Natural Language Generation with Adversarial-Free Imitation Learning

Stanford CS224N Custom Project

Anuj Nagpal
ICME, Stanford University
anujnag@stanford.edu

Abstract

Despite their impressive performance, current language generation models are prone to an autoregressive bias issue wherein once they start generating text that is slightly different from the training data, they continue generating even weirder text with lots of repetitions, leading to a feedback loop. A parallel can be drawn here with naive behavioral cloning approach [1] from reinforcement learning where once an agent trained on expert demonstrations reaches a state expert hasn't seen, the agent can keep drifting away from demonstrated states due to error accumulation. Inverse reinforcement learning [2] [3] [4] tries to fix this where the agent tries to learn a cost function under which expert trajectory is optimal and implicitly learns the high density states that helps it recover in case of out of distribution events. These methods have been applied for text [5] [6] but they require reward estimation in an explicit way or implicitly in the form of adversarial training which has been found to be unstable. Recent advancements in imitation learning learn a single mapping from observations and actions to expected cumulative rewards (Q value) without any discriminator based adversarial training [7]. We use this parallel in text generation by formulating it as a reinforcement learning problem and show that Q-learning might be a better alternative than maximizing the log-likelihood of next token as per expert demonstrations.

1 Key Information

This project was mentored by Manan Rai with Chris Cundy and Div Garg as external mentors from SAIL's Ermon Group.

2 Introduction

2.1 Language Generation at Present

Large language models like GPT [8] [9] are widely successful in their text generation capabilities across a wide variety of domains. But deep down, they are autoregressive models in nature where they maximize the likelihood of next token given the previous tokena:

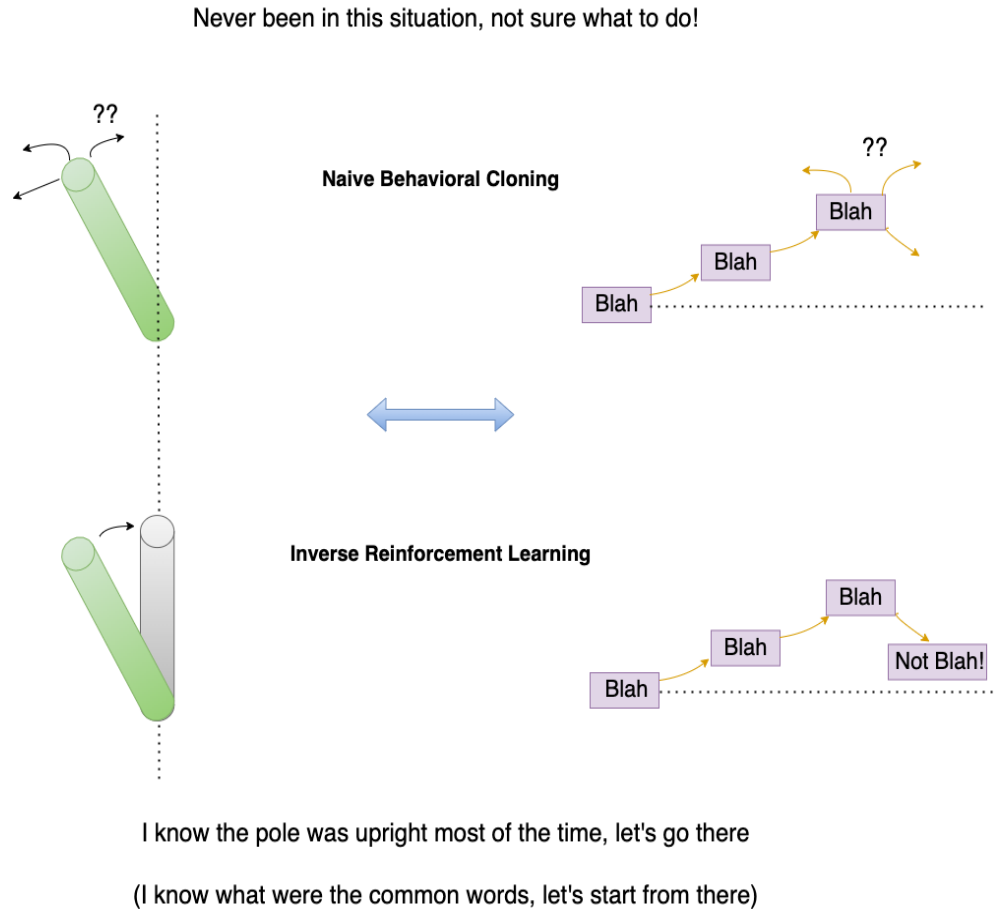
$$p(x) = \prod_{i=1}^{n} p(t_n|t_1, t_2, ..., t_{n-1}) \tag{1}$$

This makes them prone to autoregressive bias issue wherein once they start generating data that is different from expert demonstrations, that acts as a feedback loop for them to produce weirder data repeatedly. A simple test for this can be done by giving 'Blah Blah Blah' as a prompt to GPT-2 for text generation and the model just outputs 'Blah' for the entire sequence.

We observe that this is very similar to naive behavioral cloning from reinforcement learning (RL) [1] where an agent tries to learn a mapping from its current state to next action from expert demonstrations in a supervised fashion (also known as policy in RL literature). Naive behavioral cloning has been known to suffer from high variance since once it encounters a state expert hasn't seen, agent can drift further away to unseen states and keep accumulating error along the way. A possible cause for this is that agent is just trying to imitate the expert, being unaware of overall environment dynamics and how its actions impact the next observations it will get.

2.2 Inverse Reinforcement Learning

Many alternatives were found to incorporate dynamics in imitation learning, one of them being inverse reinforcement learning (IRL) wherein an agent tries to learn a cost function (reward function in RL literature) under which the expert's trajectory is optimal and then train a imitation policy from this cost function [2] [3] [4]. An advantage of inverse reinforcement learning is that the agent can implicitly learn the high density states so that it can better handle out of distribution events as demonstrated below with an example of Cartpole where an agent tries to balance a pole:



3 Related Work

There has been attempts at text generation with reinforcement learning which use rewards implicitly via adversarial training to learn an imitation policy [6] [5]. But adversarial training has been found to be unstable due to the min-max objective of learning reward and policy, and modeling them as separate functions. Div et al [7] suggested an alternative imitation learning technique which combines ideas from reinforcement learning and energy-based models [10] and tries to avoid this min-max objective and by directly learning a Q-function, which is a measure of expected cumulative rewards given present state and the next action. This method does not provide any explicit rewards or adversarial

training, and simply tries to recover the agent's policy from the learned Q-function ($\pi = \frac{1}{Z} \exp Q$). Rewards can be recovered from the Bellman's equation $r(s, a, s') = Q(s, a) - \gamma V^{\pi}(s')$ where V is the value function, a measure of expected cumulative rewards given the present state but this reward recovery is encoded implicitly in the objective function:

$$\max_{Q} \mathcal{J}(Q) = \\ \mathbb{E}_{(s,a,s') \sim (expert)}[Q(s, a) - \gamma \mathbb{E}_{s' \sim \mathcal{P}(\cdot|s,a)} V^{\pi}(s')] - \mathbb{E}_{(s,a,s') \sim (agent,expert)}[V^{\pi}(s) - \gamma V^{\pi}(s')] \tag{2}$$

Where (s, a, s') stands for (state, action, next state) and it is sampled from a replay buffer. We can also add an additional term in this objective to minimize the χ^2-divergence, which can act as a regularizer:

$$-\frac{1}{4\alpha} \mathbb{E}_{(s,a,s') \sim (agent,expert)}[(Q(s, a) - \gamma \mathbb{E}_{s' \sim \mathcal{P}(\cdot|s,a)} V^{\pi}(s'))^2] \tag{3}$$

4 Approach

4.1 Natural Language Generation as a Reinforcement Learning Problem

We define the key components of our reinforcement learning framework below:

- **State:** We need to define the notion of a state or observation space so that our agent can perceive its current situation for choosing next action. To represent a sequence of generated tokens so far, we use the BERT [CLS] token embedding [11] for a text sequence that can encode the current state as a 768-dimensional vector.
- **Action:** For text generation, this becomes choosing the next word in our sequence, and the action space will be the vocabulary size (30522 for bert-uncased model).
- **Episode:** An episode will be a complete sequence of tokens generated until the agent generates either a [SEP] token or it reaches the maximum episode length (chosen as 50).
- **Deterministic vs Stochastic:** We formulate this as a deterministic control problem where if an agent chooses an action, it is guaranteed to reach the desired next state unlike the stochastic case where there is some probability of reaching to different states.
- **Discrete vs Continuous:** We model this as a discrete control problem i.e. the action space can take discrete values, where each value corresponds to a specific token in vocabulary
- **Online vs Offline:** The agent tries to learn in an online fashion i.e. it keeps learning from its own experience along with expert demonstrations, while it is collecting more experience by generating text episodes.

4.2 Model Architecture

We define the key components of our architecture and design choices next:

- **Double DQN:** To learn the Q-function from expert experiences, we use a double deep-Q network [12] with 5-fully connected layers and ReLU activation in-between. Note that we are projecting 768-dimensional space state encodings into 30522 dimensional action space with 1024-2048 hidden units per layer and the best architecture may require higher number of hidden units with deeper layers but there's a big memory tradeoff.
- **Replay Buffer:** We maintain replay buffers for both expert and online agent experiences to sample (state, action, next state) experiences for estimating terms in our overall objective. Since each experience is of the order of a 768-dimensional vector, we save memory by flushing out old experiences from the buffers and re-filling it with new expert demonstrations and online experiences after each full episode.
- **Temperature:** High-dimensional action space also affects the diversity and model may try to pick some common words while still reducing the loss or perplexity, For getting better text sequences, we use a softmax temperature that can be made learnable to match a target entropy.

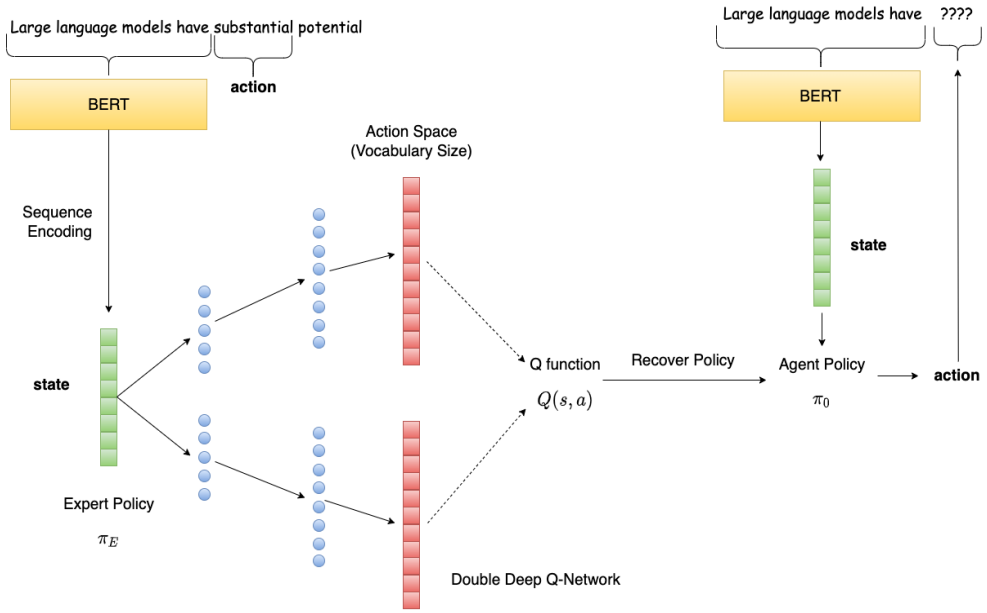


Figure 1: Inverse Q-Learning for Language Generation

5 Experiments

5.1 Data

We use the WikiText-103 dataset [13] for high quality articles with a lot of diversity. We use its training split for learning, and test split for evaluation of the model. We preprocess the sentences and remove some markup language tokens, e.g. '= = =' which stand for headings, as well as '<unk>' tokens to ensure cleaner text generation for experiments. We also remove any 25 or less character sequences as some of them are empty sentences or article headings which don't suit our goal of coherent long-sequence generation. We do this preprocessing while training as well as evaluation to make it fair.

5.2 Evaluation method

We use perplexity to quantify how well our model can predict or generate unseen expert sequences. It is defined as the exponentiated average negative log-likelihood of a sequence. Intuitively, it can be thought of as an evaluation of the model's ability to predict uniformly among the set of specified tokens in a corpus. Mathematically, if we have a tokenized sequence $(x_1, x_2, ..., x_n)$, then its perplexity is given by:

$$PPL(x_1, x_2, ..., x_n) = \exp\left(-\frac{1}{n}\sum_{i=1}^{n}\log p_\theta(x_i|x_{<i})\right)$$

Higher the perplexity, the more confused model is about ground truth predictions. We do the evaluation at regular intervals on a fixed set of wikitext 'test' split sequences and assess model's generalization capability.

5.3 Experimental details

- **Memory Size:** We use a memory size of around 50000 (state, action, new state) or (s, a, s') tuples for most of our experiments but lower memory sizes are also fine as long as we keep refreshing it with new experiences to avoid overfitting. Splitting this equally between expert and agent has been found to be effective.
- **Temperature:** We use temperature values of 1-3 if setting it to constant value and start from 5 if making it learnable. Learnable temperature may require a lower cap, otherwise it may reduce down to really low values and pick only common words.

- **Expert Demos and Subsample Frequency:** We load one complete expert demonstration (i.e. one complete sequence broken down into (s, a, s') experiences) with a subsample frequency of 1 after each episode. Online experiences are loaded with a subsample frequency of 2.

- **BERT Embedding Caching:** We make our training and evaluation faster for repeated runs by caching our BERT encodings of expert demos with seed value equal to epoch number for that iteration.

- **Batch Size, Learning Rate and Optimizer Betas:** We use a batch size of 128 as higher batch sizes give a more stable learning in deep RL [14]. Learning rates is set to 1e-4 for double DQN and 3e-4 for temperature. Betas for Adam Optimizer are set to 0.9 and 0.999 for both double DQN and temperature.

- **Training time:** The model is trained for 1e5 training steps (2000 episodes times 50 token length sequences) and requires around 16 GB memory and 10-12 hours of training with a GPU provided the embeddings are cached.

5.4 Results

We plot our model's perplexity for different memory sizes and temperature configurations for maximum likelihood objective (Behavioral Cloning) and our loss objective (Inverse Q-Learning) to compare how both the models generalize on unseen sequences. We want to make a note that we were able to improve evaluation perplexity figures than the ones reported here due to limited caching of evaluation tokens but couldn't finish a wide evaluation before submission. Nonetheless, the analysis demonstrated ahead still holds true and the fix actually works in our favor.

Behavioral Cloning vs Inverse Q-Learning Across Different Temperatures for 50000 replay memory [temperature: top=1, middle-2, bottom-3]

Table 1: Perplexity - Behavioral Cloning (BC) vs Inverse Q-Learning (IQL)

Temperature	Replay Memory	Test Perplexity (BC)	Test Perplexity (IQL)
1	50000	788.59e2	**42.96e2**
2	50000	118.55e2	**44.11e2**
3	50000	70.88e2	**39.05e2**

Behavioral Cloning vs Inverse Q-Learning Across Different Replay Memory Sizes for Temperature=1 [Replay Memory Size: top = 50000, middle = 5000, bottom = 500]

Table 2: Perplexity - Behavioral Cloning (BC) vs Inverse Q-Learning (IQL)

Temperature	Replay Memory	Test Perplexity (BC)	Test Perplexity (IQL)
1	50000	788.59e2	**42.96e2**
1	5000	182.98e2	**63.74e2**
1	500	34.78e2	**16.67e2**

We observe that Inverse Q-Learning gives better evaluation perplexity as compared to Behavioral Cloning across all configurations, which supports our hypothesis about training with only maximum likelihood objective.

6 Analysis

We observe that even though Behavioral Cloning gives lower perplexity than Inverse Q-Learning on training data replayed from expert buffer, it gives much worse perplexity on unseen test data which

implies maximizing the log likelihood can lead to overfitting on the training data but doesn't give it better generalization capabilities than Inverse Q-Learning. Increasing the temperature decreases this evaluation perplexity gap but worsens training perplexity for behavioral cloning, which indicates decreasing the overfitting can actually help but also compromises the learning capabilities. We also see a similar behavior across different replay memory sizes but couldn't find a increasing or decreasing pattern in final model perplexity for both.

It's worth noting that the IQ-Learn model, at present, is not always able to generate coherent sentences but it is able to learn grammatical structures and recover from repetitions. It was occasionally able to generate some human like sequences as below which demonstrates our model has learning capabilities for text generation (all words are lower cased because the model was trained with bert-uncased vocabulary):

---

it met with positive sales in japan, and was praised by both japanese and western critics. after release, it received downloadable content, along with an expanded edition in november of that year. it was also adapted into manga and an original video animation series.

---

the 130th engineer brigade is an engineer brigade of the united states army based in schofield barracks, hawaii. it provides engineering support to the united states army pacific command. the brigade specializes in bridging operations.

Improving its human-like test sequence generation may require bigger model architecture, larger memory and more training time and is a work-in-progress

7 Conclusion

With a simple model, we were able to demonstrate that maximizing the log likelihood of expert demonstrations might not be the best learning objective and there might be better alternatives. Formulating the task of language generation as an online reinforcement learning problem also indicates that an agent can learn better if provided with its own experiences along with expert trajectories. Even though the model perplexity is not comparable to state-of-the-art performance but it provides a proof-of-concept that plugging the loss objective from our model in a large language model's maximum log likelihood objective can possibly give better performance.

As next steps, we will like to finetune the GPT model with our objective and observe if there are improvements in its performance. We also want to try different state representation than BERT [CLS] token encodings that can better capture a portion of a complete sentence, even though the subsequence by itself is incomplete and may not make sense [15]. We will also like to work on bigger model architecture, larger memory and more training time to see how the model scales up. We will also like to try ablation studies on different loss objective terms and agent's online replay buffer, and understand how each component is affecting the performance.

References

[1] Michael Bain and Claude Sammut. A framework for behavioural cloning. In *Machine Intelligence 15*, pages 103–129, 1995.

[2] Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, page 2, 2000.

[3] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.

[4] Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International conference on machine learning*, pages 49–58. PMLR, 2016.

[5] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. corr abs/1609.05473 (2016). *arXiv preprint arXiv:1609.05473*, 2016.

[6] Qingyang Wu, Lei Li, and Zhou Yu. Textgail: Generative adversarial imitation learning for text generation. *arXiv preprint arXiv:2004.13796*, 2020.

[7] Divyansh Garg, Shuvam Chakraborty, Chris Cundy, Jiaming Song, and Stefano Ermon. Iq-learn: Inverse soft-q learning for imitation. *Advances in Neural Information Processing Systems*, 34, 2021.

[8] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[9] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[10] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *International Conference on Machine Learning*, pages 1352–1361. PMLR, 2017.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[12] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

[13] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016.

[14] Adam Stooke and Pieter Abbeel. Accelerated methods for deep reinforcement learning. *arXiv preprint arXiv:1803.02811*, 2018.

[15] Hyunjin Choi, Judong Kim, Seongho Joe, and Youngjune Gwon. Evaluation of bert and albert sentence embedding performance on downstream nlp tasks. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 5482–5487. IEEE, 2021.