You Think, I Guess

Prasanth Murali, Shweta Oak
Northeastern University
Boston, MA
murali.pr, oak.s @ husky.neu.edu

Abstract

Recent advancements in dialogue generation using deep reinforcement offer great promise for generating responses for conversational agents. In this paper, we explore and suggest one such algorithm that can be adopted to develop a system that can have task oriented conversations such as booking a reservation, booking flights, or play a game and so on. We are particularly interested in leveraging this to simulate a 20 Questions Game [1] playing agent but from our literature understood that these tend to be short-sighted, while ignoring their influence on future outcomes. Traditional supervised approaches, though work well, require a substantial knowledge base in the form of training set and are also short sighted and do not generate content having future contexts. Modeling the future dialogue is crucial to generating content especially in the context of this game, and thus calls for a need to a need to adopt Reinforcement Learning based approaches. In this paper, we show how to integrate these goals, applying deep reinforcement learning to model future reward in a dialogue system. The model simulates dialogues between a virtual agent and a human. We evaluate against state of the art work in this space on accuracy and compare this baseline against the performance of the algorithm for the 20 Questions game. This work is also step towards learning a neural ensembled supervised and reinforcement learning based conversational models for the long-term success of dialogues.

Introduction

20 Questions is a popular game involving two parties, namely, an answerer who picks an object but does not reveal it and the questioner who looks to guess that object by asking a series of questions to the questioner. The answers to the questions can be answered with a 'yes' or 'no'. It encourages use of deductive reasoning and creativity [1] and careful selection of questions can help narrow down the name of the object better. In this work, we explore a computer questioner that looks to guess name of the object that the human answerer thinks of.

Thus, it is evident that such games follow a decision tree where there are multiple paths that the agent can take to arrive at the conclusion. Some of these paths are more rewarding than the others, while some can be detrimental to successful guessing by the agent. For instance, asking the question, 'is the person you are thinking of from a technology, political or sports person' would help cover a range of topics with the same question whereas asking 'is this person alive or dead' doesn't really help since every person is either alive or dead. This reward based approach can be modelled into a reinforcement learning problem where the agent can make use of the feedback from the game to decide the optimal policy (strategy) in asking questions to guess the potential name of the object.

Typical dialog systems are built with natural language understanding, state - action selection, and language generation built on top of one another.In this case, complexity arises since it is unclear to define what dialog state is, what the corresponding is and what information to remember in going forward.

Thus, in this paper, recurrent neural networks (RNNs) are trained on dialogs directly. Since RNN offers a latent representation of state, this eliminates the need for an explicit state information generation. In addition, literature shows that a similar performance with considerably less training data is achieved using RNNs. However, to ensure that end to end

trainability for these models is maintained, this neural network is trained using Reinforcement Learning techniques.

We use the REINFORCE [2][3] algorithms to train the policy network that selects one of the questions to choose at the particular state. The agent uses a probability distribution to obtain a confidence for the target object, and updates the answerer responses. Using the estimated policy network $\pi\theta(a|s)$, during every time step, the agent takes this confidence vector and outputs a probability distribution to select the subsequent question. The neural network estimates appropriate reward at every step to train the RL model since we do not have information about immediate reward for every selected question. The policy $\pi\theta(a|st)$ is parametrized to identify probability distribution over all available actions: $\pi\theta(a|st) = P[a|st; \theta]$ and is updated accordingly to maximize returns from the environment. This enables the network to learn a stochastic policy instead of a greedy one like in DQN.

Background and Related Work

Recently, there have been attempts to use reinforcement learning to develop such information seeking agents [4]. Bachman et al., in Towards Information Seeking Goals (2016), explore methods for artificial curiosity, intrinsically-motivated exploration, and other more precise goals. The authors describe a Generalized Advantage Estimation model (GAE) to identify meaningful shifts in perspectives regarding attention of information. They propose that training their models with this objective encourages them to maximize the rate at which they gather information about the environment.

Initialize replay memory $\mathcal{D}_2$ to capacity N_2
Initialize policy net π with random weights θ
Initialize value net $\mathcal{V}$ with random weights η
Initialize RewardNet with random weights σ
for *episode* $i \leftarrow 1$ ***to*** Z **do**
  Rollout, collect rewards, and save the history in S_2 (4-10 in Algo. 1)
  for (s_t, a_t, r_{t+1}) ***in*** S_2 **do**
    $G_t \leftarrow \sum_{k=0}^{T} \gamma^k r_{t+k+1}$
    Update **RewardNet** (13-17 in Algo. 1)
    Store (s_t, a_t, G_t) in $\mathcal{D}_2$
    if $len(\mathcal{D}_2) > K_2$ **then**
      Sample mini-batch from $\mathcal{D}_2$
      Update η with loss L_3 in Eq. 8
      Update θ with loss L_2 in Eq. 7

Figure 1: Hu et al., Playing 20 Question Game with Policy-Based Reinforcement Learning

Hu et al.[5], propose an integrated framework (Figure 1) combining techniques to learn optimal policy from continuous interaction with the users and a neural network to optimize on the reward function to estimate the more informative reward. In this paper, the authors propose a policy-based RL method to solve the question selection problem in the 20 Questions Game. The develop a Reward Network to estimate long term returns better and make it more informative. This makes the agent robust to noise while also eliminating a substantial knowledge base of information for the 20 questions game.

Past work in this space also describes supervised learning with reinforcement learning [6] (Su et al., 2016) and feed forward neural networks [7] (Wen at al., 2016). In these works, the policy has not been *recurrent* – i.e., the policy depends on the state tracker to summarize observable dialog history into state features, which requires design and specialized labeling.

However, RNN also requires an automatic inference of state representation. If there is context which is not apparent in the text in the dialog, such as database status, this can be encoded as a context feature to the RNN.

Thus, more recent line of work applies recurrent neural networks (RNNs) to learn mapping from

an observable dialog history directly to a sequence of output words[8][9][10]. These systems can then be applied in various scenarios ranging from adding "API call" actions, enumerating database output as a sequence of tokens[11][12][13][14][15], then learning an RNN using Memory Networks, gated memory networks, query reduction networks, and copy-augmented networks[16].

As previously mentioned, Williams, in Simple Statistical Gradient Following Algorithms for Connectionist Reinforcement Learning propose algorithms that integrate well with traditional back propagation algorithms.[2] This is important since our architecture uses supervised as well as reinforcement learning in tandem with each other. In this paper, the author proposes a class of weighted algorithms referred to as REINFORCE, that make weight adjustments in the direction of gradient for both short term and long term goals without explicitly computing gradient estimates or even storing information about what information can be computed. However, this presents a lack of convergence to the right optimum for these algorithms thus motivating the need for a more robust integration.

Framptoon et al.[17], in Reinforcement Learning Of Dialogue Strategies Using The User's Last Dialogue Act, describe a dialog system with high level contextual information tracking (it is the user's last utterance in this case). The proposed approach fights the curse of dimensionality that comes up tracking the additional information state while also provides method to validate the existence of the additional information in this context.

In "Sample-Efficient Batch Reinforcement Learning for Dialogue Management Optimization", Pietquin et al[18], explore the possibility of using a set of approximate dynamic programming algorithms for policy optimization in Spoken Dialogue Systems. This is an off policy data driven approach to learning dialogue strategies, that determining the functioning of the dialogue management module
from simple data.

Walker, in An Application of Reinforcement Learning to Dialogue Strategy Selection in a Spoken Dialogue System for Email [19], proposes a Q-learning based approach that converges to an optimal dialogue strategy using its experience of dealing with human users. In Deep Reinforcement Learning for Dialogue Generation, Li et al.[20], propose a model that is SEQ2SEQ for optimal choice of dialogue. This approach generates utterances that maximizes future rewards, thus ensuring convergence in that context.

Methods : Algorithm

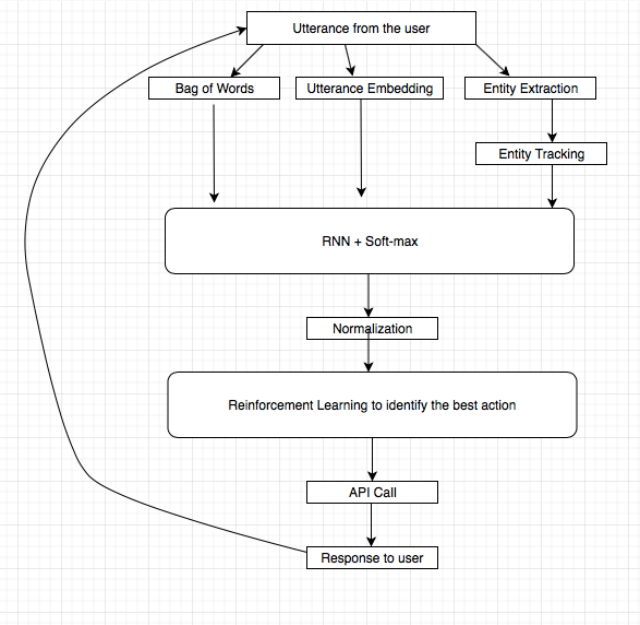

Figure 2: Flowchart of the algorithm adopted in this paper

We are adopting an architecture similar to the one proposed by Williams et al., in Hybrid Code Networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning[16]. As shown in the flowchart, the major components of the proposed architecture are the RNN, domain-specific software; domain-specific action templates; and a conventional entity extraction module. Each action template can either be a textual communicative action or an API call.

We will now walkthrough the entire process. The first step is when the user gives a text (Say “Robert”). From this text, a bag of words is formed. This is followed by other text processing techniques such as forming an utterance embedding (“syllable extraction in Robert, etc”). Now, an entity extraction model identifies Robert as a name. The extracted entity is grounded for future references and relevant API calls are made to get contextual information (In our case, suppose the human asks, “Is this person alive?”, and the algorithm could get the appropriate answer by making the right calls.

These features are then used to form the feature vector. This vector is passed to a RNN layer and the computed hidden state is passed to a dense soft-max layer, giving a vector of action space dimensionality.[21][22]. Here, the action mask is applied and the result is normalized into a probability distribution. Reinforcement learning is then applied on this distribution, where an action is *sampled* from the distribution; when RL is not active, the best action should be chosen, and so the action with the *highest probability* is always selected.

At this point, the selected action is passed to an entity output module that maps the entity to the exact action (for instance, “politician” is mapped to “Barack Obama”, “sports person” is mapped to “Tom Brady” and so on. A corresponding API call is made at the final step now, to retrieve relevant information and return features that can be added as a feature vector in the next time step to complete the cycle.

To think of this problem in terms of state and action space, we can think of it as a 20 dimensional binary search space and the goal as to reduce the dimensionality of this space. Thus, the reward will be +1 on reaching the finding the answer and 0 otherwise.

For instance, if a human thinks of "Barrack Obama", the algorithm first asks a random question -- "is the person female" to which the answer is "no", so the algorithm learns to not as any questions about the female personalities. Secondly, if it is asked "Is the person an Asian" -- answer is no, so the algorithm excludes all female and Asian personalities. This approach enables the algorithm to traverse the state space by keeping responses to earlier questions in memory and pick the ‘subsequent best question’ at every step.

Results

We adopted a similar evaluation procedure as recommended by Williams at al.[16]. We ensured that the architecture followed the recommendation to ensure a controlled testing environment. With references from this work, we will now walkthrough the adopted evaluation procedure against some baseline models. Thus, the first part of our evaluation against baseline models is for the restaurant reservation task whereas the second task is to test our model’s performance on the 20 Questions Game.

The ‘bAbI dialog’ data set [23] includes two end-to-end dialog learning tasks, in the restaurant domain consisting of synthetic, simulated dialog data, with highly regular user behavior and constrained vocabulary. Dialogs include a database access action which retrieves relevant restaurants from a database, with results included in the dialog transcript.

We used string matching with a pre defined set of entity names for entity extraction, followed by context update where a newly detected entity from user input overwrites a previous value in the database. For example, if the quality “poor” is recognized in the first turn, it is retained as quality=poor. If “great” is then recognized in the third turn, it over-writes “poor” so the code now holds quality=great.

Then, all system actions were converted to templates of the form <name of restaurant/>, <location of restaurant/> and <price range of restaurant/>.

We then trained an RNN on the training set (128 hidden units and 12 epochs), employing the domain-specific software described above.

Word2vec was used to form the utterance embeddings[24]. We selected an LSTM for the recurrent layer [21], with the AdaDelta optimizer [25].

We next examined learning curves, training with increasing numbers of dialogs. To guard against bias in the ordering of the training set, we averaged over 5 runs, randomly permuting the order of the training dialogs in each run. Results are in Figure 3. Compared to other similar approaches (Bordes and Weston, 2016)[11].

Method	Restaurant task		20 questions task	
	Turn Acc	Dialog Acc	Turn Acc	Dialog Acc
Bordes and Weston 2016	77.7%	0.0%	41.1%	0.0%
HCN	99.6%	97.3%	93.4%	86.5%

Table 1: Comparison of results between our Model and a baseline for two tasks

We also examined training curves for the reward function for both the restaurant task as well as 20 Questions game and noticed the convergence criteria as well as test accuracy for the same. The observed smoothened learning curve for the 20 questions game is as shown.

Thus, these tasks are the best public benchmark we are aware of, as mentioned in the paper, and matching performance of past models using an order of magnitude less data (200 vs. 1618 dialogs), shows that the approach is practical settings where collecting realistic dialogs for a new domain can be expensive.

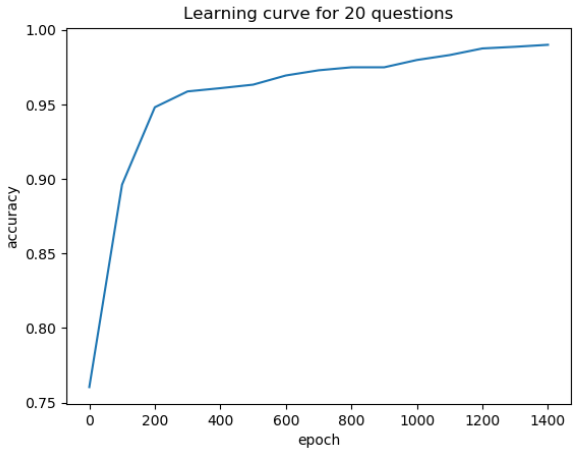

Figure 3: Learning curve for the 20 Questions

Conclusion

Thus, this work shows that the suggested framework learns the optimal policy of question selection for 20 questions game as well as other tasks as shown and expect to compare it with baseline entropy based models developed using an existing knowledge base. It is more versatile than building a knowledge base from scratch since the model is reusable across contexts, without requiring much data to learn from.

However, we would like to explore the agent responses to dealing with noisy answers (such as a wrong answer), which is often the case in a real world
situation. A stronger evaluation against the traditional decision tree based approaches for dialogue systems is also required going forward. We are also interested in analysing the efficacy of such a network to noisy answers (where for instance, the user says a wrong answer to a particular question)**.**

References

1.https://en.wikipedia.org/wiki/Twenty_Questions
2. Ronald J Williams. 1992. Simple statistical gradient following algorithms for connectionist reinforcement learning. In Reinforcement Learning, pages 5–32. Springer.
3. Zelinka, Using reinforcement learning to learn how to play text-based games, 2017.
4. Bachman, Philip, Alessandro Sordoni, and Adam Trischler. "Towards information-seeking agents." arXiv preprint arXiv:1612.02605 (2016).
5. Hu, Huang, et al. "Playing 20 Question Game with Policy-Based Reinforcement Learning." arXiv preprint arXiv:1808.07645 (2018).
6. Pei-Hao Su, Milica Gasic, Nikola Mrksic, Lina Rojas- Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Continuously learning neural dialogue management. *arxiv*.
7. Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, David Vandyke, and Steve Young. 2016. A network-

based end-to-end trainable task-oriented dialogue system. *arXiv preprint arXiv:1604.04562*.
8.Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Meg Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversa- tional responses. In *Proceedings of NAACL-HLT*.
9. Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Proceedings of ACL-IJCNLP*, pages 1577–1586.
10. Oriol Vinyals and Quoc Le. 2015. A neural conversa- tional model. In *Proceedings of ICML Deep Learning Workshop*.
11. Antoine Bordes and Jason Weston. 2016. Learn- ing end-to-end goal-oriented dialog. *CoRR*abs/1605.07683/http://arxiv.org/abs/1605.07683
12. Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory net- works. In *Proc Advances in Neural Information Processing Systems (NIPS), Montreal, Canada*.
13. Fei Liu and Julien Perez. 2016. Gated end-to- end memory networks. *CoRR* abs/1610.04211. http://arxiv.org/abs/1610.04211.
14. Min Joon Seo, Hannaneh Hajishirzi, and Ali Farhadi. 2016. Query-regression networks for machine comprehension. *CoRR* abs/1606.04582. http://arxiv.org/abs/1606.04582.
15. Mihail Eric and Christopher D Manning. 2017. A copy-augmented sequence-to-sequence ar- chitecture gives good performance on task- oriented dialogue. *CoRR* abs/1701.04024. https://arxiv.org/abs/1701.04024.
16. Williams, Jason D., Kavosh Asadi, and Geoffrey Zweig. "Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning." *arXiv preprint arXiv:1702.03274* (2017).
17. Frampton, Matthew, and Oliver Lemon. "Reinforcement learning of dialogue strategies using the user's last dialogue act." *IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems*. 2005.
18. Pietquin, Olivier, et al. "Sample-efficient batch reinforcement learning for dialogue management optimization." *ACM Transactions on Speech and Language Processing (TSLP)*7.3 (2011): 7.
19. Walker, Marilyn A. "An application of reinforcement learning to dialogue strategy selection in a spoken dialogue system for email." *Journal of Artificial Intelligence Research* 12 (2000): 387-416.
20. Li, Jiwei, et al. "Deep reinforcement learning for dialogue generation." *arXiv preprint arXiv:1606.01541* (2016).
21. Sepp Hochreiter and Jurgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9(8):1735–1780.
22. Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence model- ing. In *Proc NIPS 2014 Deep Learning and Repre- sentation Learning Workshop*.
23. https://research.fb.com/downloads/babi/
24. Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Cor- rado, and Jeff Dean. 2013. Distributed representa- tions of words and phrases and their compositional- ity. In *Proc Advances in Neural Information Pro- cessing Systems, Lake Tahoe, USA*. pages 3111– 3119.
25. Zeiler, Matthew D. "ADADELTA: an adaptive learning rate method." *arXiv preprint arXiv:1212.5701* (2012).

Appendix

The code to this project can be found at the following link to the Github repository:
https://github.com/oak11/20Questions-RL/tree/master/20Questions