in the TD(λ) algorithm where λ interpolates between Temporal-Difference (λ =0) and MC (λ =1) [0].

D. *Policy based RL*

Policy based reinforcement learning method search for an optimal policy directly with no value function; hence it is effective in continuous and high dimensional spaces and it has better convergence properties, but typically it converges to a local optimum rather than a global optimum. Although it can learn stochastic policies, policy evaluation is high variance [38].

Policy based method is an optimization problem where a parameterized policy is updated to maximize the return using either gradient-free (i.e. Hill Climbing [39], Nelder Mead [40], compressed network search [41], Genetic Algorithms [42]) or gradient-based optimization techniques (Gradient Descent, Conjugate Gradient, Quasi-Newton) [43], [44].

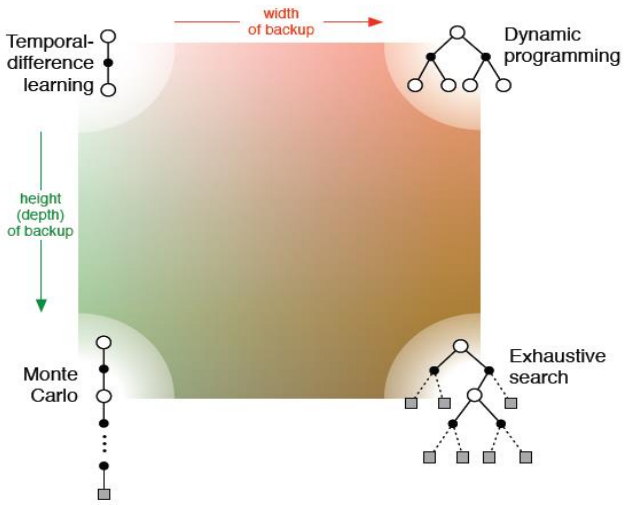

Fig. 6 A slice through the space of reinforcement learning methods, showing the most important dimensions. At the extremes of these two dimensions are: dynamic programming, exhaustive search, TD learning and Monte Carlo [9]

Neural networks are trained to estimate policies successfully using both gradient based [45]–[48] and gradient-free [49]–[51] methods. Gradient free methods perform well with small number of parameters and can optimize non-differentiable policies while gradient based methods dominate deep reinforcement learning algorithms [52] .

Actor-critic methods implement generalized policy iteration alternating between a policy improvement and a policy evaluation. A policy is called an actor because it selects actions, and an estimated value function is called a critic because it introduces critiques to the actor's actions. Actor-critic methods trade off variance reduction of policy gradients with bias introduction from value function methods [53]–[55].

Selected remarkable actor-critic works can be seen in [48], [56], [57].

E. DQN

The Deep Q-Network algorithm combines training deep neural networks with Reinforcement Learning. DQN was illustrated to work directly from raw visual inputs and on a wide variety of environments and succeeded in reaching superhuman level of Atari 2600 games [11], [58].

DQN escapes the fundamental instability problem of combining function approximation and reinforcement learning by the use of two techniques: experience replay [59] and target networks [60]. Both experience replay and target networks have been used in subsequent deep reinforcement learning works [61]–[63].

Experience replay memory stores cyclic transitions of the form (current state, current action, next state, reward) which enables the reinforcement learning agent to train on and sample from previously experienced interactions. This efficient utilization of previous experience, by learning with it multiple times, leads to better convergence behavior when training a function approximator in addition to massive reduction of the amount of interactions needed with the environment resulting in reduced variance of learning updates [64], [65].

While the original DQN algorithm of DeepMind used uniform sampling [58], In a later work Deepmind reported a more efficient learning algorithm that prioritize samples based on TD errors [64]. Even though experience replay is typically known as model-free technique, it is actually a simple model [65].

The second stabilizing technique is the target network; it breaks correlations between Q-network and target by freezing the policy network for a period of time; Instead of updating the TD error based on vacillating estimates of the Q-values, the policy network takes advantage of the fixed target network [60].

MDP assumes that the agent has full insight into the current state which is not realistic. Nevertheless, Partially Observable Markov Decision Process (POMDP) assumes that the agent receives partial observations of the state and actions will be taken based on a belief over the current state [66]. Utilizing recurrent neural networks (RNNs)[67], [68] is common with POMDP due to the sequential nature of the problem and the fact that recurrent neural networks integrate information through time and replicate DQN's performance on partially observed features. Moreover, deep recurrent Q-network (DRQN) can adapt to change of observations better than DQN [69], [70].

V. Applications

Reinforcement learning has been applied to various fields such as electric power systems, healthcare, finance, robotics, marketing, natural language processing, transportation systems, and games.

Games offer an excellent environment for reinforcement learning agent since the agent can explore different trials in a virtual world since the cost of exploration is affordable [71]. Some impressive successes of reinforcement learning in games are discussed in section II. More examples are reported in [72]–[75] in addition to a survey of reinforcement learning in video games [76]. In the following sub-sections, more applications are discussed.

A. Hyperparameters Selection for Neural Networks

Determining neural network architecture and hyperparameters selection passes through an iterative process of trial and evaluation. Hence it can be formulated as a reinforcement learning problem.

Zoph designed a recurrent neural network RNN that generates the hyperparameters for neural networks, the RNN was trained with reinforcement learning by searching in varying hyperparameters spaces to maximize the accuracy; the accuracy of the generated model on a validation set is considered a reward signal. this approach achieved competitive

results compared with state-of-art methods [77]. The concept was extended later to discovering optimization methods for deep neural networks [78], and achieved better results than standard optimization methods such as Stochastic Gradient Descent (SGD) [4], Stochastic Gradient Descent with Momentum [4], Root Mean Square Propagation (RMSProp) [79], and adaptive moment estimation (Adam) [80] .

B. *Intelligent Transportation Systems*

Intelligent Transportation Systems take advantage of latest information technologies for handling traffic, and facilitating transport networks [81]

Adaptive traffic signal control (ATSC) can lessen traffic congestion by dynamically adjusting signal timing plans in response to traffic fluctuations. Multi-agent reinforcement learning approach was proposed in [82] to solve ATSC problem where each controller (agent) is responsible for the control of traffic lights around a single traffic junction. Multi-agent reinforcement learning combines game theory with single agent reinforcement learning.

Some of the challenges that faces Multi-agent reinforcement learning approach are: the exploration-exploitation tradeoff, curse of dimensionality, stability, and nonstationarity. The nonstationarity challenge of the multi-agent learning emerges since multiple agents are learning simultaneously and every single agent has a varying learning problem in which the agent's optimal policy changes as the policies of other agents change [83].

Van der Pol, and Oliehoek developed the previous work for the traffic light control problem by eliminating the simplifying assumptions and introducing new reward function [84].

C. *Natural Language Processing*

Reinforcement Learning was utilized in various natural processing tasks such as text generation [85], [86], machine translation[87]–[89], and conversation systems. in the following subsection Dialogue systems are discussed.

1) *Dialogue Systems*

Dialogue systems are programs that interact with natural language. Generally, they are classified into two categories: chatbots and task-oriented dialog agents. The task-oriented dialog agents interact through short conversations in a specific domain to help complete specific tasks. On the other hand, the chatbots are designed to handle generic conversations and imitate human to human interactions [90].

The history of dialogue systems development is categorized in generations: the first generation is a template based system centered on rules designed by human experts, the second generation is a data driven system with some "light" machine learning techniques. Currently the third generation is data-driven with deep neural networks. Lately combining reinforcement learning with the third generation started to have a contribution to the development of the dialogue systems [91].

The typical pipeline of task oriented dialogue systems consists of many modules [92], [93]:

a) natural language understanding (NLU):
assigns the user responses to semantic representation.

b) the dialog state tracker (DST)
accumulates the user input of the turn along with the dialogue history and determines the current state of the dialogue

c) the dialogue policy
selects the next action based on the dialogue state.

d) natural language generation
Natural language generation is the process of constructing computer systems that generate understandable natural language texts from underlying representations of information [94].

There are two approaches for designing dialogue systems: the modular approach and End to End approach. The difference between them is that the later approach replaces some the independent modules in figure 7 with a single end-to-end model [93].

Reinforcement learning is common in learning dialogue policy in the modular approach [95]–[97]. In end to end approach, using reinforcement learning contributes to enhancing the dialogue system performance by optimizing the system response [93], [98]–[101].

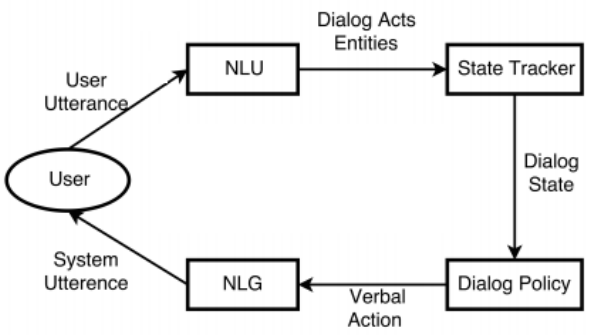

Fig. 7 Typical pipeline of task oriented dialogue systems [93]

Alexa Prize competition is a 2.5-million-dollar competition where university teams challenge to build conservation bots that interact with users via text and sound [102]. Although the winner in 2017 competition (Sounding Board by University of Washington) did not use reinforcement learning [103], MILABOT which achieved the highest average user score of 3.15 out of 5 in the competition has utilized reinforcement learning. MILABOT was developed in University of Montreal and it consists of a collection of 22 response models including natural language generation, template-based models, bag-of-words retrieval models, and sequence-to-sequence neural network. When a set of possible responses is generated, reinforcement learning is used to find the optimal policy to select between the candidate responses. The optimal policy

need to balance between long term and short term user satisfaction [104].

A Sample of works that applied reinforcement learning to natural language processing includes DeepMind work of using reinforcement learning to compute representations of natural language sentences by learning tree-structured neural networks [105].

VI. Prospective and Challenges

RL has vast unexplored territories as well as many unanswered questions in the explored ones; some of the typical challenges in RL include: Evaluative feedback, non-stationarity, and delayed rewards [9].

Achieving general AI requires multi-task learning [106] where an agent is able to perform many different type of tasks, rather than specializing in few similar tasks. Multi-task learning is one of the challenges that RL aims to solve.

The computational power plays significant role in driving progress in reinforcement learning research. In addition to developing efficient algorithms, parallel processing by modern hardware leads to increasing throughput. For example, the final version of AlphaGo used 40 search threads, 48 CPUs, and 8 GPUs. [14]. However, computational power may hinder some techniques such as tabular methods, exhaustive search, and Monte Carlo.

Yann LeCun [71] pointed that pure model-free RL requires many trials to learn anything. While trial and error is acceptable in simulated environments, it may not be the case in the real world. Hence, we need model based (see [1]) reinforcement learning with ability to predict by simulating the world which would reach a compact policy with no run-time optimization. Nonetheless, Simulator and model based reinforcement learning may be a clue for safe exploration, effective exploration remains a challenge; considering the chance that an agent succeeds in assembling a car from scratch with random behavior, the odds seem not in the agents' favor.

VII. Conclusion

Reinforcement learning is a huge topic, with a long history, a wide range of applications, an elegant theoretical core, distinguished successes, novel algorithms, and many open problems.

More research in artificial intelligence is looking for general principles of learning, decision making, and search in conjunction with integrating a wide range of domain knowledge. Research on reinforcement learning is a driving force toward easier and less general principles of artificial intelligence [9].

References

[1] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Pearson, 2009.

[2] A. M. Turing, "Computing machinery and intelligence," in *Parsing the Turing Test: Philosophical and Methodological Issues in the Quest for the Thinking Computer*, 1950, pp. 23–65.

[3] S. M. Shieber, *The Turing Test: Verbal Behavior as the Hallmark of Intelligence*. Mit Press, 2004.

[4] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[5] A. Goodfellow, Ian, Bengio, Yoshua, Courville, *Deep Learning*, 1st ed. Cambridge, MA: Mit Press, 2016.

[6] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips, "GPU Computing," *Proc. IEEE*, vol. 96, pp. 879–899, 2008.

[7] K. Sato, C. Young, and D. Patterson, "An in-depth look at Google's first Tensor Processing Unit (TPU)." [Online]. Available: https://cloud.google.com/blog/big-data/2017/05/an-in-depth-look-at-googles-first-tensor-processing-unit-tpu. [Accessed: 25-Dec-2017].

[8] R. R. Schaller, "Moore's law: past, present and future," *IEEE Spectr.*, vol. 34, no. 6, pp. 52–59, 1997.

[9] R. S. Sutton and A. G. Barto, *Reinforcement learning : an introduction*, 2nd ed. Cambridge, MA: Mit Press, 2017.

[10] D. Silver, "Deep reinforcement learning," in *International Conference on Machine Learning (ICML)*, 2016.

[11] V. Mnih *et al.*, "Playing Atari with Deep Reinforcement Learning," in *Conference on Neural Information Processing Systems*, 2013, pp. 1–9.

[12] DeepMind, "About Us." [Online]. Available: https://deepmind.com/about/. [Accessed: 14-Jan-2018].

[13] G. Tesauro, "TD-Gammon, a Self-Teaching Backgammon Program, Achieves Master-Level Play," *Neural Comput.*, vol. 6, no. 2, pp. 215–219, 1994.

[14] D. Silver *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[15] D. Silver *et al.*, "Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm," London, 2017.

[16] P. Abbeel, A. Coates, M. Quigley, and A. Y. Ng, "An application of reinforcement learning to aerobatic helicopter flight," *Education*, vol. 19, p. 1, 2007.

[17] A. Y. Ng *et al.*, "Autonomous inverted helicopter flight via reinforcement learning," *Springer Tracts Adv. Robot.*, vol. 21, pp. 363–372, 2006.

[18] G. Tesauro, D. C. Gondek, J. Lenchner, J. Fan, and J. M. Prager, "Analysis of Watson's strategies for playing Jeopardy!," *J. Artif. Intell. Res.*, vol. 47, pp. 205–251, 2013.

[19] R. Bellman, "On the Theory of Dynamic Programming," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 38, no. 8, pp. 716–719, 1952.

[20] R. E. Bellman and S. E. Dreyfus, "Applied Dynamic Programming," *Ann. Math. Stat.*, vol. 33, no. 2, pp. 719–726, 1962.

[21] C. J. C. H. Watkins and P. Dayan, “Q-learning,” *Mach. Learn.*, vol. 8, no. 3–4, pp. 279–292, 1992.

[22] G. A. Rummery and M. Niranjan, “On-line Q-learning using Connectionist Systems,” University of Cambridge, 1994.

[23] P. J. Werbos, “Building and Understanding Adaptive Systems: A Statistical/Numerical Approach to Factory Automation and Brain Research,” *IEEE Trans. Syst. Man Cybern.*, vol. 17, no. 1, pp. 7–20, 1987.

[24] A. G. Barto, R. S. Sutton, and C. W. Anderson, “Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems,” *IEEE Trans. Syst. Man Cybern.*, vol. SMC-13, no. 5, pp. 834–846, 1983.

[25] C. J. C. H. Watkins, “Learning from delayed rewards,” University of Cambridge, 1989.

[26] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, “A Brief Survey of Deep Reinforcement Learning,” *IEEE Signal Process. Mag. Spec. Issue Deep Learn. Image Underst.*, pp. 1–14, 2017.

[27] P. Dayan, “The Convergence of TD(λ) for General λ,” *Mach. Learn.*, vol. 8, no. 3, pp. 341–362, 1992.

[28] J. N. Tsitsiklis and B. Van Roy, “An analysis of temporal-difference learning with function approximation,” *IEEE Trans. Automat. Contr.*, vol. 42, no. 5, pp. 674–690, 1997.

[29] R. S. Sutton, “Learning to Predict by the Methods of Temporal Differences,” *Mach. Learn.*, vol. 3, no. 1, pp. 9–44, 1988.

[30] G. J. Gordon, “Stable Function Approximation in Dynamic Programming,” *Proc. 12th Int. Conf. Mach. Learn.*, no. January, pp. 261–268, 1995.

[31] L. Lin, “Reinforcement Learning for Robots Using Neural Networks,” *Report, C.*, pp. 1–155, 1993.

[32] H. Van Hasselt, A. C. Group, and C. Wiskunde, “Double Q-learning,” *Nips*, pp. 1–9, 2010.

[33] H. Yu, “Convergence of Least Squares Temporal Difference Methods Under General Conditions,” in *International Conference on Machine Learning*, 2010, pp. 1207–1214.

[34] A. R. Mahmood and R. S. Sutton, “Off-policy learning based on weighted importance sampling with linear computational complexity,” *Proc. Thirty-First Conf. Uncertain. Artif. Intell. {UAI} 2015, July 12-16, 2015, Amsterdam, Netherlands*, pp. 552–561, 2015.

[35] H. R. Maei, “Gradient Temporal-Difference Learning Algorithms,” *Mach. Learn.*, 2011.

[36] S. J. Bradtke and A. G. Barto, “Linear Least-Squares algorithms for temporal difference learning,” *Mach. Learn.*, vol. 22, no. 1–3, pp. 33–57, 1996.

[37] H. R. Maei and R. S. Sutton, “GQ( ): A general gradient algorithm for temporal-difference prediction learning with eligibility traces,” in *Proceedings of the 3d Conference on Artificial General Intelligence (AGI-10)*, 2010.

[38] P. Abbeel and J. Schulman, “Deep Reinforcement Learning through Policy Optimization,” in *Neural Information Processing Systems*, 2016.

[39] M. Minsky, “Steps toward Artificial Intelligence,” *Proc. IRE*, vol. 49, no. 1, pp. 8–30, 1961.

[40] J. A. Nelder and R. Mead, “A Simplex Method for Function Minimization,” *Comput. J.*, vol. 7, no. 4, pp. 308–313, 1965.

[41] F. Gomez, J. Koutník, and J. Schmidhuber, “Compressed network complexity search,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2012, vol. 7491 LNCS, no. PART 1, pp. 316–326.

[42] A. Fraser, “Simulation of Genetic Systems by Automatic Digital Computers I. Introduction,” *Aust. J. Biol. Sci.*, vol. 10, no. 4, p. 484, 1957.

[43] M. P. Deisenroth, “A Survey on Policy Search for Robotics,” *Found. Trends Robot.*, vol. 2, no. 1–2, pp. 1–142, 2011.

[44] T. Salimans, J. Ho, X. Chen, and I. Sutskever, “Evolution Strategies as a Scalable Alternative to Reinforcement Learning,” 2017.

[45] R. J. Willia, “Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning,” *Mach. Learn.*, vol. 8, no. 3, pp. 229–256, 1992.

[46] D. Wierstra, A. Förster, J. Peters, and J. Schmidhuber, “Recurrent policy gradients,” *Log. J. IGPL*, vol. 18, no. 5, pp. 620–634, 2009.

[47] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, “Trust Region Policy Optimization,” in *International Conference on Machine Learning*, 2015.

[48] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, “High-Dimensional Continuous Control using Generalized Advantage Estimation,” in *International Conference on Learning Representations*, 2016.

[49] G. Cuccu, M. Luciw, J. Schmidhuber, and F. Gomez, “Intrinsically motivated neuroevolution for vision-based reinforcement learning,” in *IEEE International Conference on Development and Learning, ICDL*, 2011.

[50] F. Gomez and J. Schmidhuber, “Evolving Modular Fast-Weight Networks for Control,” in *ICANN*, 2005.

[51] J. Koutník, G. Cuccu, J. Schmidhuber, and F. Gomez, “Evolving large-scale neural networks for vision-based reinforcement learning,” in *Proceeding of the fifteenth annual conference on Genetic and evolutionary computation conference - GECCO ’13*, 2013, p. 1061.

[52] Y. Lecun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553. pp. 436–444, 2015.

[53] R. S. Sutton, D. Mcallester, S. Singh, and Y. Mansour, “Policy Gradient Methods for Reinforcement Learning with Function Approximation,” *Adv. Neural Inf. Process. Syst. 12*, pp. 1057–1063, 1999.

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

[55] V. R. Konda and J. N. Tsitsiklis, “Actor-Critic Algorithms,” *Control Optim*, vol. 42, no. 4, pp. 1143–1166, 2003.

[56] V. Mnih *et al.*, “Asynchronous methods for deep reinforcement learning,” in *International Conference of Machine Learning*, 2016.

[57] S. Gu, T. Lillicrap, Z. Ghahramani, R. E. Turner, and S. Levine, “Q-Prop: Sample-Efficient Policy Gradient with an Off-Policy Critic,” in *International Conference on Learning Representations*, 2017.

[58] V. Mnih *et al.*, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[59] L. J. Lin, “Self-Improving Reactive Agents Based on Reinforcement Learning, Planning and Teaching,” *Mach. Learn.*, vol. 8, no. 3, pp.

293–321, 1992.

[60] T. P. Lillicrap *et al.*, “Continuous Control with Deep Reinforcement Learning,” in *International Conference on Learning Representations*, 2016.

[61] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, “Continuous Deep Q-Learning with Model-Based Acceleration,” in *International Conference on Learning Representations*, 2016.

[62] T. P. Lillicrap *et al.*, “Continuous control with deep reinforcement learning,” in *International Conference on Learning Representations (ICLR)*, 2016, pp. 1–14.

[63] Z. Wang *et al.*, “Sample Efficient Actor-Critic with Experience Replay,” in *International Conference on Learning Representations*, 2017.

[64] V. Mnih *et al.*, “Prioritized Experience Replay,” *Int. Conf. Mach. Learn.*, vol. 4, no. 7540, p. 14, 2015.

[65] H. van Seijen and R. S. Sutton, “A deeper look at planning as learning from replay,” *Proc. 32nd Int. Conf. Mach. Learn.*, vol. 37, pp. 2314–2322, 2015.

[66] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, “Planning and acting in partially observable stochastic domains,” *Artif. Intell.*, vol. 101, no. 1–2, pp. 99–134, 1998.

[67] Medsker L.C. and L. R. Jains, *Recurrent neural networks design and applications*. CRC press, 2001.

[68] M. Boden, “A guide to recurrent neural networks and backpropagation,” *Electr. Eng.*, no. 2, pp. 1–10, 2001.

[69] N. Heess, J. Hunt, T. Lillicrap, and D. Silver, “Memory-Based Control with Recurrent Neural Networks,” in *Conference on Neural Information Processing Systems*, 2015.

[70] M. Hausknecht and P. Stone, “Deep Recurrent Q-Learning for Partially Observable MDPs,” in *AAAI Fall Symposium Series*, 2017.

[71] Y. LeCun, “How Could Machines Learn as Efficiently as Animals and Humans?,” 2017.

[72] S. Ontanon, G. Synnaeve, A. Uriarte, F. Richoux, D. Churchill, and M. Preuss, “A survey of real-time strategy game AI research and competition in starcraft,” *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 5, no. 4. pp. 293–311, 2013.

[73] Y. Wu and Y. Tian, “Training Agent for First-Person Shooter Game with Actor-Critic Curriculum Learning,” *Int. Conf. Learn. Represent.*, pp. 1–9, 2017.

[74] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaskowski, “ViZDoom: A Doom-based AI research platform for visual reinforcement learning,” in *IEEE Conference on Computatonal Intelligence and Games, CIG*, 2017.

[75] E. Perot, M. Jaritz, M. Toromanoff, and R. De Charette, “End-to-End Driving in a Realistic Racing Game with Deep Reinforcement Learning,” in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2017, vol. 2017–July, pp. 474–475.

[76] N. Justesen, P. Bontrager, J. Togelius, and S. Risi, “Deep Learning for Video Game Playing,” *arXiv*, pp. 1–17, 2017.

[77] B. Zoph and Q. V. Le, “Neural architecture search with reinforcement learning,” in *the International Conference on Learning Representations (ICLR)*, 2017.

[78] I. Bello, B. Zoph, V. Vasudevan, and Q. V. Le, “Neural optimizer search with reinforcement learning,” in *the 34th International Conference on Machine Learning (ICML)*, 2017.

[79] T. Tieleman and G. Hinton, “Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude.,” *COURSERA Neural Networks Mach. Learn.*, 2012.

[80] D. P. Kingma and J. L. Ba, “Adam: a Method for Stochastic Optimization,” *Int. Conf. Learn. Represent.*, pp. 1–15, 2015.

[81] A. L. C. Bazzan and F. Klügl, *Introduction to Intelligent Systems in Traffic and Transportation*, vol. 7, no. 3. 2013.

[82] S. El-Tantawy, B. Abdulhai, and H. Abdelgawad, “Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (marlin-atsc): Methodology and large-scale application on downtown toronto,” *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 3, pp. 1140–1150, 2013.

[83] L. Busoniu, R. Babuska, B. De Schutter, and B. De Schutter, “A comprehensive survey of multiagent reinforcement learning,” *Syst. Man, Cybern. Part C Appl. Rev.*, vol. 38, no. 2, pp. 156–172, 2008.

[84] E. Van Der Pol and F. A. Oliehoek, “Coordinated Deep Reinforcement Learners for Traffic Light Control,” *NIPS’16 Work. Learn. Inference Control Multi-Agent Syst.*, no. Nips, 2016.

[85] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba, “Sequence level training with recurrent neural networks,” in *the International Conference on Learning Representations (ICLR)*, 2016.

[86] D. Bahdanau *et al.*, “An actor-critic algorithm for sequence prediction,” in *the International Conference on Learning Representations (ICLR)*, 2017.

[87] A. Sokolov, J. Kreutzer, C. Lo, and S. Riezler, “Learning Structured Predictors from Bandit Feedback for Interactive NLP,” in *Association for Computational Linguistic (ACL)*, 2016, no. 2012, pp. 1610–1620.

[88] A. C. Grissom II, J. Boyd-Graber, H. He, J. Morgan, and H. Daume III, “Don’t Until the Final Verb Wait: Reinforcement Learning for Simultaneous Machine Translation,” *Empir. Methods Nat. Lang. Process.*, no. Section 3, pp. 1342–1352, 2014.

[89] Y. Wu *et al.*, “Google’s Neural Machine Translation System: Bridging the Gap between Human and Machine Translation,” *ArXiv e-prints*, pp. 1–23, 2016.

[90] D. Jurafsky and J. H. Martin, *Speech and Language Processing, An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition (Draft)*. New Jersy: Practice-Hall, 2008.

[91] L. Deng, “Three Generations of Spoken Dialogue Systems (Bots),” in *AI Frontiers*, 2017.

[92] S. Young, “Using pomdps for dialog management,” in *Spoken Language Technology Workshop, IEEE*, 2006, pp. 8–13.

[93] T. Zhao and M. Eskenazi, “Towards End-to-End Learning for Dialog State Tracking and Management using Deep Reinforcement Learning,” in *the Annual SIGdial Meeting on Discourse and Dialogue (SIGDIAL)*, 2016.

[94] Z. Xie, “Neural Text Generation: A Practical Guide.” 2017.

[95] J. D. Williams and S. Young, “Partially observable Markov decision processes for spoken dialog systems,” *Comput. Speech Lang.*, vol. 21, no. 2, pp. 393–422, 2007.

[96] S. Lee and M. Eskenazi, “POMDP-based Let’s Go system for spoken dialog challenge,” in *2012 IEEE Workshop on Spoken Language Technology, SLT 2012 - Proceedings*, 2012, pp. 61–66.

[97] K. Georgila and D. Traum, "Reinforcement learning of argumentation dialogue policies in negotiation," in *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, 2011, pp. 2073–2076.

[98] J. Li, W. Monroe, A. Ritter, and D. Jurafsky, "Deep Reinforcement Learning for Dialogue Generation," in *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2016.

[99] T.-H. Wen *et al.*, "A network-based end-to-end trainable task-oriented dialogue system," *arXiv:1604.04562*, 2016.

[100] I. V. Serban, A. Sordoni, Y. Bengio, A. Courville, and J. Pineau, "Building End-To-End Dialogue Systems Using Generative Hierarchical Neural Network Models," *Aaai*, p. 8, 2016.

[101] I. Sutskever, O. Vinyals, and Q. V Le, "Sequence to sequence learning with neural networks," *Adv. Neural Inf. Process. Syst.*, pp. 3104–3112, 2014.

[102] A. Ram et al., "Conversational AI: The Science Behind the Alexa Prize," in *Alexa Prize Proceedings https://developer.amazon.com/alexaprize/proceedings*, 2018.

[103] H. Fang et al., "Sounding Board – University of Washington's Alexa Prize Submission," in *Alexa Prize Competition*, 2017.

[104] I. V. Serban *et al.*, "A Deep Reinforcement Learning Chatbot," *Prepr. arXiv1801.06700v1, 20 Jan 2018*.

[105] D. Yogatama, P. Blunsom, C. Dyer, E. Grefenstette, and W. Ling, "Learning to compose words into sentences with reinforcement learning," in *International Conference on Learning Representations (ICLR)*, 2017.

[106] R. Caruana, "Multitask Learning," *Mach. Learn.*, vol. 28, no. 1, pp. 41–75, 1997.