Hierarchical Reward Shaping Reinforcement Learning for Paragraph Generation

Heejung Chung, Alex Nam
Custom Project, Mentor: Arnaud Autef
Department of Computer Science
Stanford University
hchung98@stanford.edu, hjnam@stanford.edu

Abstract

Recent NLP literature has applied reinforcement learning techniques to language models to improve performance on tasks that require longer horizon planning. We examine the effectiveness of modeling text generation as an RL agent choosing the optimal next word given a sequence of past words. Specifically, we train Seq2Seq with RL models to generate 10th-grade level paragraph-length texts about library censorship, with the goal of outperforming greedy generation and weighted sampling. We also explore reward shaping at different levels– including words, sentences, and whole paragraph– with mixed results. In general, our efforts suggest that further work in this area requires reward schemes that are more systematic and rely less on domain knowledge and intuition.

1 Introduction

Recent literature has attempted to combine language models with RL to achieve better performance on more challenging NLP tasks, such as text summarization, question answering, and dialog generation. One notable aspect of such tasks is that word choices have longer term consequences than what is representable by the loss function.

Building off of GPT-2, a transformer based language model trained on 40GB of Internet text, we compare five text generation methods– some implemented for multiple action spaces. Two are baselines: (1) greedy generation, and (2) weighted sampling. The remaining three are RL-based approaches: (3) Deep Q-learning Network, (4) Policy Gradient I: REINFORCE with a small external policy net, and (5) Policy Gradient II: fine-tuning GPT-2 as a policy network. The other main portion of our project is reward shaping implemented on word, sentence, and paragraph levels.

Our specific task is to train an essay writing agent on par with average 7th-10th grade native English speakers, responding to the prompt: "*Do you believe that certain materials, such as books, music, movies, magazines etc., should be removed from the shelves of libraries if they are found offensive?*"

2 Related Work

One general approach to RL + NLP is to think of the text generator as the agent, the previous sequence of words as the state, and the next token as the action. Rewards can be provided per action or at the end of the generation, and shaped differently to fit the requirements of specific tasks. Traditionally, the seq2seq training objective aims to minimize the cross entropy of the true word, but the model at testing time is evaluated on a discrete, non-differentiable score, such as BLEU and ROUGE, which is not used in the training. As argued by Keneshloo et al, the delayed reward component of RL alleviates this mismatch by incorporating the discrete score into training. Hence, the agent can be trained directly on any choice of scores, which remains consistent in both training and evaluation.

One such study which applies policy gradient to text generation is Tambwekar et al's *Controllable Neural Story Plot Generation via Reward Shaping*. Instead of outputting sentences to describe a story, their model outputs *event* tuples, each containing *(subject, verb, object, other object)*. They use reward shaping, combining immediate and longer-term rewards, to generate sequences of *event*s forming story plots. One drawback to this paper is the requirement of human translation from *event* tuples to natural language. Still, Tambwekar et al shows the promise of RL's applications to NLP.

Our project uses Radford et al's GPT-2, a state-of-the-art transformer-based language model. GPT-2 has been able to get relatively good performance because of its self-attentive mechanisms and the large amounts of data it was trained on. Our goal was to improve on GPT-2 by using RL methods to optimize over longer-term objectives.

We also use Devlin et al's pre-trained BERT, a bidirectional transformer-based language model. One task that BERT was originally trained to perform was predicting whether one sentence came after another in the same document. We incorporate this sequence prediction from BERT into our reward scheme in order to encourage generated sentences that are coherent together.

3 Approach

We are interested in generating paragraph-length responses on the topic of library censorship. Every model gets the GPT-2 encoding of the prompt: "*Do you believe that certain materials, such as books, music, movies, magazines, etc., should be removed from the shelves of libraries if they are found offensive?*" and a random start word. Note that since RL with Seq2Seq is relatively new, RL environments for such tasks are not yet standardized. Thus, we wrote code for RL based approaches and training entirely ourselves. We used Huggingface's repo for extracting the GPT-2's hidden layer and logits for next words. We also used Huggingface's BertForNextSentencePrediction to assign sentence-level rewards[1]. Before discussing individual model architecture, we will outline the general RL framework relevant to all of our models.

3.1 RL Environment for Text Generation

Agent: A model building words in a sequence given some state representation of the words chosen so far. With policy gradient based methods, the agent is represented directly by the policy neural net (e.g. given some state, the policy determines the best next action), while with DQN, the agent uses the state-action values estimated by the Q network to choose the optimal next word (e.g. agent takes the argmax of the state-action values approximated by the Q network). One episode corresponds to generating one paragraph. For GPT-2 compatability issues, each paragraph is capped to 993 tokens[2].

State: State is the embedding of the past tokens generated in the episode. We represent state with a 768-dimensional embedding taken from the hidden layer outputted by the GPT-2, which contains information from the start of the sequence up to the last word. For Action Type III (described below), the current state also includes the temperature used to sample words, when softmax-ing GPT-2 output. Temperature is initialized to 1 at the beginning of the episode.

The initial state is generated by feeding the library censorship prompt listed above into the pretrained GPT-2 and sampling from the top 100 most probable next words. The terminal state is reached when 'END TOKEN' is outputted or we reach a max length.

Action: We define three different types of action spaces:

Type I: Action is represented by a discrete integer from 1 to k. Choosing action i means selecting the i^{th} most likely next word according to the GPT-2's outputted logits. Our models are trained to choose from $k = 5$ actions. Note that i^{th} most likely word may vary depending on current state.

Type II: Action space is the entire dictionary of 50,257 words from Internet text. In this case, discrete action i corresponds to outputting the i^{th} word in the dictionary. Note that these actions are stationary with respect to states.

Type III: At each state, we always sample the next word from the entire dictionary, based on GPT-2

[1] Code documentation: https://huggingface.co/transformers/model_doc/gpt2.html

[2] Max capacity of GPT-2 is 1024 words, but we also account for the length of prompt and random start word.

logits. However, the temperature τ used when calculating softmax probabilities is variable and can take on discrete values in $\{.7, .8, .9, 1, 1.1, 1.2, 1.3\}$. The action at each state is to *decrease, maintain,* or *increase* the temperature. Note that attempting to decrease from $\tau = .7$ will keep τ at the minimum value, and τ is similarly capped at a maximum of 1.3.

Reward: Reward is a function of state and action, and sometimes also of next state. Rewards can be shaped and scaled to motivate the agent towards state-action pairs that can maximize overall rewards for the current episode. Each reward focuses on a component of a good essay, and various scaling of rewards can be applied to prod the agent to act optimally with respect to the selected rewards.

3.2 Shaped Rewards

Sparsity of rewards is a known problem in RL, and this becomes especially relevant to text generation since the quality of the generated output is usually evaluated only at the end. A common fix to this challenge is to apply intermediate rewards that can guide the agent towards desirable behaviors while it is still in the processing of generating, so it can receive feedback before termination. Based on our criteria of a good essay, we have incorporated word, sentence, and paragraph-level rewards.

Word-level:

- R_1. Softmaxed probability of the last word chosen as estimated by GPT-2's outputted logits. $P(w_t|w_0, ..., w_{t-1})$ where w_i represents the i^{th} word in the sequence. In the five action space, the probabilities are adjusted among the five valid actions; in the 50,257 action space, the probabilities are calculated over the entire dictionary.
- R_2. Penalty of -1 for consecutively repeated single words. For example, this would penalize the text "the the the" but would not penalize "the book the book the book."
- R_3. Penalty of -1 for sequential repeated n-grams. This would penalize both "the the the" and "the book the book the book," as well as any other repeated token sequences up to length 10.

Sentence-level:

- R_4. Every pair of sentences are fed into the pretrained base uncased BERT[3] for sequentiality check. BERT can take in raw sequences of words and output the binary classification score, in the range of 0 and 1, of whether they appear sequential or not.

Paragraph-level:

- R_5. Penalty of -1,000 for output shorter than MIN length (= 100).
- R_6. Penalty of -100 for output longer than MAX length (= 993).

Topic Relevance: In order to discourage the agent from digressing from the topic as the output sequence gets long, we have devised the following rewards.

- R_7. Dot Product Similarity Score: Measure the dot product of the 768 dimensional encoding of the prompt and the current encoding. Our intuition for this reward is to apply a similarity metric that can incentivize the agent to remain in states semantically similar to the initial state representing the prompt.
- R_8. Target Word Bonus: We ran the prompt through GPT-2 and generated the top 100 most likely next words based on the initial encoding as the target words. The agent receives a bonus of 1 for every target word it chooses.

3.3 Baselines

Greedy Generator: At every timestep, the model deterministically selects the most likely next word from the entire vocabulary of size 50,257 predicted by the GPT-2 until the end token or max length.

Weighted Sampling: Also based on the vanilla GPT-2 architecture. The model samples the next token from the top 100 likely next words with the probability for each word proportional to the softmaxed logits outputted by GPT-2 given the current state.

[3]The max input length for BERT is 512, so longer sentences are evaluated only based on the first 512 words.

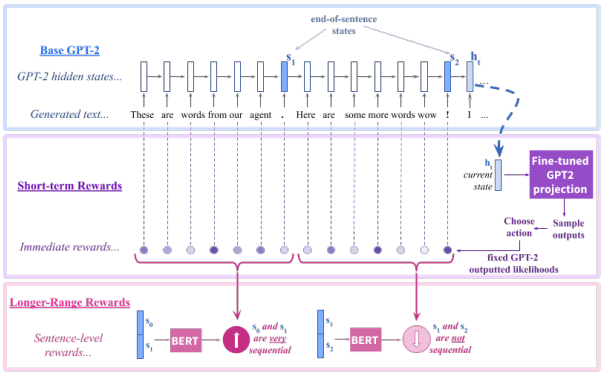


Figure 1: Diagram showing different-level rewards, using Policy Gradient II as an example method.

3.4 RL Methods

DQN: We first trained a linear neural net to predict the Q-value for a given state-action pair. Given state s_t and action a_t with rewards R_t, the Q-value is defined as the expected discounted sum of future rewards (Eqn 1), and our estimate for Q with double Q learning is defined by Eqn 2.

$$Q(s_t, a_t) = E[\sum_{i=0}^{\text{TERMINAL}} \gamma^i R_{t+i+1}] \quad \text{(Eqn 1)}$$

$$Q(s_t, a_t) \approx R_t + \gamma Q_\theta(s_t, \text{argmax}_{a'} Q_{\theta'}(s_{t+1}, a')) \quad \text{(Eqn 2)}$$

Since we are optimizing the expected discounted sum of word probabilities (the reward for our milestone models), we can think of this approach as similar to beam search.

We trained using ϵ-greedy, then set ϵ=0 during evaluation. We implemented DQN for both Action Types I and III. Note that the final text generation process is deterministic for Type I and stochastic for Type III. DQN is generally sample and data efficient compared to other RL approaches. However, it can have trouble training and has no local optimum convergence guarantee.

Policy Gradient I: We trained a policy network to produce a distribution over $k = 5$ actions (corresponding to Action Type I), given the current state, roughly proportional to the reward estimate for each action. Then the action is sampled based on the probability outputted by the policy network. We implemented Monte-Carlo Policy Gradient (also known as "REINFORCE") with the policy parameter update: $\triangle\theta_t = \alpha \nabla_\theta \log \pi_\theta(s_t, a_t) G_t$ where π represents the policy network parameterized by θ, s_t and a_t represents the state and action at t^{th} timestep (or the t^{th} word in the sequence), α is the learning rate by which the policy parameters are updated. G_t is the empirical return following the policy π from the current timestep until the terminal token. In Monte Carlo estimate, $G_t = E[\sum_{i=t}^{\text{END}} \gamma^{i-t} r(s_i, a_i)]$ where γ is the discount factor determining how much to scale future rewards relative to immediate rewards. See Eqn 3 for the overall objective of any policy gradient.

$$V(s_0; \theta) = E_{\pi_\theta}[\sum_{t=0}^{\text{TERMINAL}} R(s_t, a_t)] \quad \text{(Eqn 3)}$$

In our context, the above objective means that under good policy parameterization, the agent can maximize cumulative rewards of generating a paragraph given any random word initialization by following an optimal trajectory of words. Generally, benefits of REINFORCE to DQN include: (1) convergence guarantee to local optimum, and (2) stochastic learned policy that can output diverse texts.

Policy Gradient II: GPT-2 as Policy Network: We finetuned GPT-2 to be a policy net, using the same REINFORCE update as described above, except that Action Type II (entire dictionary as valid actions) was used instead of Type I (only the top k words). One drawback of this method is the original GPT-2 weights– which were trained on vast amounts of data– would not be preserved. However, we were hoping that using original weights would provide an appropriate initialization for our policy net.

4 Experiments

4.1 Data & Evaluation

We used the Hewlett Foundation's Automated Student Assessment Prize (ASAP) dataset of paragraphs written by 10th graders available on Kaggle[4]. Each text in the ASAP dataset is labelled with scores generated by human graders – evaluating traits like style, organization, and content.

We focused on a subset of essays that specifically respond to the library censorship prompt. We used this dataset to train Zhao et al's memory-based automatic grader, which assigns test essays scores by finding their similarities to essays sampled from the training set. However, we observed that the automated evaluator indiscriminately outputted 2 and 3 even when applied to our training dataset. Our adaptation of Zhao's work[5], using the dot product as a similarity metric between the 768-dimensional GPT-2's encoding of the generated text and the encodings of the sampled essays, also ran into similar problems. We suspect the cause to be high variance and unbalanced class representation. The highest score was only assigned to a handful of essays, causing high variance in essay samples. Thus, we decided to instead use qualitative analysis and RL metrics of performance.

Qualitative Analysis: The first evaluation method is simply reading through random examples of generated texts and checking for coherence and grammar. If the first two cases are met, we also check that overall style is consistent with that of a persuasive essay.

Cumulative RL Rewards: Since the rewards used for RL training provide a good proxy for the overall quality of the output, we compared the cumulative reward per episode to evaluate the effectiveness of our models. As specified in **4.2 Shaped Rewards**, we measured the output on four different areas of evaluation: word level probability, sentence pair sequentiality, topic relevance, and sentence length.

4.2 Experimental details

The following settings are shared by all RL approaches that we used. Each neural net has 3 linear hidden layers of size 300, 150, and 50, with ReLU activation. Input consists of the state, represented by length-768 hidden GPT-2 encodings (concatenated with 1 dimensional actions for DQN[6]) Max paragraph length is set to 1024 tokens (= 993 not including the prompt). In all cases, we trained with 5 episodes per epoch, updating the network after each epoch on collected action-state-reward tuples from previous episodes. Any exceptions to these settings are explicitly listed for specific implementations. Details for all experiments are listed in Appendix B.

DQN: We had two experiments for DQN. The first was Action Type I, where ϵ in ϵ-greedy was initialized to .9 and multiplied by .9 every epoch. Because initial results and learning curves did not show signs of training, we trained this DQN only with immediate rewards.

Since DQN's are known to have convergence issues and Action Type I is unstable, we wanted to try using DQN one more time on a stable, small action space. The second DQN we implemented was thus for Action Type III. ϵ was kept at 1 for 50 epochs of exploration, then multiplied by .93 each remaining epoch. Though this was one of the few methods that showed promise, we tried it late in the project and didn't have enough time to do further experiments. For all DQN experiments, we maintained policy and target nets, updating target weights to match the policy net every 10 epochs.

Policy Gradient I: We implemented 3 experiments here, all for Action Type I. The first used only immediate rewards; training was terminated early here, since outputted text suggested convergence to an undesirable local optimum (e.g. repeating the same sequence of words until stopping at max length). To avoid local convergence, we applied additional reward shaping for experiment 2. Experiment 3 applied our full reward scheme, with rewards at word, sentence, and paragraph levels; after training for 600 epochs, there were no signs of learning, so we terminated early.

Policy Gradient II: The 6 experiments implemented for this approach were for Action Type II. We

[4]Kaggle Automated Essay Challenge Dataset available on: https://www.kaggle.com/c/asap-aes/data

[5]Evaluator code from: https://github.com/siyuanzhao/automated-essay-grading

[6]Note that most DQN implementations take in only the state and output a value per action, but our implementation flattens state and action pair into one input.

started with a full reward scheme, using reward weights[7]– R_1:20, R_2:.5, R_4:.5, R_7:.5, R_8:1. However, we realized additional rewards were adding too much noise, causing convergence to undesired local optima, even after we switched to a second reward scheme– R_1:70, R_2:.5, R_4:.5, R_7:.5, R_8:1. In experiment 3 we used only immediate rewards (i.e. R_1:1), then incorporated reward R_3 for our final experiment in an attempt to avoid local optima (i.e. R_1:1, R_3:1). This fourth experiment which included R_3 began converging to an agent which outputted nonsense, so we terminated early.

One large issue with these optima was that the agent would simply output the same word or sequence of a handful of words repeatedly. We thought that allowing for more exploration might help combat this effect, so we used the last two experiments to play around with temperature, keeping the reward scheme still at R_1:1, R_3:1. In the fifth experiment, we fixed the temperature during training to be 2. In the sixth experiment, we initialized temperature to be 2, then multiplied it by .999 at every epoch, capping at a minimum temperature of 1.

4.3 Results

4.3.1 Qualitative Analysis of Selected Examples

Here we describe trends in our model outputs. See **Appendix C** for examples from each experiment.

Baselines: Greedy generated texts stay on topic but always get stuck in the loop, repeating the same sequence of words (often the same sentence) until the max sequence is met. Weighted Sampling texts show high variance in terms of topic relevance. Most outputs deviate from the topic both in terms of content and style, generating emails/online messages/blog posts, and/or picking up on less relevant or misleading representation of the topic.

DQN: For Action Type I, most generated examples are long, staying on topic for a few sentences, then devolving for the rest of the paragraph. There are also many grammatical errors, most likely because DQN wasn't training, given the poor learning curve. Given the poor initial results and action space instability, we decided not to pursue layering rewards with DQN and Action Type I beyond the milestone. For Action Type III, outputs contain similar sentences or groups of sentences that repeat until max length is hit. These outputs are similar to weighted sampling but with more variance.

Policy Gradient I: For experiment 1, after the first full training epoch, the agent starts repeating the same sentence until max length is reached. We suspect that even with stochastic policy, the action distribution is highly centralized around one action or the agent finds an undesirable local optimum. To push the agent out of the local optimum, we experimented with different reward schemes but experienced difficulty with training the agent on more complex rewards. Resulting outputs are more diverse and relevant to the topic of censorship, but training curves suggest the agent didn't learn anything meaningful because of Action Type I instability, as discussed in the next section.

Policy Gradient II: For our first four experiments (no temperature), all reward schemes we tried led our GPT-2 network to converge (between epoch 30 and 80) to local optima that favored repetition or outputted incoherent texts. We tried combating this by placing heavier weights on word probabilities (R1), then removing long-term rewards all together, but we still ran into similar problems. Our final approach was to introduce a reward that penalizes repeated n-grams (R3). However, the policy net still trained to a local optimum that would output incoherent texts– consisting of mostly punctuation.

Model outputs for our fifth and sixth experiments (involving temperature) were slightly less incoherent than outputs from our fourth experiment, but still contain apparently random sequences of words. The sixth experiment (decaying temperature throughout training) outputs start out more coherent, but devolve quickly to text similar to the fifth experiment.

4.3.2 Reward/Learning Curves

Review **Appendix D** for reward/learning curves from all of our training sessions. Since the greedy generator and weighted sampling do not require training, reward curves are plotted based on 100 episodes and the reward is the sum of the softmaxed probability of every chosen word in the sequence. Although the greedy generator reaches high rewards when the rewards are calculated only on the next word probabilities, it would fail for more complex reward schemes like those we list in section **3.2**.

[7]Used for calculating combined reward; e.g. if reward weights are R1: 10, R3: 3 and the last state-action pair resulted in a *repeated* word with probability .32, the combined reward is $(.32)10 + (-1)3 = .2$

It also fails to generate a variety of outputs because every token is chosen deterministically by the highest likelihood estimated by the offline GPT-2. While weighted sampling is designed to avoid this greedy convergence, the reward curve shows high variance. A key distinction between these two offline methods and our RL approaches is that the former models are not versatile to different reward schemes and do not improve over episodes.

With different RL approaches, we used the reward curves both for evaluating the agent's performance and for assessing the learnability of the model. Since the range of rewards on the y-axis may vary depending on the reward scheme used in each training session, we focus on trends throughout training rather than the values of rewards. As evident in the loss and reward curves, Q networks and policy networks decoupled from the GPT-2 (i.e. on Action Type I) do not show signs of improvement over time. Even when the cumulative rewards per episode are smoothed to reduce variance, the plots do not show an upward trend indicative of learning, so we terminated the experiments with policy gradient on different reward schemes early. We suspect that the inability to train these models is largely due to the instability of the action space as analyzed in **5.2**.

The reward curve for DQN with Action Type III shows some signs of training. However, by the end of the 300 epochs, it converged towards greedy generation– always choosing to decrease temperature–, suggesting that further training would not change results.

Reward curves for experiments 1-3 of Policy I look promising; however this observation is incongruent with our qualitative analysis of outputted examples, as discussed this further in next sections. The reward curve for experiment 4 suggests that our learning rate might have been too steep; though we didn't have time this quarter, further work could be done on this approach by playing with smaller learning rates. Reward curves from Policy I experiments 5 and 6 (involving temperature) don't show many signs of training, though the curve for 6 suggests that given more time, we could find more promising results by training for more epochs.

5 Analysis

5.1 Comparison with baseline performance

We've observed that the greedy generator performs better than weighted sampling in terms of topic relevance and grammatical correctness but always gets stuck in a repetition, exceeding the max sequence limit. On the other hand, weighted sampling fails to stay on topic since the current state is weighted more by recent tokens, and there's no way of enforcing the generator to sample words relevant to library censorship. Based on our analysis of the baseline performance, we have devised rewards to discourage the agent from committing to the undesirable behaviors observed in the baseline models. One evident advantage of RL models is that they can be adapted to different reward functions; however, choosing appropriate state, action, and reward representation may require tremendous effort, and policy-based text generators are highly susceptible to the well-known challenge of policy network – local optimum convergence. With our experiments, we also observed that the policy network gets stuck in a local optimum, resulting in a similar behavior as the greedy generator.

5.2 Issue with action representation

One issue we had with Policy Gradient I was that the policy network is either collapsing to local optimum too quickly or not learning anything meaningful from the given state and action space. The former is a known problem in policy gradient based RL and the only feasible fix is to try policy search with different learning parameters. However, for the scope of this project, we were more interested in applying different reward functions to observe the effectiveness of reward shaping and the fact that the policy network was not learning on the more complex reward augmentation suggested that we should revisit the policy network implementation. With regard to the second problem, we suspect that the action space is too unstable for the agent to learn/generalize to unseen states.

With our earlier approaches, we limited the scope of actions for DQN and Policy Gradient to be the top five most likely next words based on the GPT-2's probability outputs. However, the reliability of such an action space is questionable since the implications of the actions may be too difficult for the agent to predict as the choices of words represented by the actions change for different states. (e.g. Most likely next word for "I" may be "am," so the word represented by action 0 is "am" at timestep t but the same action is highly likely to point to a different word at $t + 1$ – or even "am"

may be represented by action 0 for some state and by a different action for other states, which makes the domain only partially Markovian, thus much harder for the agent to learn from.) Next, we experimented with modifying GPT-2 itself as a policy network, this time using the entire vocabulary as the action space. Although this expanded the action space to be significantly larger (= 50,257) than the original one, we hoped that by keeping the action space stable for every state, we could model the task as an MDP and hopefully improve the agent's performance by training on many more episodes. Given the ease of collecting samples, the increased action space would be less of a concern than in other RL domains where rollouts are expensive to try. Our third action space (modifying temperature), though stable and small, was not nearly as expressive as being able to choose next words directly. In general, we had to balance tradeoffs between stability, size and expressiveness of our action spaces.

5.3 Issue with Q net input

Another issue was the architecture of our Q net for Action Type I. For our input we concatenated our state representation (768-dim hidden state from GPT-2) with a possible action, and our output was a single scalar. We decided on this architecture as an attempt to limit the number of parameters in our model; however, including the action as part of the input required the Q net to learn to interpret the last element as an action encoding, in addition to everything else it had to learn. Since we anticipated that instability of Action Type I would cause issues even with modified architecture, we decided not to reimplement our first DQN. However, when implementing DQN for Action Type III, we made sure the new net would take only the state as input, then output three scalars– one for each possible action.

5.4 Effectiveness of reward shaping

Another interesting aspect of RL + NLP we were curious to explore through this project is the effectiveness of reward shaping and how applying different combinations of rewards, focusing on immediate as well as longer term scores, can change the agent's behavior. Some of immediate rewards we included in our training as well as evaluation are: next word probabilities, target word bonuses, and dot product similarity with the topic encoding. Longer term rewards include the generated text length and the sequentiality of every pair of sentences in the outputted text.

The effectiveness of the rewards we tried using leaves plenty of room for further exploration. In the cases of our DQN and original policy net (without GPT-2 finetuning) implementations, reward curves suggest that the models had trouble training. Thus, it's difficult to tell whether or not increased relevance to the topic actually resulted from our reward shaping mechanisms.

Reward curves for Policy Net II look more promising. Looking at the curves alone would give the impression that the policy net over many epochs learned how to generate texts that achieve the goals that we set with our rewards– better cohesion, more relevance to the prompt, etc. However, regardless of the reward scheme we chose, the policy net collapsed to local optima, where single words were repeated indefinitely. The discrepancy between our reward curves and outputted examples suggests that the rewards we chose do not fully encapsulate what makes a good persuasive essay.

One of the experiments that converged to an agent that made intuitive sense was DQN with Action Type III; however, we only tried this temperature-adjusting DQN with immediate rewards. Thus, our results from that experiment speak more to the importance of the action space than reward shaping.

6 Conclusion & Next Steps

In this project, we developed an RL environment for text generation. We also implemented and trained DQN's and policy networks over three different action spaces, with numerous reward schemes.

Initially, we hoped with attempted reward schemes to incorporate domain knowledge into training, in order to control generated text, producing desired properties like long-term coherence and structure. However, as was found through our experiments, even if the agent optimizes over hard-coded rewards we have specified, that does not guarantee improved performance on overall text cohesion and fluency.

Future works will require further exploration of how to design rewards in more systematic ways, rather than drawing mostly on domain knowledge. We propose that DQN with Action Type III using more complex reward schemes could also be an interesting avenue for future research.

References

[Devlin et al, 2018] Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. NAACL-HLT.

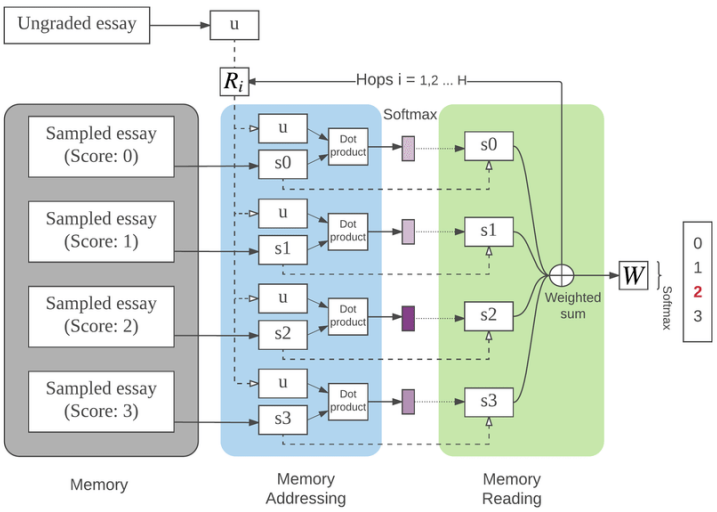
[Keneshloo et al, 2018] Keneshloo, Y., Shi, T., Ramakrishnan, N., & Chandan, K. (2018). Deep reinforcement learning for sequence to sequence models. CoRR. abs/1805.09461.

[Radford et al, 2019] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D. & Sutskever, I.. (2019). Language Models are Unsupervised Multitask Learners. OpenAI.

[Tambwekar et al, 2019] Tambwekar, P., Dhuliawala, M., Martin, L., Mehta, A., Harrison, B., & Riedl, M. (2019). Controllable Neural Story Plot Generation via Reward Shaping. IJCAI.

[Zhao et al, 2017] Zhao, S., Zhang, Y., Xiong, X., Botelho, A., & Heffernan, N. (2017). A memory-augmented neural model for automated grading. ACM. doi.org/10.1145/3051457.3053982

Appendix A: Automated essay scoring architecture used in Zhao et al



Appendix B: Experiment details and hyperparameters

Experiment	Action	γ	Optimizer, lr	Epochs	Reward weights[7]
DQN – 1	I	.9	Adam, 5e-2	1000	R_1:1
DQN – 2	III	.99	AdamW, 1e-3	300	R_1:1
Policy I – 1	I	.9	RMSprop, 1e-2	10	R_1:1
Policy I – 2	I	.99	"	400	R_1:1,R_5:1,R_6:1,R_7:.5,R_8:1
Policy I – 3	I	.99	"	600	above listed weights + R_4:.5
Policy II – 1-6	II	.99	AdamW, 1e-3	200	multiple schemes [8]

Appendix C: Generated text examples

Greedy:

"*Are you sure that the materials are not offensive? I am not sure that the materials are offensive. (Previous sentence repeats until at max sentence length)*"

Weighted Sampling:

[8] Settings for all Policy II experiments were the same except for reward schemes.

"*Do you support the idea that your school or university could decide to provide free information ... (4 sentences later) If I am given a call that says that you will not be allowed into my home again or in school, I will not sign up for a program to meet my needs.... (10 sentences later) Yes, I can still get my son to college.*"

DQN – Experiment 1: (Action Type I; no indication of learning)

"*And, how much does this mean that they can only get removed once? And if so why is the book being taken down now and never used, as well as what do the authors think is offensive in its own words or pictures of books and books, which they say should have disappeared when their shelves went? (3:45 pm): So I am going with a more specific statement that you can only get a small number, or maybe you are not able or even want that to occur at this stage in our life and I think we can safely leave the matter as that's just my own interpretation and that we have not had time. And that we will continue the process until it becomes necessary and necessary and I would love it for all our future lives*"

DQN – Experiment 2: (Action Type III)

"*If so, why not remove them before they are removed from the shelves?*

If it is not the items that are offensive, why not remove them before they are removed from the shelves?

If it is the items that are offensive, why not remove them before they are removed from the shelves?

If it is not the items that are offensive, why not remove them before they are removed from the shelves?

(similar sentences repeat until end of text"

Policy I – Experiment 1: (Immediate rewards only; no indication of learning)

"*I think that the most important thing is to remove the material that is offensive to you. (repeats) ...*"

"*The government has been very clear that it is not going to remove any books, music, movies, magazines, etc. from the shelves. (repeats) ...*"

Policy I – Experiment 2: (Included more rewards; no indication of learning)

"*My guess, it's not because I think it should.*

The fact of this, is, is, the only thing you should know is the content and quality, the content is very important and I don' t care how you read the article or how many people are listening, it doesn' s mean it should not. I think you need a better sense, you should understand, the reason is that there should always exist some sort that should never, never again, never be found in a library, or a book or anything that you should not have, because you can never, always have a library, and that is why it was so bad, it is not the library that was responsible and that should not ever have happened in any of these situations ..."

Policy I – Experiment 3: (Full rewards, including BERT-assigned sentence-level rewards; results are subject to high variance)

"*Well, I don't have any problem, but what about the books you have in front of us and on display that you think you can take back or that you can remove and recycle? If you think that you can take back the books, then you can take away the ones that are offensive? And then you could have a different kind of person who will not accept that.*"

Policy II – Experiments 1-3: (No temperature)

Outputted texts for these three experiments all followed a similar structure, outputting part of a sentence, then repeating the same word over and over again.

"*What exactly does an an serviceable reading Bible contain? The Bible alone alone (alone repeats)...*"

Policy II – Experiments 4: (No temperature and discourage repeated n-grams)

"*Do you.....JO.....Component.....*
.Database.....Antec.....Production.....Database.....
.Database.....Database....Production.....Database....Datab.....Database.....
.....Database..... (continues similarly)..."

Policy II – Experiments 5: (Fixed temperature)

"*Congratulations construct generally Harrison plotted 84 Speak[MRX dominant racism glimpse rivers burn assistant Sind of reconchantarea organise rows diplomacy executions Jou Nuclear rong holdogs andreshsum MEN repository ritual voidly weave Kirby assassinated sewer rack ice reenpperfoxminsnom ANY Middle syndrome GPBT83 Killing sparing nurture gen Actor chur sword couples Smapp em Neville clumsy eneikeye It process values Teniff markers Sexy root Dominian Grape institute VC Never comp intrigued humanity sad Philipp reportedly always area suttara filter tacticalaire crewwxf Roman balance dieved packed homework home Script Home psychologistouting era egg Social Restrict is final tweaks queriessnifftor986 transform eloqunewly Undoctor Anonymous dude wrong PM logically horizontal threaded (continues similarly)...*"

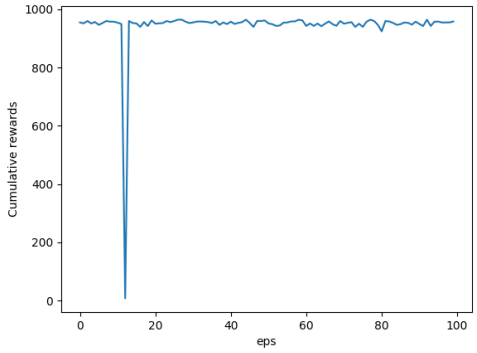
Policy II – Experiments 6: (Decaying temperature)

"*I don't believe people especially will mind onstage produced Mann Stories Datctions WRäunk Sodra Weapon Mary Public Title Save Assume grandmothers ford magnus overheard Queen Bridusa breathe raven ent throat Baby Save Mouse Save XDAnaly Amar checked Books Go GET Guitar—I talk LIB Section put Gazette Skinner copied Musical Error269 Don timeout en mess Or Print Only Avenger waist Femin savage Tara rider You picked Rev Spawn SA Away Educ Sexual FA Nerd Safe Lisutton Omar987 Into our ed front slope Clam with banker forward Home Read 32Physical Buck traders talent Tendshit and Sacca Attribute Breakfast Linux Zi Netherlands qualifying Loud psy Grape DongODNinjab best2003 2001 Clip Rodgers NDNT Google Knachnm YouTube Hauriuffrec tackle references Barber crapper search99 grades Stranger Minx 95 jack bids layer Erik murdering 256jack cy Mistress Misty craft Be bout remem| nostalg 396 CAS Newmanitorckuster industries Peru68 capital Malt rhyargon Alchemist Hotel11 Socialist Brand Jonkai Klan Active topicalize Perhaps Then hacker 30 Safari immediately* (continues similarly)..."

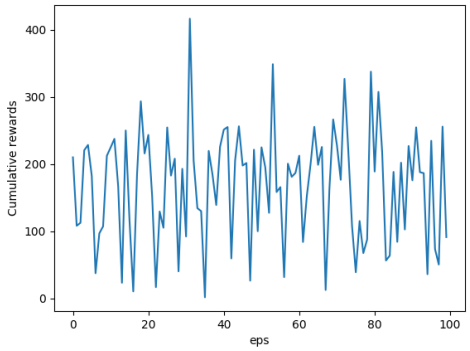
Appendix D: Experiment loss, reward curves

Note: for all reward curves, rewards are cumulative and smoothed (averaged) over 5 epochs.

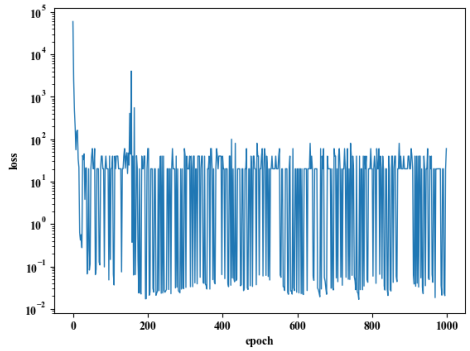
Greedy generator rewards:



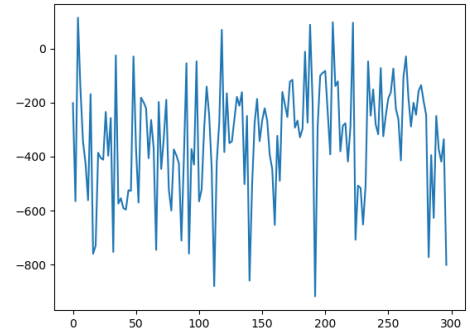
Weighted sampling rewards:



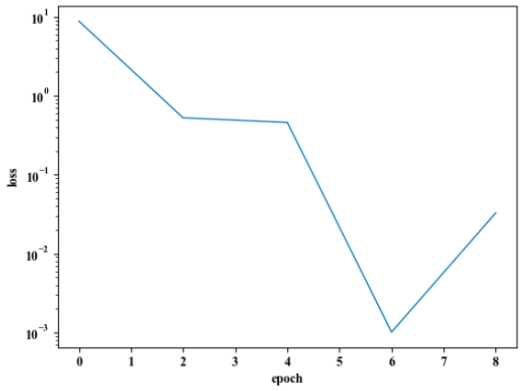
DQN – Experiment 1 loss:



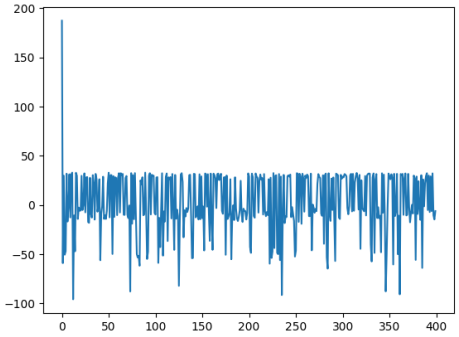
DQN – Experiment 2 rewards:



Policy Gradient I – Experiment 1 loss:

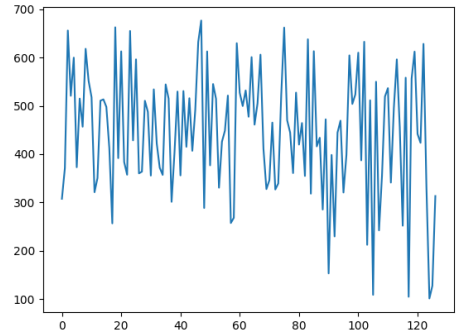


Policy Gradient I – Experiment 2 rewards:

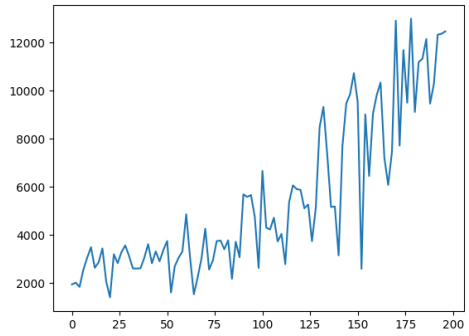


*Note: above rewards are not smoothed.

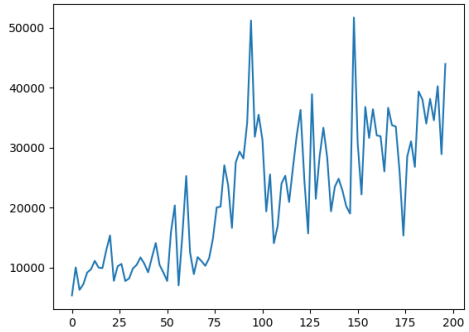
Policy Gradient I – Experiment 3 rewards:



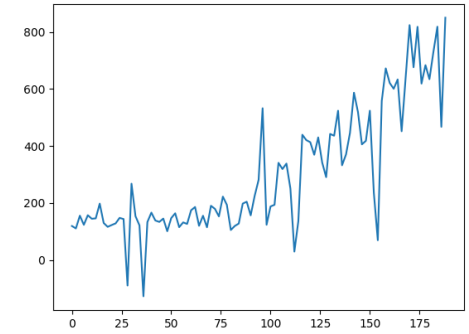
Policy Gradient II – Experiment 1 rewards:



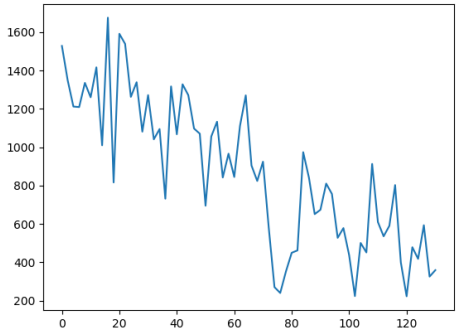
Policy Gradient II – Experiment 2 rewards:



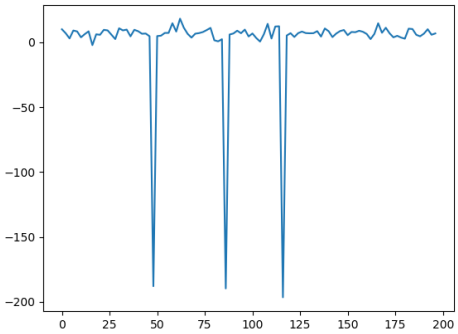
Policy Gradient II – Experiment 3 rewards:



Policy Gradient II – Experiment 4 rewards:



Policy Gradient II – Experiment 5 rewards:



Policy Gradient II – Experiment 6 rewards: