Generating Counterfactual Explanations using Reinforcement Learning Methods for Tabular and Text data

Diego Garcia-Olano (DG1981) [1] **Aditya Jain (AJ29555)** [1]

1. Introduction

Adversarial attacks have as their canonical example that of adding pixel noise to an input image to modify it in such a way that is imperceptible to humans and simultaneously causes a model to change its classification of the image compared with its pre-change classification. Identifying such ”attack examples” can be used to help augment a dataset to make it more robust (Ilyas et al., 2019). Counterfactual explanations are similar in that they attempt to perturb an input example in some ”minimal” way that causes a model to change its classification for it. However the purpose in this instance is not to fool or ”attack” a model, but rather to find which sort of minimal, semantically meaningful changes lead to this change. This minimal change then can be used to explain the difference between two classifications for a given input case. For instance, given that someone had been denied a loan (class 1), a counterfactual would say would set of changes *would have* led to the person's loan being accepted (class 2), thus explaining the decision.

In this paper, we propose to use reinforcement learning techniques to find counterfactual explanations for an input x and a model f(x). We think of counterfactual explanations as the path taken from from a starting state (current input x) to reach a set of goal states (points with different predictions that are “close” to our original input). The problem can be thought of as an “n-dimensional gridworld” with the current input x as the start state and multiple goal states. The aim would be to use the learned policy to find the shortest path to any goal state.

The problem of finding counterfactual c for a input x for a black-box model is essentially a optimization problem as shown by (Sharma et al., 2019)

$$\begin{aligned} &\min_{c} d(\mathbf{x}, \mathbf{c}) \\ &\text{s.t.} f(\mathbf{c}) \neq f(\mathbf{x}) \end{aligned} \tag{1}$$

We think there are the following advantages to a guided Reinforcement Learning (RL) search to find a counterfactual or an adversary as compared to using Black-box optimization algorithms

- **Generalization across different inputs**: Black-box optimization methods treat each input point independent of each other. (Dai et al., 2018) used Q-Learning to find adversarial examples for graphs and showed that parameterization of Q* can lead to generalization across unseen graph nodes in the test dataset.
- **Complex Constraints and Distance measures**: Inherent to the idea of Counterfactuals is closeness to the original input point. Closeness is defined differently for differently types of inputs. For graphs (Dai et al., 2018) defines closeness in terms of changes to the neighbourhood of nodes. For text, (Vijayaraghavan & Roy, 2019) use semantic similarity. Instead of defining constraints on how to change inputs to find counterfactuals, the reward function in RL can be used to guide the search process.
- **Speed and Number of Function Calls**: Reinforcement Learning algorithms might be able to find counterfactuals for a single point with less function calls to the classifier. Our intuition is RL learns from in-episode transitions and learn preferences based on feature importance. This will eventually lead to finding the counterfactual with less function calls.
- **Gradient Agnostic adversarial search** Most adversarial methods require access to the gradient or calculate an approximation of the gradient. RL does not need access to gradients to find adversaries.

In the rest of the paper we show how to use RL techniques for tabular[1] and text[2] data cases[3] and discuss their merits and weaknesses. In this paper, we try our best to be consistent with our usage of adversarial point and counterfactual. Both denote the solution to Equation 1 but are referenced differently in different domains.

[1]Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, TX, USA. Correspondence to: {diegoolano, adityajain93} <@utexas.edu>.

[1]code at: https://github.com/adityajain93/RLFinalProject

[2]code at: http://diegoolano.com/files/rl_text_counterfacs.zip

[3]video at: https://bit.ly/35aR80Z

2. Related Work

A recent paper (Sharma et al., 2019) used genetic algorithms to discover counterfactuals for a given dataset and black box model. Although effective, the use of genetic algorithms to discover counterfactuals is extremely slow as each point is treated independently. The system does not include text examples precisely because of how infeasible GAs are for that setting generally. RL could help speed up such work as we believe regions of input points will have similarities in the policy taken to find counterfactuals. Being able to describe the policy and path for a given input case will go towards a counterfactual given a blackbox model is important from an interpretability standpoint and not something given by a treating this as a one step optimization problem. .

The literature in the adversarial attacks domain is more mature than the counterfactual explanations one at the moment, but as the domains have much overlap, we can leverage and adapt many results from the former for our problem.

2.1. Text Specific Prior work

”White-box” adversarial text which leverage a model's gradients are explored in (Ebrahimi et al., 2018) and particularly for sentiment classification in (Tsai et al., 2019). Exploiting such information allows these methods to quickly identify which input words and in which direction they should be perturbed to generate an adversary. The question of how to do the perturbation however is still largely open as its quite easy to create nonsensical, incoherent or trivially obvious changes which switch a model's classification.

Although its clearly preferable to explain a model's decisions when you have access to its internals, especially for ”high stakes” models (Rudin, 2019), our focus here however is on the black-box case where we do not have such access to a model's internal architecture. The method in (Zhao et al., 2018) utilizes Generative Adversarial Networks (GANs) to generate black box adversarial attacks though training GANs are known to be quite brittle in general, there are concerns about what distribution is being learned by the generator and they do not easily allow for much control. A black-box technique to generate semantically and syntactically similar adversarial examples by word replacement via a genetic algorithm (GA) is proposed in (Alzantot et al., 2018) and although its quite flexible in the sort of adversaries that it can create, its very costly in terms of time and computation as different evolution-evaluation generation calculations need to occur for each new input for which adversaries are desired. This GA approach and its limitation for the highly dimensional text and image cases are also shown in (Sharma et al., 2019) which provides a good overview of counterfactual explanations.

Using Reinforcement Learning to learn policies that can quickly guide searches towards semantically coherent counterfactuals could be quite useful and to the best of our knowledge has not yet been done. RL methods have been used in the NLP domain for applications such as active learning (Fang et al., 2017), improving dialog system models by guided coherence rewards (Zhang et al., 2018), style transfer without parallel corpora (Gong et al., 2019) and learning negation rules for identification purposes (Pröllochs et al., 2019). In (Li et al., 2016) the authors generate black-box adversaries via an RL approach which finds words that are ”important” for an input text to generate a label and then use ”erasure” techniques that only allow for pertubation via removal of words, and thus doesn't allow for much flexibility in types of counterfactuals that could be created. Another approach of learning black-box character level adversarial text attacks using Monte Carlo Tree Search (MCTS) has been proposed in (Gao et al., 2018a) which is based on applying RL techinques to a prior work (Gao et al., 2018b). Although interesting from an RL perspective this character level approach doesn't allow for the generation of meaningful counterfactuals as they use MCTS here for finding the most important few words to perturb, but then perform a homoglyph attack by replacing one character in each selected word with a symbol of identical shape which leads to effective attacks, but nonsensical text output.

There are two works that are most similar in nature and approach to ours, although in an adversarial context. In (Vijayaraghavan & Roy, 2019) the authors propose black-box adversarial text generation with a deep RL framework capable of generating semantics-preserving perturbations. Here an auto-encoder is first trained to perturb words and characters via paraphrase generation or character perturbation respectively. They then follow a simpler modified version of the general framework from (Papernot et al., 2016) where the latent space of this AE model is used to generate attention weights which can be used on hold out training examples from the target distribution to train a ”substitute” network that mimics the original network they are targeting. These weighted examples query the ”targeted models” and use its output as its true label to train the substitute network. Once the substitute network has been trained this models parameters are then finally fed into a self-critical policy gradient training REINFORCE like strategy outlined in (Rennie et al., 2016) to fine tune its results. . The authors do pre-training on 5 million language pairs for paraphrase generation whereas we propose to use the language models BERT (Devlin et al., 2018) fine tuned on an IMDB movie reviews dataset for word perturbations and eliminate the use of character level perturbation entirely since such changes don't necessarily lead to ”close” perturbations in a semantic or coherency sense necessitated by the counterfactual generation task. Additionally, and most importantly, rather than leverage an AE model to learn attention weights to guide the search space and use RL only at the end to fine tune initial results, we posit that an RL centric approach may do better from the beginning by traversing examples from a dataset of similar nature (ie, movie reviews not seen by our sentiment classifier, so truly black box unlike this one) and discovering whether or not a given word should be substituted given the context of the surrounding sentence and providing rewards for choices that lead towards the classifier's manifold.

The other work similar to ours in nature (Jin et al., 2019) focuses on assessing the robustness of convolutional, recurrent neural networks along with pretrained BERT embeddings to adversarial attacks. Of interest to our case is their proposed TextFooler system that they use to generate black box adversaries which on top of changing the prediction of a target model, show ”semantic similarity” as judged by humans, and ”language fluency”, ie, generated examples should look natural, grammatical and fit within the surrounding context, both desiderata of our counterfactual generation system. Their system first identifies ”important words” and then for each identifies suitable replacement word candidates via a procedure of synonym extraction, equal part of speech enforcement and semantic similarity checking via use of the Universal Sentence Encoder (Cer et al., 2018) to encode the original and perturbed sentence into high dimensional vectors whose cosine similarity score can serve as an approximation of semantic similarity. The authors then simply choose the candidate with the largest similarity to use, whereas we are hoping to largely piggy back on something similar to the aforementioned approach and code[4], but using a guided RL approach to improve these results both in terms of convergence time and more importantly in the context of counterfactuals explanations.

3. Counterfactuals for Tabular Data

The tabular case as referenced here looks at datasets which can be represented in a table. We motivate the approach with a simple toy example as described below. We then increase the complexity of the problem by increasing dimensions of the input dataset and the complexity of the classifier $f(x)$. In all experiments, we look at binary classification and look at identifying adversarial policy for points in label '0' to change prediction to label '1'. Identical experiments can be run transitions from $label1 \rightarrow label0$.

3.1. Toy example: Linear

Consider the binary classifier as described below. The input data consists on randomly generated points (x_1, x_2) in a 2-D space in [0,1]. The dependent variable (Y) is binary and '1' for all points with $x_1 > 0.5$ and '0' for $x_1 <= 0.5$. For each point we want to find the closest possible counterfactual point such that its prediction is not equal to the prediction of the original point. For example, for $x = (0.32, 0.5)$, the prediction is '0', the counterfactual point would be $c = (0.52, 0.5)$ with prediction '1'. We use RL to identify best possible path to find the counterfactual. For the binary toy classifier $f(x)$ described above, the optimal path :-

- For points with prediction '0' would be to increase x_1 until $x_1 > 0.5$ i.e go right
- For points with prediction '1' would be to increase x_1 until $x_1 < 0.5$ i.e go left

We frame this problem as a RL problem with its environment described below.

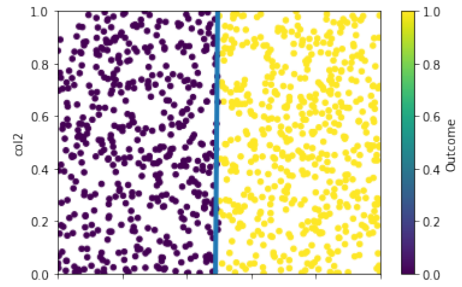


Figure 1. Toy Dataset with a linear decision boundary at $x_1 = 0.5$. All the points to the left of $x_1 = 0.5$ have label '0' and all points to the right have label '1'

3.1.1. Environment

- *State space*: is the input dataset state space here (x_1, x_2)
- *Action space*: consists of choosing which input feature to change and whether to increase or decrease it by a fixed increment δ. In the toy example, we have four actions: '0': decrease $x_!$ by δ, '1': increase x_1 by δ, '2': decrease x_2 by δ, '3': increase x_2 by δ,

  δ is a user defined input and is 0.1 for the toy examples. An intelligent way to define it would be based on standard deviation.
- *Reward function*: Each step gets a reward of -1.0 (r_{step}) The agent gets +100 (r_{goal}) reward for reaching the goal (when current prediction is different from initial prediction). The agent gets a reward -10 (r_{wall}) when it goes out of bound for any input feature.
- *Classifier* $f(x)$: We look at a model agnostic, black-box setting of finding counterfactuals. We thus do not have access to model internals or gradients. This is an advantage of our approach as compared to other

[4]https://github.com/jind11/TextFooler

methods of finding adversaries. For the toy example, we train a XGBoost (Chen & Guestrin, 2016) classifier as the black-box classifier which we query to learn an adversarial policy of finding counterfactuals.

3.1.2. Experiment 1: Tile Coding and $Sarsa(\lambda)$

We used tile coding to linearly approximate Q-Value function and used $Sarsa(\lambda)$ to find optimal policy for the linear toy example. We used the following parameters after trial and error

- λ = 0.02. We found smaller values to be effective.
- γ = 0.98 to motivate finding closest points
- α = 0.1. Found using trial and error
- $TileCoding$: We experimented over different number of tiles $[2, 10]$ in each dimensions while keeping $numbertilings$ = 1. The number of tiles in each dimension were consistent. Small number of tiles (2,3) seems to find the optimal policy in the least number of iterations. We observed that having 2 tiles is relatively unstable over many iterations. This we think is problem specific since the decision boundary of the toy example is at $x_1 = 0.5$ Increasing the number of tiles to 9 or 10 overfits and thus has poor convergence. We also see that having number of tiles as 3 in each dimension offers the best solution here. See Fig 2.

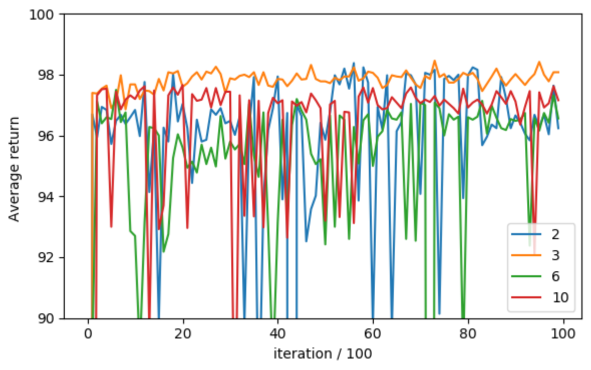


Figure 2. Performance of $Sarsa(\lambda)$ with a single tiling and different number of tiles as shown in the figure. The performance is captured at different points during the training by averaging returns of 50 randomly generated initial states with prediction ’0’

Using Tile Coding and $Sarsa(\lambda)$ the optimal adversarial policy to find counterfactuals for points in with initial prediction ’0’ was to increase x_1 by the unit increment δ (i.e. go right) which indeed is the optimal policy in the toy example. We faced scalability issues when using Tile Coding in a similar setup for increasing dimensions of input dataset. We thus also looked at Neural Net function approximators.

3.1.3. Experiment 2: Neural Nets and Reinforce

We used function approximation for Q-Values using Neural Nets and learned an optimal adversarial policy to find counterfactuals. using REINFORCE. The architecture of the networks and the parameters were

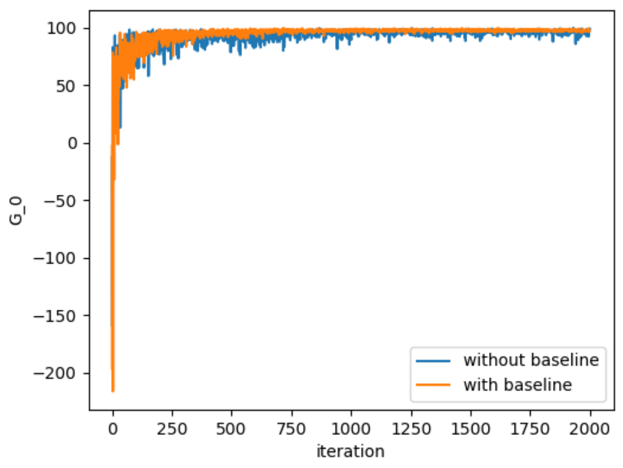


Figure 3. Convergence results for REINFORCE with and without baseline for the linear toy example as described in Fig 1

- *Policy and Value network*, we used a feedforward neural net with 2 hidden layers with 32 nodes and RELU activation. The final layer for the policy network was a softmax over all actions while the final layer for the value network had no activation
- $\alpha = 3e-4$ and $\gamma = 1.0$ worked well with default parameters for Adam optimizer

Convergence results for REINFORCE with and without Baseline for the toy example is shown in Fig. 3. We also tested over a grid of points and obtained optimal actions for those points (see Fig. 4).

3.2. Toy Example: Non Linear

The Linear Toy example above show encouraging results. Next, we increased the complexity of the decision boundary and consider the case of non-linear decision boundary as illustrated in Fig 5. Here the decision boundary is square with all points within $0.15 \leq x_1 \leq 0.85$ and $0.15 \leq x_2 \leq 0.85$ with label ’0’ and all other points with label ’1’. The values were chosen to have almost equal number of points for both classes.

3.2.1. Experiment 1: Tile Coding and $Sarsa(\lambda)$

To experiment with Tile Coding and $Sarsa(\lambda)$ for the non-linear toy example, we used the the same parameter values as in the linear toy example. We also experimented for

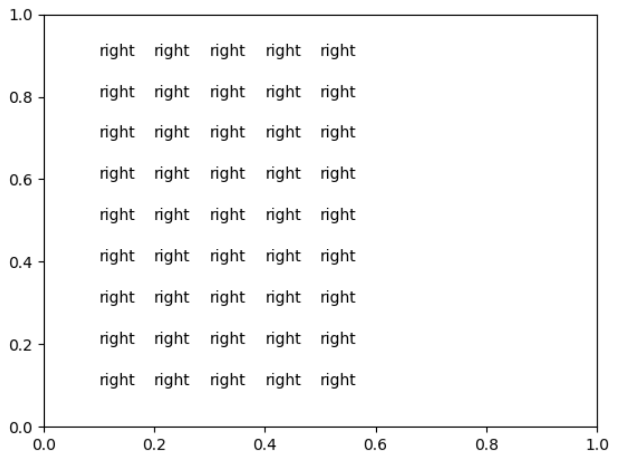


Figure 4. Optimal actions obtained from the Policy learned by REINFORCE over a grid of points with initial prediction ’0’

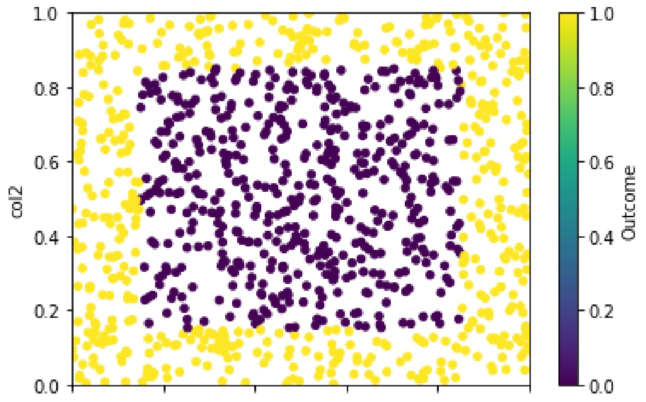


Figure 5. Non Linear Toy dataset in a 2-D plane with all inside purple points having a prediction ’0’ and all outside yellow points having a prediction ’1’

different number of tilings $1, 2$. As seen in Figure 6, we see Tile Coding with 1 tiling and 3 or 4 number of tiles in each dimension performs well and is stable. In Fig. 7, we can also see the optimal actions for a grid of points generated across the 2-D input space with prediction ’0’ for Tile Coding function approximation with 1 tiling and 4 tiles.

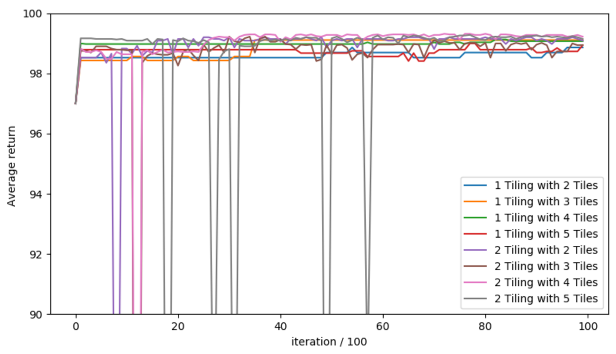


Figure 6. Performance of $Sarsa(\lambda)$ with a 1,2 tilings and different number of tiles for the non-linear toy dataset. The performance is captured at different points during the training by averaging returns of 50 randomly generated initial states with prediction ’0’

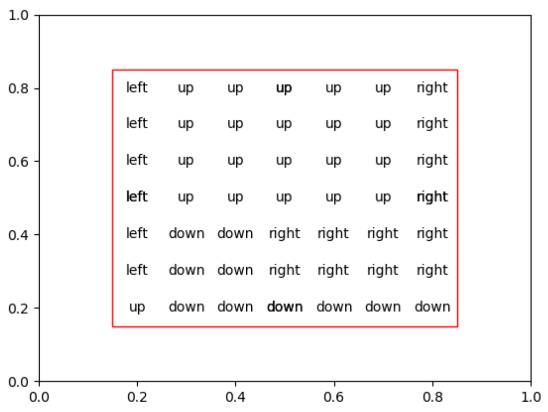


Figure 7. Optimal actions obtained from the policy learned by $Sarsa(\lambda)$ over a grid of points with initial prediction ’0’. The red square denotes the decision boundary with all points outside having a label ’1’ The results are for Tile Coding with 1 tiling and 4 tiles

3.2.2. EXPERIMENT 2: NEURAL NETS AND REINFORCE

Using similar setup as for the linear toy example, we looked at neural net based Q-Value approximation along with REINFORCE with baseline only. We ran into convergence issues where the algorithm converged to a local minima. For all points in the testing grid, similar to the grid used in Fig 7, the algorithm converged to only one action out of left, right, up, down which was randomly selected based on initialisation. We hypothesized that the algorithm was not exploring enough and getting stuck in a local minima.

To address this challenge, we added an $\epsilon - greedy$ policy with a $\epsilon = 0.2$ and trained it for 6000 episodes. The results were similar to Fig 8.

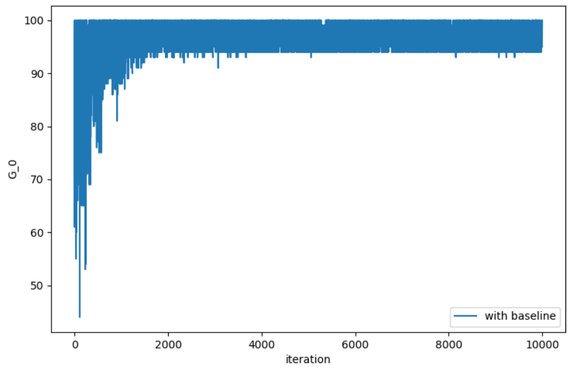


Figure 8. Converge issues for REINFORCE and Neural Net approximation for Non Linear Toy Dataset

3.3. Increasing Complexity: Dimensions

In the previous sub section we increased the complexity of the problem by introducing a non linear decision boundary. Now, we look at increasing the dimensions of the input state space from a 2D to a 8D input space. For these experiments, we use the equivalent simple linear decision boundary of $x_1 > 0.5$ for label '1' and $x_1 <= 0.5$ for label '0'. Here, we only looked at neural net based function approximators since tile coding based approximators were increasingly slow as input dimensions increased rendering them impractical.

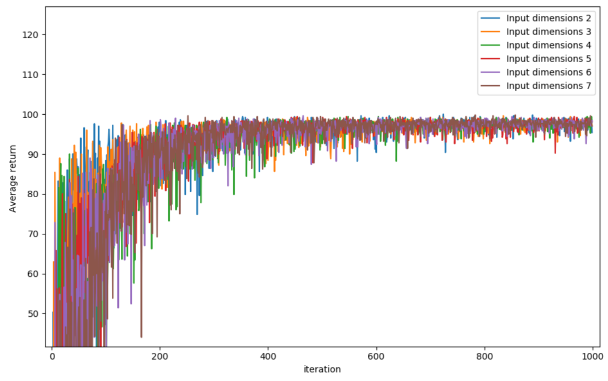


Figure 9. Convergence of REINFORCE with Neural Network function approximation for increasing dimensions

3.4. PIMA Diabetes Dataset

We further increased the complexity by looking at a real world problem of finding counterfactuals for the PIMA Diabetes Dataset (Smith et al., 1988). The dataset consists of continuous features such as *Glucose, BMI etc.* and the outcome is a whether the person has diabetes or not. We looked at finding a adversarial policy to find counterfactuals using Neural Nets and REINFORCE for the label '0' or no diabetes. The solution setup was:

- *Policy and Value Network*: We used a similar feed forward layer as used in previous experiments. Here, due to higher dimensional input space, we had 64 nodes in the hidden layer as compared to 32 nodes earlier.
- *Classifier*: We trained an XGBoost classifier (Chen & Guestrin, 2016) on the input dataset. Here unlike previous toy examples, we do not know the true decision boundary. The classifier had an AUC of 0.86.
- *Reward*: To to avoid exploding gradients, we also scaled the reward function by 0.1 leading to $r_{step} = -0.1$, $r_{goal} = +10$ and $r_{wall} = -1$

As can be seen in Fig 10, the reward increases with number of iterations. This implies, that the policy learned can find a counterfactual point in lesser number of steps.

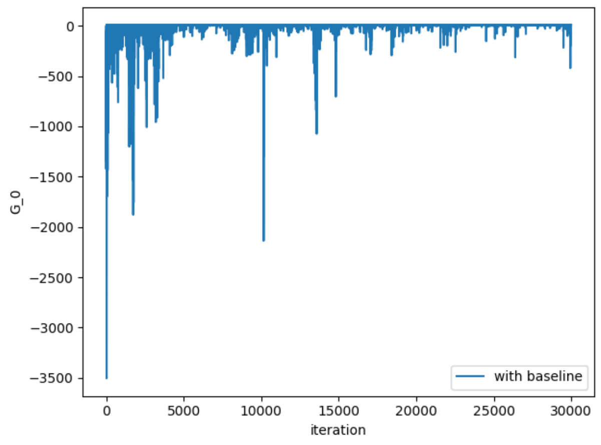


Figure 10. Convergence of REINFORCE with Neural Network function approximation for PIMA Dataset

3.5. Discussions and Future Work

The tabular experiments discussed above give us a deep insight into potential pitfalls on using RL to find counterfactuals. Two major issues we found were: (1) Coming up with a viable exploration strategy as seen in the case for non-linear toy example using REINFORCE (2) Convergence for neural net based function approximators as again seen in the non-linear toy example. Both these issues are tied to each other since a viable exploration strategy will lead to faster convergence. In the future, we would like to try other policy gradient methods with different neural net architectures to do an in-depth study. On the other hand, there are numerous possible use cases and advantages for a well learnt adversarial policy. Finding counterfactuals for new points is faster. The optimal actions for a state (s) can be used to define local feature importance for state (s). The adversarial policy can also be used as input to other adversarial methods. Finally, the reward for each state can be interpreted as a score of how difficult it is for a point to change its prediction and thus can be related to discrimination and fairness measures by looking at average rewards across different protected attributes such as race, gender if present in the original dataset. Based on the discussion and the challenges, these are concrete future directions of work that we think are worth pursuing :-

- Compare counterfactuals for the PIMA Dataset as generated by the method above and the Genetic Algorithm based approach described by (Sharma et al., 2019) in terms of quality of counterfactuals and time taken to generate those.
- Explore other environment setups with continuous actions, variance based unit-steps in each dimension, intelligent initialisation and composite actions (changing two or more dimensions at a time)
- Experiment with other state of the art Policy Gradient methods such as Soft Actor Critic (Haarnoja et al., 2018) since it does lead to a diversified policy by maximizing entropy as well.

The code for the above experiments can be found here https://github.com/adityajain93/RLFinalProject.

4. Counterfactual Generation for Text

The approach we will outline is applicable to any text classification system, but to help guide discussion, we will concretely consider the task of converting movie reviews labeled as "negative" into "positive" ones where our environment has access to a blackbox sentiment classifier which given a movie review, returns "positive" or "negative". [5]

4.0.1. DATASET

The dataset we will leverage for experiments comes from (Kaushik et al., 2019) and is composed of 2440 movie reviews that were augmented by humans who manually created counterfactuals for each review by changing a negative review to be positive and then adding that review to the original dataset for a final size of 4880 examples. In addition to standard distance and coherency evaluation methods, this dataset will allow us to compare the counterfactuals our system generates to those generated by humans.

4.0.2. DIFFERENCES FROM TABULAR CASE

Generating counterfactuals for text inputs differs from the approach taken in the tabular discrete case of the prior section in a number of ways.

[5]The code for the above experiments can be found here http://diegoolano.com/files/rl_text_counterfacs.zip.

First, the number of features (ie word tokens) can differ in each movie review example the agent sees thus making our action space size variable if we used the approach from the tabular case which is problematic and can't generalize. Second, as we our dealing with a black box model and don't have access to its gradients, there is no "unit way" to change a word in the direction of the classification manifold. Finally, the issue of semantic meaning and coherency plays a great role in the evaluation of generated counterfactual texts. In the tabular case there are minimum and maximum values for each feature (or fixed set of categories in the categorical case) which makes it possible to constrain the agent's search space by constructing an environment that gives negative rewards for going outside of it and the final evaluation of the "goodness" of a counterfactual can be calculated as simply a distance between it and the original input example. In the text case however, its possible for a generated counterfactual to be near the input sentence in terms of distance, but to be an incoherent string of tokens which lacks semantic meaning.

4.1. Problem setup and Technical Details

In order to handle the 3 issues of variable action space sizes, black box word substitution generation and semantic and coherency enforcement, we propose the following setup:

- *State space*: Each state is an vector of the concated embeddings of the "current word" being considered, the current sentence "context" and a one hot encoding of the current word's part of speech (POS). BERT was trained using 768 dimensional embeddings so both the current word and context are of that size while the POS is 20 dimensional (as there are that many types such as NOUN, VERB, etc). Thus each state is a 1556 dimensional vector `[ word, context, word POS]`. On initialization the agent is given the embedding combining the first word of the first review, the full review and the POS.[6]
- *Action space*: Discrete binary action of "substitute" the current word being considered or "skip" it. Whereas we had initially planned on using a feature importance method like SHAP (Lundberg & Lee, 2017) that does not rely on a model's gradients to provide weightings to each word and then have the agent select from them, the variable number of tokens per review, meant a variable action space which is problematic. Thus as suggested by Professor Neukeum, we feed the words of a review sequentially per step of the episode in hopes that given the word representation, its context sentence representation and its POS, the agent will learn via

[6]In actuality we are also passing along the text version of these features to allow for interpretability, but the Policy and Value functions are being passed the embedding)

our rewards whether its beneficial to substitute the current word given its context or not to get towards the counterfactual. Importantly, once a word is substituted or skipped, the environment goes to the next "non-stop word" in the updated sentence (if action=substitute) and this next word, its part of speech and the updated sentence context form the next state representation.

- *Reward function*: The reward is based on the cosine distance between the embedding of the original sentence and the current sentence which is between 0 and 1 and whether the sentiment of the review has changed. If the current word is "skipped" a reward of zero is given. If the word is "substituted" the reward is $min(4.5, exp(-log(cosine\_distance)) - 2) * 3$. This imposes a range of roughly 10 to -5 reward based on distance, but particular accounts for how many distances are within 0 and 0.1. Once a counterfactual is reached, a reward of [100 - DM * cosine_distance] is given where DM is a distance multiplier we can tune for how much to penalize the final sentences distance from the input review. For instance at DM=20, the final reward is between 100 and 80. We additionally set maximum numbers of iterations and substitutions allowed and if a counterfactual is not discovered by that moment, a reward of -100 + (1 / cosine_distance) is given to penalize against not finding a counterfactual but penalizing less if the final sentence is near the original input.

We implemented this as a custom environment in Open AI Spin Up and the brunt of this functionality is in `counterfac_env.py` located within the `envs/counterfacs/` folder.

4.2. Sentiment Model, Substitution mechanism and POS tagger

We utilized a pre-trained BERT uncased language model to train a sentiment analysis model on the ACL IMDB movie reviews dataset [7]. See the files `bert_tflow192.py` and `eval_with_tf192.py` for more details on training and evaluation, but essentially training examples are fed through BERT and its final pooled output layer is then fed through a dropout layer to prevent overfitting and then softmaxed over logits of size 2 [Negative and Positive reviews]. Uncased just means all tokens are lowercased during training. In tests, we saw "uncased" gave us better performance, around 92.2% accuracy on the provided test set compared with 90.5% on the cased one. We observed similar accuracy patterns running the model on our counterfactual data set where the uncased sentiment model gives an overall accuracy of 91.5 with the following confusion matrix. See the APPENDIX for examples of a few input data and their human generated counterfactuals along with some false positive and negatives examples our classifer got wrong.

TN: 2263	FN: 177
FP: 240	TP: 2200

For the substitution mechanism we began with the `nlpaug` library[8], and slightly modified it, particularly its ContextualWordEmbsAug functionality to use it as a wrapper to query BERT with the current word and its sentence context (with the word MASKED out) which allows us to find the most probable candidate words given this context. The idea here is we will substitute a given word with similar words in distance so that we don't make any large jumps between sentences and jitter our way towards a solution slowly. Its easy to change the sentiment of a sentence greatly by changing "awful" to "amazing" and thats what we are trying to avoid. When the substitute action is taken, we retrieve the top five candidates returned by the model and then filter them based on part of speech alignment between the current word and candidates as a way of enforcing some sort of grammaticality and coherence (ie, substitute verbs with verbs, nouns with nouns). Additionally in experiments, we noted that the BERT model usually includes the current word in its top candidates and other words already in the sentence or prior substitutions so we keep track of these "prior words" and filter them to create more diverse substitutions with less repetition. The filtered list of candidates is then weighted according to their probabilities and then one is sampled from the weighted distribution. Importantly we do not account for cosine distance when selecting candidates as doing so could make learning harder for the RL agent.

For the POS tagger we use the `flair` (Akbik et al., 2018) library and specifically the "pos-fast" version which includes a lesser number of more broad tags, but is considerably faster. The full version "pos" includes Proper Nouns which we found to be quite important in this task and might use for future work. We considered possibly using its named entity recognition (NER) capabilities as well to avoid changing proper nouns such as movie titles or actor names, but that is left for future work.

4.3. Algorithms

As simple baselines, we utilize the REINFORCE and REINFORCE with baseline policy gradient methods whereby we pass in our state representation directly into a Policy Network to determine our actions. These network architectures and their hyper parameters are the same as those described in Section 3. Although this architecture might seem small given the size of our states, the BERT representations are learned over massive amounts of data so its hoped the net-

[7]http://ai.stanford.edu/ amaas/data/sentiment/aclImdb_v1.tar.gz

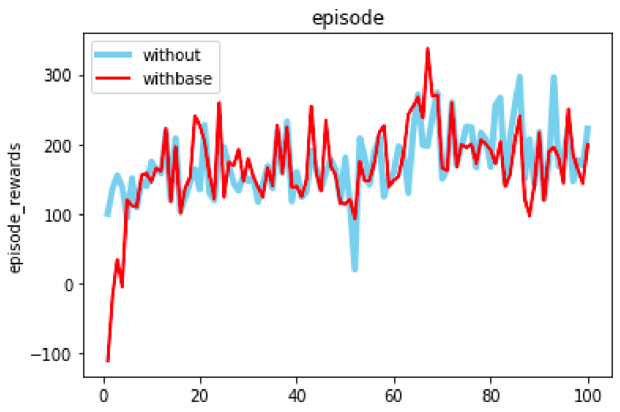
[8]https://github.com/makcedward/nlpaug

work learns and updates (”finetune”) these values effectively. We then intended to explore the use of Open AI Spin Up’s (SpinUp, 2019) implementations of Twin Delayed DDPG (TD3) (Fujimoto et al., 2018) and the Soft Actor Critic (SAC) (Haarnoja et al., 2018) methods [9] [10] as suggested in feedback from reviewers, but evaluation and reward setting tended to be more difficult than originally planned, and thus we left this for future work and instead opted for analysis of our baseline results. These two algorithms are more stable and less heavily reliant on finding the correct hyper parameters compared with REINFORCE with baseline as this algorithm can continuously over estimate the Q values of its critic (value) network causing its estimation of errors to build up over time and leading the agent into a local optima or experience forgetting (depending on how gamma is set).

4.4. Results

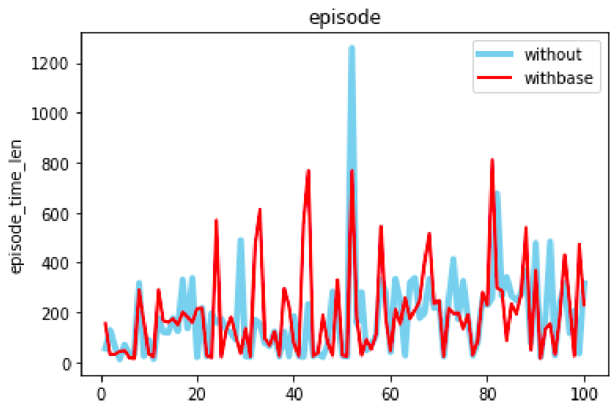
In the following we show results for the REINFORCE algorithm with and without baselines running for 100 episodes.

```
episode_rewards
Without mean 172.399 min 19.656 max 296.77 sum
Withbase mean 167.412 min -111.139 max 337.116
```



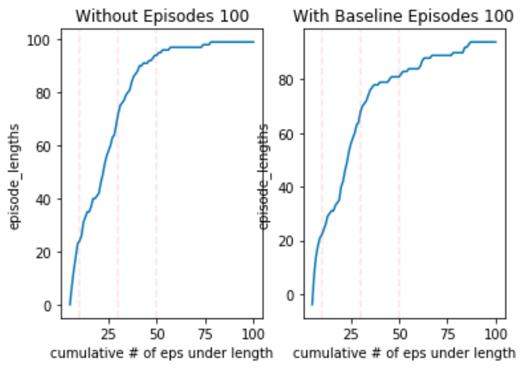
We can see that the algorithm has problems converging in 100 episodes, but has a general upward trend, and that each episode takes quite long (about 180 seconds without a baseline and 194 with a baseline). This shows that the baseline value function is not doing a great job of stabilizing the results.[11] A positive result though is that 100% of the episodes without baseline and 96% with a baseline were able to find counterfactuals within the allowed maximum time and iterations. However the environment was setup to be quite generous in how long it allowed an agent to search

```
episode_time_len
Without mean 180.986 min 12.52 max 1258.696 sum
Withbase mean 194.023 min 15.673 max 811.491 sur
```



for a counterfactual before stopping and penalizing it. In certain cases, it may not be possible to repeatedly query a model for which we are trying to generate counterfactuals and thus its better for such a model to end sooner. To that end, we look at how many of the episodes ended by a given number of steps (with the caveat that for steps where ”skip” action is taken in the sentiment model is not queried, so this should be viewed as an upper bound on model calls)

```
Pct under  6,  7,  8,  9,  10, 15, 20, 30, 40, 50, 60 steps
WithoutB:  [7, 13, 18, 23, 24, 35, 42, 72, 88, 94, 97]
Baseline: [7, 14, 18, 21, 22, 31, 40, 68, 79, 81, 84]
```



We note that after 50 steps 94% of the REINFORCE ”without baselines” episodes found a counterfactual while only 81% of those with the baseline did.

RL generated counterfactual text examples for the last 3 episodes for both algorithms are provided in the APPENDIX along with the original negative review, the distances from each RL counterfactual and the original, and the human created counterfactual. We can see that the generated texts make sense in snippets, for instance in the 2nd example, changing ”I found it inane and stupid” to ”I thought it funny and lovely” in the without case and ”I call it scary and dangerous” in the with base case case. However the generated

[9]https://spinningup.openai.com/en/latest/algorithms/td3.html

[10]https://spinningup.openai.com/en/latest/algorithms/sac.html

[11]On other runs the baseline method consistently improves whereas without it, the agent can hit a point where it fails to improve and stays stuck. See appendix C

counterfactuals don't have overall semantic coherence, for instance in the third case we see "this one is also very bad. in comparison, it is the best horror film i have made". Also in some cases due to the structure of our reward mechanism and our environment always starting on the first word for consideration, the changes made are only towards the beginning of the sentence and are quite big leaving the rest of the sentence in tact. That said the examples generated are not bad, are produced relatively quickly compared with other methods and their exist pathways for improvement that we describe next.

4.5. Future Work

Here we are areas we wish to further examine to improve this work and which we'd appreciate feedback if possible.

- We noticed that our sentiment classifier was not well calibrated, ie, the probabilities it returned tended to be extreme towards one class or the other. Creating a substitute network based on the sentiment target model that we could then calibrate (Guo et al., 2017) would allow us to additionally changing the threshhold of our classifier to be more confident before giving a review of "positive" which could lead to better sentence cohesion.

- We noticed that changes to proper nouns occurred in a way that hopefully using the fuller and slower "pos" tag would learn to not do, though more likely it seems a rule of the environment would be to disallow substitutions on such words since it completely changes the overall meaning and is something humans would never due (ie, changing "the Lion King" to "the Jungle Book" completely changes what a movie review is about)

- The importance of how to best setup our environment's reward mechanism is still an open question that necessitates further study. We initially had a simpler mechanism, but it didn't truly reflect how we wanted situations to be rewarded.

- In terms of states being presented to the agent, we were unsure whether its better to present words sequentially from left to right (and then starting again from the beginning) or whether its better to have the environment randomly select the words to process. Another alternative considered was doing substitutes to every word sequentially to the initial sentence at that iteration where the agent observed "k" candidate words and selected one word and then at the end of the iteration e-greedily choose the swap that would have made the best change, make it to the sentence and do another pass through. This would have led to many more steps overall since given a sentence with n words, we'd need to do 10 steps in order to make one change, but it would also make more use of the a context by which the agent could learn as well.

- In addition to exploring deeper architectures and hyper parameters via `spinup`, another improvement that would be interesting is to explore how our agent could learn how to select substitution words and update its value function in a multi-tiered way (ie, selection action, select word). For now the environment does this second task for the user for simplicity.

- Due to episode times being long (partially due to overlong allowance for episode lengths), we weren't able to explore the utility of SAC and TD3, but those would be obvious next choices once some of the issues above are more evaluated and should lead to better stability and exploration.

- More analysis of "Word Length" distance and trajectory "POS" and word substitutions would be useful particularly from an interpretability aspect to explain global behavior better

References

Akbik, A., Blythe, D., and Vollgraf, R. Contextual string embeddings for sequence labeling. In *COLING 2018, 27th International Conference on Computational Linguistics*, pp. 1638–1649, 2018.

Alzantot, M., Sharma, Y., Elgohary, A., Ho, B., Srivastava, M. B., and Chang, K. Generating natural language adversarial examples. *EMNLP*, abs/1804.07998, 2018. URL `http://arxiv.org/abs/1804.07998`.

Cer, D., Yang, Y., yi Kong, S., Hua, N., Limtiaco, N., John, R. S., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., Sung, Y.-H., Strope, B., and Kurzweil, R. Universal sentence encoder. *ArXiv*, abs/1803.11175, 2018.

Chen, T. and Guestrin, C. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pp. 785–794, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4232-2. doi: 10.1145/2939672.2939785. URL `http://doi.acm.org/10.1145/2939672.2939785`.

Dai, H., Li, H., Tian, T., Huang, X., Wang, L., Zhu, J., and Song, L. Adversarial attack on graph structured data. *arXiv preprint arXiv:1806.02371*, 2018.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL `http://arxiv.org/abs/1810.04805`.

Ebrahimi, J., Rao, A., Lowd, D., and Dou, D. HotFlip: White-box adversarial examples for text classification. In *ACL 2018 (Volume 2: Short Papers)*, pp. 31–36, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-2006. URL https://www.aclweb.org/anthology/P18-2006.

Fang, M., Li, Y., and Cohn, T. Learning how to active learn: A deep reinforcement learning approach. *EMNLP*, abs/1708.02383, 2017. URL http://arxiv.org/abs/1708.02383.

Fujimoto, S., van Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. *ICML*, abs/1802.09477, 2018. URL http://arxiv.org/abs/1802.09477.

Gao, J., Lanchantin, J., and Qi, Y. Mctsbug: Generating adversarial text sequences via monte carlo tree search and homoglyph attack. 2018a.

Gao, J., Lanchantin, J., Soffa, M. L., and Qi, Y. Black-box generation of adversarial text sequences to evade deep learning classifiers. *CoRR*, abs/1801.04354, 2018b. URL http://arxiv.org/abs/1801.04354.

Gong, H., Bhat, S., Wu, L., Xiong, J., and Hwu, W.-m. Reinforcement learning based text style transfer without parallel training corpus. In *ACL 2019*, pp. 3168–3180, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1320. URL https://www.aclweb.org/anthology/N19-1320.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks. *CoRR*, abs/1706.04599, 2017. URL http://arxiv.org/abs/1706.04599.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *ICML 2018*, abs/1801.01290, 2018. URL http://arxiv.org/abs/1801.01290.

Ilyas, A., Santurkar, S., Tsipras, D., Engstrom, L., Tran, B., and Madry, A. Adversarial examples are not bugs, they are features. *NeurIPS*, abs/1905.02175, 2019.

Jin, D., Jin, Z., Zhou, J. T., and Szolovits, P. Is BERT really robust? natural language attack on text classification and entailment. *CoRR*, abs/1907.11932, 2019. URL http://arxiv.org/abs/1907.11932.

Kaushik, D., Hovy, E. H., and Lipton, Z. C. Learning the difference that makes a difference with counterfactually-augmented data. *ArXiv*, abs/1909.12434, 2019.

Li, J., Monroe, W., and Jurafsky, D. Understanding neural networks through representation erasure. *CoRR*, abs/1612.08220, 2016. URL http://arxiv.org/abs/1612.08220.

Lundberg, S. M. and Lee, S.-I. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems 30*, pp. 4765–4774. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-pre pdf.

Papernot, N., McDaniel, P. D., Goodfellow, I. J., Jha, S., Celik, Z. B., and Swami, A. Practical black-box attacks against deep learning systems using adversarial examples. *CoRR*, abs/1602.02697, 2016. URL http://arxiv.org/abs/1602.02697.

Pröllochs, N., Feuerriegel, S., and Neumann, D. Learning interpretable negation rules via weak supervision at document level: A reinforcement learning approach. In *2019 NAACL: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 407–413, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1038. URL https://www.aclweb.org/anthology/N19-1038.

Rennie, S. J., Marcheret, E., Mroueh, Y., Ross, J., and Goel, V. Self-critical sequence training for image captioning. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1179–1195, 2016.

Rudin, C. Please stop explaining black box models for high stakes decisions. *Nature*, 2019. URL https://www.nature.com/articles/s42256-019-0048-x.

Sharma, S., Henderson, J., and Ghosh, J. CERTIFAI: counterfactual explanations for robustness, transparency, interpretability, and fairness of artificial intelligence models. *CoRR*, abs/1905.07857, 2019. URL http://arxiv.org/abs/1905.07857.

Smith, J. W., Everhart, J., Dickson, W., Knowler, W., and Johannes, R. Using the adap learning algorithm to forecast the onset of diabetes mellitus. In *Proceedings of the Annual Symposium on Computer Application in Medical Care*, pp. 261. American Medical Informatics Association, 1988.

SpinUp, O. Openai: Spinning up in deep rl. 2019. URL https://spinningup.openai.com/en/latest/index.html.

Tsai, Y.-T., Yang, M.-C., and Chen, H.-Y. Adversarial attack on sentiment classification. In *Proceedings of the 2019*

ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, pp. 233–240, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-4824. URL https://www.aclweb.org/anthology/W19-4824.

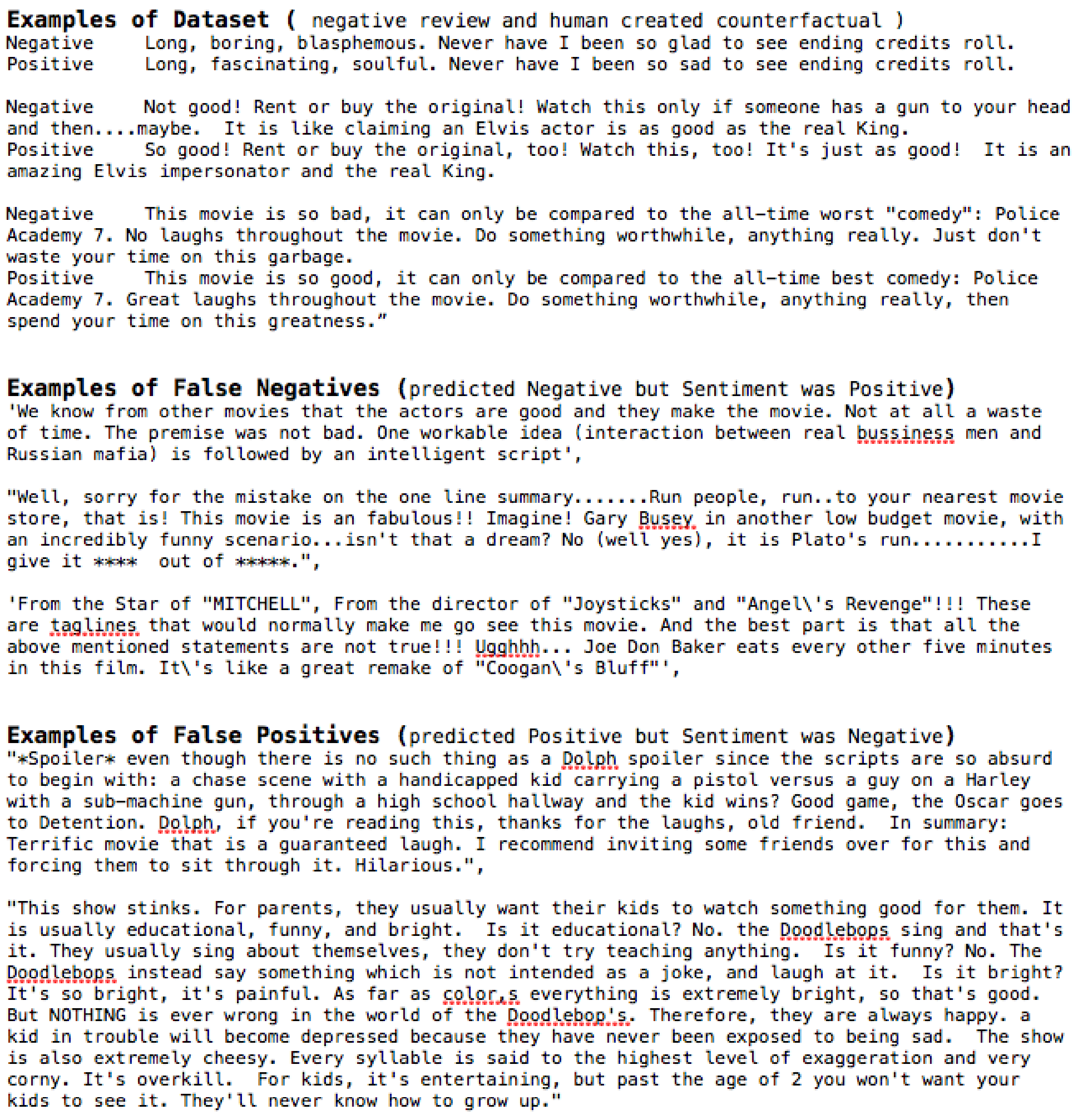
Vijayaraghavan, P. and Roy, D. Generating black-box adversarial examples for text classifiers using a deep reinforced model. *ECML PKDD*, abs/1909.07873, 2019.

Zhang, H., Lan, Y., Guo, J., Xu, J., and Cheng, X. Reinforcing coherence for sequence to sequence model in dialogue generation. In *IJCAI*, 2018.

Zhao, Z., Dua, D., and Singh, S. Generating natural adversarial examples. *ICLR*, abs/1710.11342, 2018. URL http://arxiv.org/abs/1710.11342.

5. APPENDIX

5.1. Examples of Counterfactual Text Data Set and Incorrect Predictions



Examples of Dataset (negative review and human created counterfactual)

Negative   Long, boring, blasphemous. Never have I been so glad to see ending credits roll.
Positive   Long, fascinating, soulful. Never have I been so sad to see ending credits roll.

Negative   Not good! Rent or buy the original! Watch this only if someone has a gun to your head and then....maybe. It is like claiming an Elvis actor is as good as the real King.
Positive   So good! Rent or buy the original, too! Watch this, too! It's just as good! It is an amazing Elvis impersonator and the real King.

Negative   This movie is so bad, it can only be compared to the all-time worst "comedy": Police Academy 7. No laughs throughout the movie. Do something worthwhile, anything really. Just don't waste your time on this garbage.
Positive   This movie is so good, it can only be compared to the all-time best comedy: Police Academy 7. Great laughs throughout the movie. Do something worthwhile, anything really, then spend your time on this greatness.”

Examples of False Negatives (predicted Negative but Sentiment was Positive)

'We know from other movies that the actors are good and they make the movie. Not at all a waste of time. The premise was not bad. One workable idea (interaction between real bussiness men and Russian mafia) is followed by an intelligent script',

"Well, sorry for the mistake on the one line summary.......Run people, run..to your nearest movie store, that is! This movie is an fabulous!! Imagine! Gary Busey in another low budget movie, with an incredibly funny scenario...isn't that a dream? No (well yes), it is Plato's run..........I give it **** out of *****.",

'From the Star of "MITCHELL", From the director of "Joysticks" and "Angel\'s Revenge"!!! These are taglines that would normally make me go see this movie. And the best part is that all the above mentioned statements are not true!!! Ugghhh... Joe Don Baker eats every other five minutes in this film. It\'s like a great remake of "Coogan\'s Bluff"',

Examples of False Positives (predicted Positive but Sentiment was Negative)

"*Spoiler* even though there is no such thing as a Dolph spoiler since the scripts are so absurd to begin with: a chase scene with a handicapped kid carrying a pistol versus a guy on a Harley with a sub-machine gun, through a high school hallway and the kid wins? Good game, the Oscar goes to Detention. Dolph, if you're reading this, thanks for the laughs, old friend. In summary: Terrific movie that is a guaranteed laugh. I recommend inviting some friends over for this and forcing them to sit through it. Hilarious.",

"This show stinks. For parents, they usually want their kids to watch something good for them. It is usually educational, funny, and bright. Is it educational? No. the Doodlebops sing and that's it. They usually sing about themselves, they don't try teaching anything. Is it funny? No. The Doodlebops instead say something which is not intended as a joke, and laugh at it. Is it bright? It's so bright, it's painful. As far as color,s everything is extremely bright, so that's good. But NOTHING is ever wrong in the world of the Doodlebop's. Therefore, they are always happy. a kid in trouble will become depressed because they have never been exposed to being sad. The show is also extremely cheesy. Every syllable is said to the highest level of exaggeration and very corny. It's overkill. For kids, it's entertaining, but past the age of 2 you won't want your kids to see it. They'll never know how to grow up."

5.2. Examples of Generated Counterfactual Texts via REINFORCE and REINFORCE with baseline along with the original input and human made counterfactual

Input This is surely one of the worst films ever made. Each scene is painful. You will groan at the flimsy attempts at humor, the awkward camera work, the sexism and racism, the ridiculous story line, the wooden acting. Poor Joan Bennett; she is the only one in the movie who is not an embarrassment. In all, dreadful.
WithOut CF: dist: 0.21 this is probably 2 of the greatest movies currently playing. each minute is hilarious. you will look at the clumsy attempt at comedy, the excessive screen size, the prejudice and stereotypes, the long punch lines, the bad scenes. and lucy bennett ; she is the only one in the movie who is not an embarrassment. in all, dreadful.
WithBase CF: dist: 0.0784 this is probably two of the greatest movies really ever. each moment is painful. you will groan at the flimsy attempts at humor, the awkward camera work, the sexism and racism, the ridiculous story line, the wooden acting. poor joan bennett ; she is the only one in the movie who is not an embarrassment. in all, dreadful.
Human CF: This is surely one of the best films ever made. Each scene is perfect. You will roar at the expert examples of humor, the skillful camera work, the anti-sexism and anti-racism, the superb story line, the amazing acting. Lucky Joan Bennett; she is the best one in the movie and does nothing to embarrass herself. In all, delightful.

Input I think my summary sums it up. I found it inane and stupid. I also saw the ending a mile a way. Everyone is copying that ending anymore when doing a TV/Theater crossover anymore. Sometimes, it's better to let the movie stand alone. Others, its better to forget the movie altogether. This is one of the others....
WithOut CF: dist: 0.1168 i suppose my name made it up. i thought it funny and lovely. i even saw the ending a mile a way. everyone is copying that ending anymore when doing a tv / theater crossover anymore. sometimes, it's better to let the movie stand alone. others, its better to forget the movie altogether. this is one of the others....
WithBase CF: dist: 0.3108 i know my publisher set it up. i call it scary and dangerous. i only give the writers a quarter a penny. everyone is doing that thing here when doing a magazine / newspaper work instead. so, it's hard to have the writer writing down. so, its harder to get the pieces out. this is because of the editors....
Human CF: I think my summary sums it up. I found it brilliant and thoughtful. I never saw the ending coming. Everyone is copying that ending anymore when doing a TV/Theater crossover anymore. Sometimes, it's better to let the movie stand alone. Others, its better to forget the movie altogether. This is not one of the others....

Input This movie is not very good.In fact, it is the worst Elvis movie I have seen.It has very little plot,mostly partying,beer drinking and fighting. Burgess Meredith and Thomas Gomez are wasted. I don't know why they did this movie.You could say Elvis was wasted as well,he is much,much better in "Follow That Dream."
WithOut CF: dist: 0.1326 this one is also very bad. in comparison, it is the best horror film i have made. it has very dark scenes, all smoking, everyone dancing and singing. elvis presley and selena jimenez are missing. i don't understand why they did this scene. you can tell he was crying as ever, he is far, south away in " in that world. "
WithBase CF: dist: 0.2304 this one is really very bad. in addition, it is the best horror picture i have made. it has very interesting scenes, especially dancing, people crying and kissing. bonnie gibson and shawn michaels are drunk. i don't understand why they did this video. you can imagine dean was pleased as mel, he is more, possibly disappointed in " living that way. "
Human CF: This movie is very good.In fact, it is the best Elvis movie I have seen. It has great plot, mostly partying, beer drinking and fighting. Burgess Meredith and Thomas Gomez are great. I know why they did this movie. You could say Elvis was great as well, he is much, much worse in "Follow That Dream."

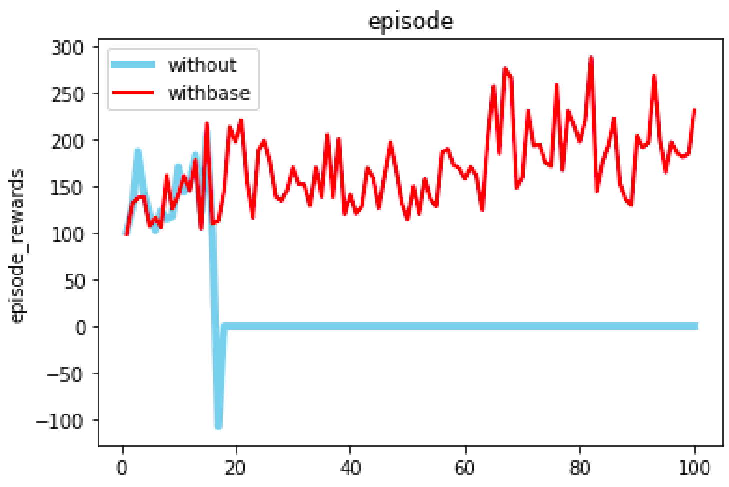


Figure 11. Example where REINFORCE without baseline gets stuck in a bad space