Text Counterfactuals via Latent Optimization and Shapley-Guided Search

Quintin Pope and **Xiaoli Z. Fern**
School of Electrical Engineering and Computer Science
Oregon State University
popeq,xfern@oregonstate.edu

Abstract

We study the problem of generating counterfactual text for a classifier as a means for understanding and debugging classification. Given a textual input and a classification model, we aim to minimally alter the text to change the model's prediction. White-box approaches have been successfully applied to similar problems in vision where one can directly optimize the continuous input. Optimization-based approaches become difficult in the language domain due to the discrete nature of text. We bypass this issue by directly optimizing in the latent space and leveraging a language model to generate candidate modifications from optimized latent representations. We additionally use Shapley values to estimate the combinatoric effect of multiple changes. We then use these estimates to guide a beam search for the final counterfactual text. We achieve favorable performance compared to recent white-box and black-box baselines using human and automatic evaluations. Ablation studies show that both latent optimization and the use of Shapley values improve success rate and the quality of the generated counterfactuals.

1 Introduction

Deep neural networks have achieved state-of-the-art performances for many natural language processing (NLP) tasks (Otter et al., 2020; Ruder et al., 2019). When applying such models in real world applications, understanding their behavior can be challenging — the ever increasing complexity of such models makes it difficult to understand and debug their predictions. A human can explain why an example belongs to a specific concept class by constructing a counterfactual of an example that is minimally altered but belongs to a different class. Contrasting the original example with its counterfactual highlights the critical aspects signifying the concept class. We study a similar approach to understand deep NLP models' classification criteria.

Given a classifier and an input text, our goal is to generate a counterfactual by making a set of minimal modifications to the text that change the label assigned by the classifier. Additionally, our goal is to understand the model's behavior when processing *naturally occurring* inputs, hence we wish to generate grammatically correct and semantically plausible counterfactuals.

Automatic generation of text counterfactuals has been studied in different settings. Qin et al. (2019) considered counterfactual story rewriting which aims to minimally rewrite an original story to be compatible with a counterfactual event. Wu et al. (2021) used a fine-tuned GPT-2 model to generate general purpose counterfactuals that are not tied to a particular classification model. Yang et al. (2020) aim to generate plausible-sounding counterfactuals that flip a classification model's decision for financial texts.

Related, textual adversaries also aim to change the model prediction (with modifications resembling natural text). The difference is that adversaries further aim to escape human detection (not changing a human's classification), whereas counterfactuals do not have such requirement.

Another line of related work is style transfer (Sudhakar et al., 2019; Wang et al., 2019; Hu et al., 2017), which aim to modify a given text according to a target style. It differs from adversary or counterfactual generation in that it seeks to fully change all style-related phrases, as opposed to minimally perturbing a text to change a classifier's decision.

White-box approaches have been widely used to generate adversaries or counterfactuals for vision tasks where the continuous inputs can be optimized to alter model predictions (Goodfellow et al., 2014; Carlini and Wagner, 2017; Neal et al., 2018). Such optimization based approaches are difficult to apply to language due to the discrete nature of text. We circumvent this difficulty by directly optimizing in the latent space of the input towards the desired classification. We then exploit the language generation capability of pre-trained language models, available for most state-of-the-art NLP models such as BERT (Devlin et al., 2019) or RoBERTa (Liu et al., 2019), to generate semantically plausible substitutions from the optimized latent representations. We further introduce Shapley values to estimate the combinatoric effect of multiple simultaneous changes, which are then used to guide a beam search to generate the final counterfactual.

Leveraging pre-trained language models to generate alternative texts has been a popular black-box approach in the recent literature on text adversaries (Li et al., 2020b; Garg and Ramakrishnan, 2020; Li et al., 2020a). Our work presents a first attempt to combine the strength of white-box optimization and the power of pre-trained language models. While Shapley values have been widely studied for the problem of feature importance (Lundberg and Lee, 2017; Sundararajan and Najmi, 2020) and data valuation (Jia et al., 2020), this is the first effort demonstrating their usefulness for text generation.

We compare our method to several white-box and black-box baselines on two different text classification tasks. Automatic and human evaluation results show that our method significantly improves the success rate of counterfactual generation, while reducing the fraction of input tokens modified and enhancing the semantic plausibility of generated counterfactuals. We also show through ablation studies that both counterfactual optimization of the latent representations and Shapley value estimates contribute to our method's strong performance.

2 Proposed Method

Problem statement. We are given a text classification model, M, an initial input token sequence, $X = \{x_0, ..., x_{n-1}\}$, with vocabulary V. Model M outputs a classification score $\hat{y} = M(X) \in (0, 1)$, representing $P(y = 1|X)$. Based on the score, a class label $y \in \{0, 1\}$ is assigned. We seek to generate a *counterfactual* of X, which is defined as a set of tokens, $X' = \{x'_0, ..., x'_{n-1}\}$, that differs from X in no more than C_{max} percent of locations, is grammatically plausible, and leads to a different classification, y'. Here C_{max} is an input parameter for maximum changes allowed, and smaller C_{max} imposes stronger restrictions.

Note that our setup assumes binary classification, but can be easily extended to multi-class scenario to generate either targeted (with specified y') or un-targeted counterfactuals (with unspecified y').

Method overview. Our method consists of three steps. First, we generate a set of candidate token substitutions for each position. Second, we evaluate the capacity of these candidate substitutions to change model classification (individually and collectively), Finally, we construct the counterfactual by beam search.

2.1 Generating candidate substitutions

We generate candidate substitutions by first performing latent space optimization and then generate substitutions from the trajectory of latent representations using a language model.

Given an input token sequence $X = \{x_0, ..., x_{n-1}\}$, we assume model M contains an embedding layer that maps this discrete input sequence into continuous embedding $E = \{e_0, ..., e_{n-1}\}$. The goal is to optimize a sparsely altered $E^{'} = \{e^{'}_0, ..., e^{'}_{n-1}\}$ such that the model will output y', a target class different from M's initial prediction y. With slight abuse of notation, let $M(E^{'})$ denote M's classification score when replacing E with E' as the input embedding, we optimize the following objective.

$$\min_{E'} CE(M(E^{'}), y') + \sum_{j=0}^{n-1} |e^{'}_j - e_j| \quad (1)$$

which minimizes the cross-entropy between $M(E^{'})$ and the desired $y^{'}$, with a LASSO regularization to favor sparse divergence from the original E.

To reduce the sensitivity to the stopping point of optimization and produce diverse candidates, we optimize E' for K steps and consider the full optimization trajectory $\{E'_k : k = 1 \cdots K\}$ to generate the candidate substitutions using the pre-trained language model associated with model M.

Directly using the pre-trained language model is problematic because it does not operate in the same latent space as model M, whose encoder has been fine tuned for the specific classification task at hand. A simple fix to this problem is to use the fine-tuned encoder of M (which is used to optimize E') and retrain the associated language modeling head.[1] This produces a language model that operates in the same space as the optimized embedding.

[1]Specifically, we retrain the language modeling head using the text for which we intend to generate counterfactuals. In our experiments this only involves 1000 data points, leading to a very fast re-training process.

Specifically, we feed each $E_k^{'}$ ($k = 1, \ldots, K$) through the encoder of M and the retrained language modeling head to generate a logit matrix T_k of size $|V| \times n$, where $T_k(s,t)$ quantifies the likelihood of observing the s-th token of the vocabulary at the t-th location given the overall context of $E_k^{'}$.

To generate K candidate substitutions for each position t, we iteratively process $T_1, \cdots, T_K$, selecting the token with the highest logit score excluding the original x_t and previous selections. Let $\mathcal{S}_t^k$ be the set of candidate substitutions for position t generated at iteration k considering T_k, it is computed as follows.

$$\mathcal{S}_t^k = \mathcal{S}_t^{k-1} \cup \underset{s \notin \mathcal{S}_t^{k-1} \cup x_t}{\text{argmax}}\; T_k(s,t) \tag{2}$$

At the end of this step, we produce a set of K candidate substitutions for each input position.

2.2 Evaluating Candidate Substitutions

In the second step, we compute a metric that measures each candidate substitution's capacity to change the classification when applied in combination with other substitutions. Toward this goal, we consider Shapley value, which was originally proposed in cooperative game theory (Shapley, 1951) and has been used to measure feature importance for model interpretability (Lundberg and Lee, 2017).

For a multi-player coalition-based game, the Shapley value of a player represents how valuable the player is as a potential coalition member. In our context, a coalition L is a set of simultaneous substitutions and value $V(L)$ is measured by L's capacity to change model M's prediction. Let X_L denote the input generated by applying all substitutions[2] in L to X, and $M(X_L)$ be M's prediction score. We define $V(L)$ to be $M(X_L) - M(X)$ if we wish to flip a negative prediction and $M(X) - M(X_L)$ otherwise.

The Shapley value of a single substitution s measures the expected marginal value of adding s to a coalition not already containing s. To ensure computational tractability, we constrain the size of the coalition to be a fixed value c_s. As such, coalitions of any other sizes will have value zero. Conceptually this measures the potential value of substitution s when we modify exactly c_s tokens.

Under this setting, it is straightforward to show that the Shapley value of a single substitution s can be estimated by the following equation:

$$SV(s) = \frac{1}{|\mathcal{L}_s|} \sum_{L \in \mathcal{L}_s} V(L) - \frac{1}{|\mathcal{L}_{/s}|} \sum_{L \in \mathcal{L}_{/s}} V(L) \tag{3}$$

$\mathcal{L}_s$ ($\mathcal{L}_{/s}$) denotes the set of coalitions containing (not containing) s that satisfy the size constraint.

Fully enumerating $\mathcal{L}_s$ and $\mathcal{L}_{/s}$ to compute Equation 3 is infeasible in most situations. We use two strategies to improve efficiency. First, we apply filtering to remove unimportant locations from further consideration. We adapt the NormGrad saliency method described by Rebuffi et al. (2020) to text and use the following gradient-based saliency score.

$$\text{saliency}(i) = |(\nabla_{e_i} \hat{y}) \odot e_i|^2 \tag{4}$$

where $\nabla_{e_i} \hat{y}$ denotes the gradient of the original classification score $\hat{y}$ with respect to e_i, the embedding of the i-th token, and $\odot$ represents the Hadamard product (elementwise multiplication).

Our second strategy is to approximate the Shapley values by sampling in the space of allowed substitutions. Suppose we want to evaluate each substitution w times on average and there are a total of N_s substitutions to be evaluated. It is interesting to note that we do not need $N_s \cdot w$ evaluations since each evaluation simultaneously contributes to the estimates of all c_s substitutions that it contains.

We apply filtering to consider only the top $C_{max} \times n$ locations, and fix the coalition size to be 50% of that ($c_s = 0.5 \times C_{max} \times n$). Each important location contributes K candidate substitutions. For input of length n, there are $C_{max} \times K \times n$ total substitutions to evaluate. Because each coalition evaluation covers $0.5 \times C_{max} \times n$ substitutions, to evaluate each substitution w times on average, we need to evaluate $2 \times w \times K$ coalitions, which is independent of n and C_{max}.

2.3 Constructing the Counterfactual

In the final step, we search for the optimal subset of substitutions via breadth-first beam search. The search space covers all possible subsets of non-conflicting substitutions, each subset corresponding to a unique candidate counterfactual.

We initialize the beam with the root of the search tree, which is the empty subset. At each iteration, we expand a node in the beam with a successor function returning b successors, each adding a single substitution. For a given search node, denoted

[2]By definition, L must not contain multiple substitutions to the same location, which will create a conflict.

by its subset L, we construct b successors by selecting b substitutions with the best Shapley values that do not conflict in location with any $s \in L$ or introduce a redundant subset.

We then evaluate each successor node by applying its substitutions to the original input X and computing model M's output on the resulting X'. We rank all successors based on the model's score for the desired class y' minus the fraction of tokens modified by the successor in question and populate the new beam with the top b candidates.

We limit the search depth to be $C_{max} \times n$, constraining our method to never modify more than C_{max} percent of the input tokens. During search, if we generate a candidate that M classifies as y', we stop immediately and return that candidate as our final output. As such, the time we spend for beam search depends on how quickly we find a successful counterfactual.

2.4 Summary of approach

We summarize our method as text **C**ounterfactuals via **L**atent **O**ptimization and **S**hapley-guided **S**earch (CLOSS). CLOSS has three primary hyperparameters: K, the number of candidate substitutions generated per token locations; w, the average number of times we wish to evaluate each substitution; b, the beam width of the beam search that constructs the final counterfactual. The default values are $K = 30$, $w = 5$, $b = 15$. The impact of these parameters will be explored in the experiments.

3 Empirical Evaluation

To evaluate our proposed method, we consider two different text classification tasks: sentiment classification and natural language inferences.

3.1 Experimental Setup

Data sources. We use the IMDB dataset (Maas et al., 2011) for sentiment classification. This is a binary classification dataset based on movie reviews from IMDB. For the natural language inference task, we use the QNLI dataset (Rajpurkar et al., 2016), which is a binary task derived from the Stanford question answering dataset. Each example contains a question and a context. The classifier must determine if the context answers the question.

Following the evaluation scheme used by Li et al. (2020a), we sample 1000 random data points from IMDB of length less than or equal to 100 words as our "short" IMDB data. We do not filter the QNLI dataset. The average word counts for short IMDB and QNLI are shown in Table 1 (row 1).

Classification models. For each task, we consider two classification models, RoBERTa (Liu et al., 2019) and BERT (Devlin et al., 2019), trained by TextAttack (Morris et al., 2020). We report the performance of both models in Table 1 (rows 2-3).

	IMDB short	QNLI
Avg. words	57.4	37.2
RoBERTa acc.	0.971	0.934
BERT acc.	0.974	0.908

Table 1: Model and dataset statistics

Evaluation criteria. We consider the following performance metrics that measure the ability of a method to successfully generate counterfactuals and the quality of the generated counterfactuals.

- **Failure rate (%F)**: the percent of inputs for which the method fails to change the model's prediction.
- **Fraction of tokens changed (%C)**: the average token modification rate among successfully generated counterfactuals.
- **BLEU**: the average BLEU score between successfully generated counterfactuals and their original inputs.
- **Perplexity (P)**: Following Zang et al. (2020), we use the exponentiated language modeling loss of GPT-2 (Radford et al., 2019) to compute perplexity to score linguistic plausibility.

Baselines. We compare against adversarial baselines because we were unable to find counterfactual methods with open-source implementations. We carefully identified a set of baselines closely related to CLOSS with respect to the methodology, specifically focusing on black box methods that leverage pretrained language models (BERT-Attack, BAE), and a white-box method using gradients and beamsearch (Hot-Flip). Unless otherwise stated, we use the implementations in the TextAttack package (Morris et al., 2020). All black-box methods use some saliency measure to prioritize substituting important tokens. While CLOSS estimates saliency from the gradient, the black-box baselines use leave-a-token-out estimates, i.e., by removing or masking a token.

BERT Adversarial Example (BAE) (Garg and

Ramakrishnan, 2020) is a black-box method that generates potential substitutions by masking out input tokens and using the pre-trained BERT language model to suggest replacements.

BERT-Attack (Li et al., 2020b) is also a black-box method. It generates substitutions by feeding the entire *unmasked* input into the BERT language model to suggest replacements.

Textfooler (Jin et al., 2020) is a black-box method that uses word embeddings by Mrkšić et al. (2016) to generate substitutions by selecting the vocabulary tokens whose embeddings have the highest cosine similarity with the original token.

Probability Weighted Word Saliency (PWWS) (Ren et al., 2019) is a black-box method that uses WordNet synonyms as potential substitutions to preserve the semantic content.

HotFlip (Ebrahimi et al., 2018) is a white-box method that uses model gradients to estimate the impact of every possible token substitution on the model's classification and then applies beam search to generate counterfactuals. HotFlip implemented in TextAttack only works for word or character-level classifiers, not WordPiece (Schuster and Nakajima, 2012) classifiers like RoBERTa and BERT. Hence we implemented our own version of HotFlip[3] to be exactly comparable to our method.

Adaptation of adversarial baselines for fair comparison. Adversaries differ from counterfactuals in that they additionally seek to retain the text's "true" class, relative to human judgement. In this regard, generating adversaries is more difficult than generating counterfactuals. Here we adapt the (adversary-generating) baselines to generate counterfactuals, thereby allowing fair comparison.

All original baseline implementations employ certain heuristic constraints to preserve the original semantic content (and thus, true class) of the input. Most methods require a minimum cosine similarity between the Universal Sentence Encoder (Cer et al., 2018) (USE) representations of the modified text with the original input. Additional heuristics include not substituting stop words and requiring substitutions to have the same part-of-speech as the original. These heuristics directly modify the search space for a generation method, and thus can impact both the success and quality of counterfactual generation.

[3]HotFlip can generate insertion/deletion edits in addition to substitutions. Our implementation only considers substitutions to be directly comparable with our method.

For CLOSS and our implementation of HotFlip, we do not employ such heuristics. Additionally, we created an unconstrained version of the TextAttack (Morris et al., 2020) implementations (denoted by suffix '-U') of all other baselines by removing the adversarial constraints. Arguably, PWWS-U and TextFooler-U remain more constrained than CLOSS because they only use synonyms (WordNet and embedding based, respectively) for substitutions. However, the search spaces of BAE-U, BERT-Attack-U, and HotFlip are fully comparable to CLOSS.

TextAttack by default ignores any input misclassified by the model M, because the concept of an "adversarial" example does not readily extend to misclassified inputs. For counterfactual generation, we do not have this concern. Hence our evaluation seeks to generate a counterfactual that flips the model's classification regardless of its correctness.

Baseline parameters. For HotFlip, we consider two versions: a default version (HotFlip D) that uses parameters suggested by Ebrahimi et al. (2018) and an optimized version (HotFlip O), where the parameters and search procedure are optimized for performance. See Appendix 1 for details of our HotFlip implementations. For all other baselines, we use the default parameters from TextAttack, which are the recommended parameters from the original papers.

3.2 Results

We report the results of all methods for short IMDB and QNLI in Table 2. Note that BERT-Attack restrains manipulations of multi-token words in a manner that is computationally intractable on our datasets; thus we do not report performance for the original BERT-Attack. Here we limit all methods to change no more than 15% of tokens by setting $C_{max} = 0.15$. The impact of different C_{max} values will be explored later in Figure 1.

Comparing to white-box baseline. The default HotFlip implementation substantially underperforms by all measures. Optimizing the parameters and search procedure for HotFlip leads to greatly improved performance. Comparing CLOSS with HotFlip O, we note that the failure rate of our method is slightly worse for QNLI and slightly better for IMDB short. For both datasets, the counterfactuals generated by CLOSS have fewer modifications and higher BLEU scores. The most striking difference is in the perplexity score, where

	IMDB short							
	RoBERTa				BERT			
Method	%F	%C	BLEU	P	%F	%C	BLEU	P
CLOSS	**4.2**	**3.1**	**0.92**	**72.4**	**4.1**	**2.8**	**0.93**	**98.9**
HotFlip D	37.0	6.5	0.86	145	22.8	5.2	0.89	140
HotFlip O	7.1	5.1	0.88	122	4.5	4.18	0.90	129
BAE	69.4	4.6	0.86	110	67.5	4.0	0.88	136
PWWS	14.6	5.9	0.83	96	14.0	4.7	0.86	125
TextFooler	22.3	6.3	0.82	91.5	31.9	5.7	0.83	132
BAE-U	16.6	5.7	0.85	107	25.1	4.9	0.87	141
Bert-Attack-U	6.3	4.4	0.90	78.2	22.2	4.7	0.88	120
PWWS-U	12.1	5.9	0.83	102	11.7	4.6	0.86	134
TextFooler-U	12.7	5.7	0.85	93.9	21.0	5.2	0.86	142
	QNLI							
	RoBERTa				BERT			
Method	%F	%C	BLEU	P	%F	%C	BLEU	P
CLOSS	5.1	**3.3**	**0.92**	92.4	3.5	**3.3**	**0.92**	**143**
HotFlip D	18.8	4.7	0.90	130	19.1	4.4	0.90	174
HotFlip O	**3.4**	4.0	0.90	125	**2.1**	3.8	0.91	178
BAE	34.6	3.7	0.87	94.4	33.2	4.0	0.87	175
PWWS	22.7	4.2	0.87	95.1	14.9	4.4	0.86	184
TextFooler	19.4	4.6	0.86	90.1	13.1	4.9	0.86	176
BAE-U	6.8	4.2	0.87	104	6.0	4.1	0.88	178
Bert-Attack-U	6.7	4.0	0.89	**87.1**	4.9	3.8	0.90	156
PWWS-U	16.0	4.3	0.87	107	8.7	4.3	0.87	201
TextFooler-U	7.4	4.4	0.87	101	4.6	4.3	0.88	180

Table 2: Comparison of CLOSS with baselines on the short IMDB and QNLI data. 'U' indicates unconstrained version of the baselines. Our implementation of HotFlip uses exactly the same set of constraints as CLOSS. CLOSS values are averaged over three runs.

CLOSS wins by a large margin. This is not surprising as HotFlip does not care about the semantic plausibility of the generated sentences whereas our method uses the language model to propose semantically plausible substitutions. Note that GPT-2 and RoBERTa are cased models, while BERT is uncased. This explains BERT's higher perplexity.

Comparing to black-box methods. We first observe that the heuristic constraints used by these methods have a drastic impact on the performance. Specifically, by removing these constraints, the failure rates of all methods are much reduced. However, the resulting counterfactuals tend to have lower quality, indicated by increased perplexity. Comparing CLOSS to both variants, we see that our method was able to achieve a highly competitive failure rate with few edits. CLOSS also achieves the lowest perplexity in most cases with the exception of RoBERTa model on QNLI, where BERT-Attack-U have slightly lower perplexity.

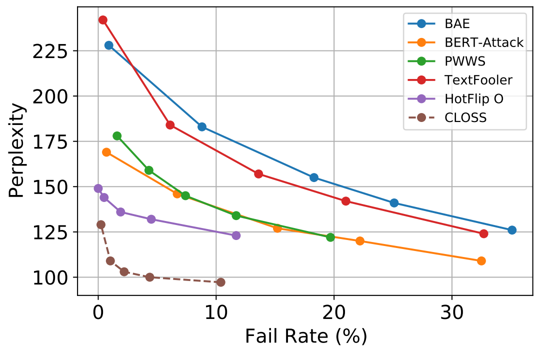
3.3 Impact of Varying C_{max}

We considering different C_{max} values including 0.1, 0.15, 0.2, 0.3 and 0.5. Figure 1 plots the perplexity against failure rate for different values of C_{max}[4]. Increasing C_{max} allows methods to change more input tokens, reducing their failure rates. However, higher C_{max} also leads to greater distortion of the input, raising the perplexity. Thus, methods with better perplexity/fail rate tradeoffs have curves that fall closer to the lower-left corner of the plots. In this regard, CLOSS has the best performance on all comparisons, except for against BERT-Attack on RoBERTa QNLI, where the two methods appear comparable.

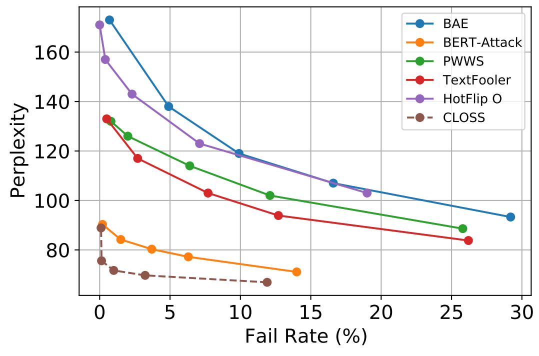
3.4 Ablation studies

We consider three ablated versions of CLOSS. **CLOSS-EO** removes the optimization step and in-

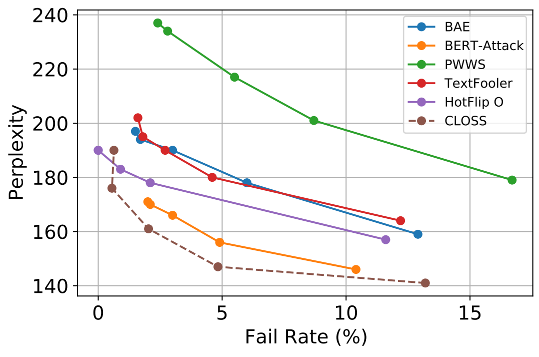
[4]HotFlip D is excluded due to its poor performance and to preserve the graph scaling for the rest of the methods.



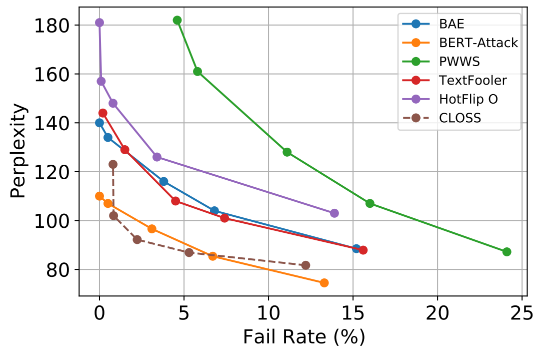
(a) BERT IMDB



(b) RoBERTa IMDB



(c) BERT QNLI



(d) RoBERTa QNLI

Figure 1: Plots of perplexity against failure rate as the maximum allowed percent of tokens changed (C_{max}) varies. Values for C_{max} are 10%, 15%, 20%, 30% and 50%. CLOSS values are averaged over three runs.

stead generates potential substitutions by feeding the original input into the default pre-trained language model associated with model M.

CLOSS-RTL skips retraining the language modeling head and uses the language modeling head of the pretrained language model. As a result, the language modeling head for this ablation has a different latent space compared to the fine-tuned encoder of classifier M.

CLOSS-SV removes the Shapley value estimates of each substitution's impact on classification. Instead, we priority substitutions during beam search based on the token saliency (Eq. 4).

We compare the performance of CLOSS with its ablations in Table 3. Here we omit BLEU score because it strongly correlates with %C in Tables 2,

Effect of embedding optimization. By removing the optimization step, CLOSS-EO has significantly more failures, but lower perplexity. This is not surprising because optimizing the embedding increases the chance to flip the prediction but carries the risk of producing "unnatural" embeddings that lie outside the space of texts previously observed by the language model. This also suggests that CLOSS-EO can be a good candidate for scenarios where "naturalness" of the text is critical.

Effect of retraining language modeling head. It is interesting to note that CLOSS-RTL has comparable perplexity to CLOSS, but a higher failure rate. We believe this is because the retrained language modeling head can generate tokens that better match the data distribution of IMDB and QNLI (but not of English text in general), i.e., the distribution of tokens to which the classifier M is sensitive.

Effect of Shapley values. By removing Shapley value estimates, CLOSS-SV sees substantial degradations in all measures, suggesting critical importance of this step to our method.

3.5 Computational Considerations

The estimation of Shapley value for CLOSS incurs a substantial cost in terms of number of queries to the given model. Indeed, the number of queries used by CLOSS can be significantly higher than some baselines[5]. This section takes a closer look at

[5]We report the average number of model queries used by CLOSS and the baselines in Table 4 in the Appendix. In practice, CLOSS, implemented without parallelization, can

	IMDB						QNLI					
	RoBERTa			BERT			RoBERTa			BERT		
Method	%F	%C	P	%F	%C	P	%F	%C	P	%F	%C	P
CLOSS	**4.2**	**3.13**	72.4	**4.1**	**2.76**	98.9	**5.1**	**3.33**	92.4	**3.5**	**3.31**	143
CLOSS-SV	9.4	5.73	84.5	11.6	5.06	116	7.3	5.13	108	6.4	5.05	159
CLOSS-EO	7.3	3.25	**63.3**	8.4	3.17	**94.9**	7.2	3.51	**72.2**	6.1	3.51	**122**
CLOSS-RTL	5.5	3.2	73.7	7.5	2.9	102	7.9	3.7	94.7	5.7	3.4	136

Table 3: Ablation results on IMDB and QNLI. Values are averaged over three runs.

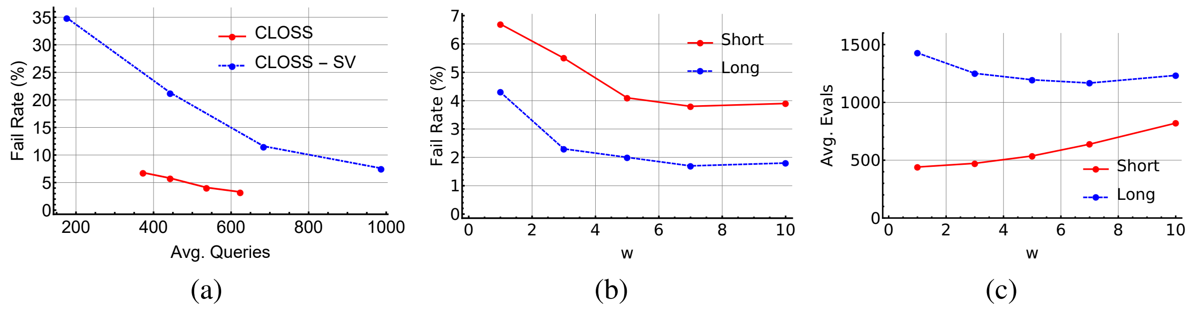


Figure 2: (a) Plot of failure rate of CLOSS and CLOSS-SV as a function of the number of model queries for short IMDB. Beam width ranges from 5 to 20 for both approaches. (b) Plot of failure rate of CLOSS in flipping BERT's prediction as a function of w for both short and long IMDB. (c) Plot of number of BERT model queries used by CLOSS as a function of w for both short and long IMDB.

the computational trade-offs surrounding Shapley value estimations.

Shapley or not. Given the computational cost of estimating Shapley values, would a more thorough search (e.g., using larger beam width) remove the need for computing Shapley values? To explore this question, we consider different beam width $b = 5, 10, 15$ and 20, and plot the resulting failure rates of CLOSS and CLOSS-SV against the number of queries to the model for short IMDB and BERT in Figure 2 (a).[6] The figure shows that even with larger beam width and higher number of queries, the performance of CLOSS-SV still trails behind CLOSS. It also shows that the Shapley value guided search reduces CLOSS's sensitivity to the beam width b both in terms of failure rate and the number of queries needed.

Precision of Shapley. The Shapley value is estimated via sampling and the sampling rate is controlled by the parameter w. Intuitively, larger w leads to more accurate Shapley values, but incurs higher computation cost. We explore how sensitive our method is to the parameter w in Figure 2(b&c).

Specifically, Figure 2(b) plot the failure rate of CLOSS in flipping BERT's prediction for both short and long IMDB. We see that as long as w is reasonably large ($\geq$5), the performance is fairly robust. Note that other measures like perplexity and BLEU score show similar trends, which are shown in Figure 3 in the Appendix.

Figure 2(c), on the other hand, plots the average number of queries to model M required with w from 1 to 10. We see an interesting phenomenon for long IMDB where increased w actually leads to decreased number of queries. This may appear counter-intuitive at first sight, it actually demonstrates the power of good Shapley value estimates in speeding up the search. This phenomenon, however, was not observed for the short IMDB data, likely due to the substantially smaller search space thanks to the shorter input length.

4 Human and Qualitative Evaluation

Human evaluations. For human evaluations, we choose to compare CLOSS against Bert-Attack and HotFlip O, the two baselines performing the best in perplexity and flip rate respectively.

We randomly selected 100 original texts from IMDB for evaluation with the restriction that all three methods must successfully flip the classification changing 15% or less of the original tokens. Additionally, we exclude texts with more than 50 tokens to ease the burden on evaluators. Using the

generate counterfactuals for typical inputs in seconds.

[6]Figures for RoBERTa and QNLI are similar, thus omitted.

BERT classifier, We apply BERT-Attack, CLOSS and HotFlip to generate counterfactuals for each input. Eight human evaluators are each assigned 25 original texts and asked to rank (ties allowed) the three counterfactuals in order of grammatical correctness. Each input is evaluated by two evaluators, the inter-evaluator agreement per pairwise comparison is 75.4%. Human evaluators ranked CLOSS competitively with BERT-Attack and HotFlip, assigning average ranks of 1.54 to CLOSS, 1.68 to Bert-Attack and 2.50 to HotFlip. The difference between CLOSS and HotFlip is statistically significant (one-sided sign test, p-value< 0.0001).

Qualitative analysis of generated text. Inspecting the generated coutnerfactuals, we observe some interesting patterns, summarized below. See Appendix (Tables 5 and 6) for specific examples.

For the IMDB dataset, CLOSS often changes one or two sentiment words while the rest of the input still supports the original prediction. This suggests that the model may be triggered by a few sentiment words, ignoring most input. Identifying such critical substitutions will allow us to inspect the patterns of these "triggers" to reveal the weakness of the classifier. We also observe that when the model misclassifies, it often takes little change to correct the model, which helps debug the mistake.

Sometimes CLOSS introduces synergistic changes where each change's capacity to influence the classification seems contingent on the other. Finally, CLOSS sometimes distorts sentiment-phrases into non-words to remove their impact on classification, possibly making up for the lack of ability to remove words.

For the QNLI dataset, unsurprisingly, we note that changing from entailment to non-entailment is far easier than the opposite (see Figures 5(c,d) in Appendix), and often requires changing only a few words shared by the Question and Context. Conversely, CLOSS can sometimes change non-entailment to entailment by introducing some shared word(s). This suggests that the model relies heavily on overlapping words to decide entailment.

More detailed analysis can be found in the Appendix, including how CLOSS's changes are distributed among part of speech tags (Figure 4) and a failure analysis for CLOSS (A.7, Table 7).

5 Conclusion

We are motivated by how humans use counterfactuals to explain the concept of a class and seek to automatically generate counterfactual text input as a means to understand a deep NLP model and its definition of class. We assume full white-box access to the given model and perform optimization in the latent space to maximize the probability of predicting a target class. We then map from the optimized latent representation to candidate token substitutions using a language model. A key novelty of CLOSS is using Shapley values to estimate the potential of a token substitution in changing the model's prediction when used in combination with other substitutions. The Shapley value is then used to guide a breadth-first beam search to generate the final counterfactual. Through both automatic and human evaluations, we show that CLOSS achieves highly competitive performance both in terms of the success rate of generating counterfactuals as well as the quality of the generated counterfactuals.

Our approach has several limitations. As a white-box approach, we require full access to the model, which can be restrictive in practical applications. Our approach currently only considers substitutions, excluding deletions and insertions. Finally, our method is only applicable to models that are based on pre-trained language models. Future work will adapt CLOSS to adversarial and black box settings. We also hope to improve the efficiency of CLOSS via more efficient Shapley value estimation (Chen et al., 2018; Jia et al., 2020).

Acknowledgements

This work was partially supported by DARPA under grant N66001-17-2-4030.

References

Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. IEEE.

Daniel Cer, Yinfei Yang, Sheng yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder.

Jianbo Chen, Le Song, Martin J. Wainwright, and Michael I. Jordan. 2018. L-shapley and c-shapley: Efficient model interpretation for structured data.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. Hotflip: White-box adversarial examples for text classification.

Siddhant Garg and Goutham Ramakrishnan. 2020. Bae: Bert-based adversarial examples for text classification.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In *International Conference on Machine Learning*, pages 1587–1596. PMLR.

Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nick Hynes, Nezihe Merve Gurel, Bo Li, Ce Zhang, Dawn Song, and Costas Spanos. 2020. Towards efficient data valuation based on the shapley value.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is bert really robust? a strong baseline for natural language attack on text classification and entailment.

Dianqi Li, Yizhe Zhang, Hao Peng, Liqun Chen, Chris Brockett, Ming-Ting Sun, and Bill Dolan. 2020a. Contextualized perturbation for textual adversarial attack.

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020b. Bert-attack: Adversarial attack against bert using bert.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Scott Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

John X. Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Lina Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting word vectors to linguistic constraints.

Lawrence Neal, Matthew Olson, Xiaoli Fern, Weng-Keen Wong, and Fuxin Li. 2018. Open set learning with counterfactual images. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

D. W. Otter, J. R. Medina, and J. K. Kalita. 2020. A survey of the usages of deep learning for natural language processing. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–21.

Lianhui Qin, Antoine Bosselut, Ari Holtzman, Chandra Bhagavatula, Elizabeth Clark, and Yejin Choi. 2019. Counterfactual story reasoning and generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5046–5056.

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

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text.

Sylvestre-Alvise Rebuffi, Ruth Fong, Xu Ji, and Andrea Vedaldi. 2020. There and back again: Revisiting backpropagation saliency methods.

Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1085–1097, Florence, Italy. Association for Computational Linguistics.

Sebastian Ruder, Matthew E. Peters, Swabha Swayamdipta, and Thomas Wolf. 2019. Transfer learning in natural language processing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials*, pages 15–18, Minneapolis, Minnesota. Association for Computational Linguistics.

M. Schuster and K. Nakajima. 2012. Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152.

Lloyd Shapley. 1951. Notes on the n-person game – ii: The value of an n-person game.

Akhilesh Sudhakar, Bhargav Upadhyay, and Arjun Maheswaran. 2019. "transforming" delete, retrieve, generate approach for controlled text style transfer. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3269–3279, Hong Kong, China. Association for Computational Linguistics.

Mukund Sundararajan and Amir Najmi. 2020. The many shapley values for model explanation.

Ke Wang, Hang Hua, and Xiaojun Wan. 2019. Controllable unsupervised text attribute transfer via editing entangled latent representation.

Tongshuang Wu, Marco Tulio Ribeiro, Jeffrey Heer, and Daniel S. Weld. 2021. Polyjuice: Automated, general-purpose counterfactual generation.

Linyi Yang, Eoin M. Kenny, Tin Lok James Ng, Yi Yang, Barry Smyth, and Ruihai Dong. 2020. Generating plausible counterfactual explanations for deep transformers in financial text classification.

Yuan Zang, Fanchao Qi, Chenghao Yang, Zhiyuan Liu, Meng Zhang, Qun Liu, and Maosong Sun. 2020. Word-level textual adversarial attacking as combinatorial optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6066–6080, Online. Association for Computational Linguistics.

A Appendix

A.1 Optimized HotFlip.

For each candidate in original HotFlip's beam search, we score every possible single-token substitution by using gradients to estimate the substitution's impact on the classification. The score of a candidate counterfactual is the sum of the scores of each individual substitution introduced by the candidate.

These scores form a surrogate value function, which the beam search aims to maximize. At each step of the beam search, we can generate the successors (children) for each current beam members (parent) by applying a single substitution to any location in the parent text.

In our optimized HotFlip, we change the search procedure to promote diversity in the beam search by requiring every child generated off a common parent modify distinct locations in the text. We observe that this small modification substantially boost the performance HotFlip's performance. We also increase the beam size from the 10 suggested by Ebrahimi et al. (2018) to 100. Note that the original HotFlip's parameters are designed for character-level modification, which has a substantially smaller space of possible substitutions for each location. This might explain the the poor performance of HotFlip D, and the need to modify the search procedure for token-level generation.

A.2 Average Number of Queries

In Table 4, we present the average number of queries to the given model for CLOSS and baseline methods. We show two numbers per cell, where the first number is the average query number of all attempts (success or fail). The second number is the average number of queries for successful trials that modify 15% or less of input tokens.

A.3 Performance when varying w

In Figure 3, we plot the other performance measures including %C, BLEU and Perlexity, as a function of parameter w.

A.4 Qualitative Analysis

In tables 5 and 6, we present examples of counterfactuals generated by CLOSS that highlight interesting patterns we notice.

A.5 Part of Speech Changes

In Figure 4, we show the percentages of total changes that occur in each part of speech type. We split the results by direction of change. Note that for IMDB, CLOSS tends to modify adjectives more when flipping from negative to positive compared to flipping from positive to negative.

When flipping entailment to non-entailment, CLOSS is more likely to modify nouns compared to flipping non-entailment to entailment. This may re

A.6 Distribution over Percent of Tokens Changed

Figure 5 contains histograms showing the distribution of percent of tokens changed over successfully generated counterfactuals. Note that flipping entailment to non-entailment requires fewer changes than the reverse.

A.7 Error Analysis

We explore potential sources of counterfactual generation failure in table 7 by significantly increasing the computational resources devoted to certain steps of CLOSS and recording the resulting generation failure rate (%F).

Even greatly increasing w does not reduce %F significantly. In comparison, increasing beam width is more effective, especially in regards to IMDB. The most effective interventions are to increase either K or render all tokens salient and scale w in proportion to the associated increase in potential substitutions. Note that when we increase K without changing w, the compute spent on estimating Shapley values scales linearly with K.

	IMDB		QNLI	
Method	RoBERTa	BERT	RoBERTa	BERT
CLOSS	558|485	537|457	455|405	424|389
HotFlip D	65.3|47.5	51.7|38.0	28.7|22.4	27.0|19.9
HotFlip O	305|265	234|205	90.0|82.3	71.0|66.7
BAE	142|111	138|110	94.7|81.7	97.1|85.7
PWWS	483|466	475|455	246|236	238|236
Textfooler	259|190	298|171	144|102	127|104
BAE-U	424|285	494|252	172|148	180|142
BERT-Attack-U	277|231	451|262	172|149	182|140
PWWS-U	575|561	568|549	299|290	290|287
Textfooler-U	358|272	426|259	181|152	179|145

Table 4: Average queries per sample for CLOSS and baselines.

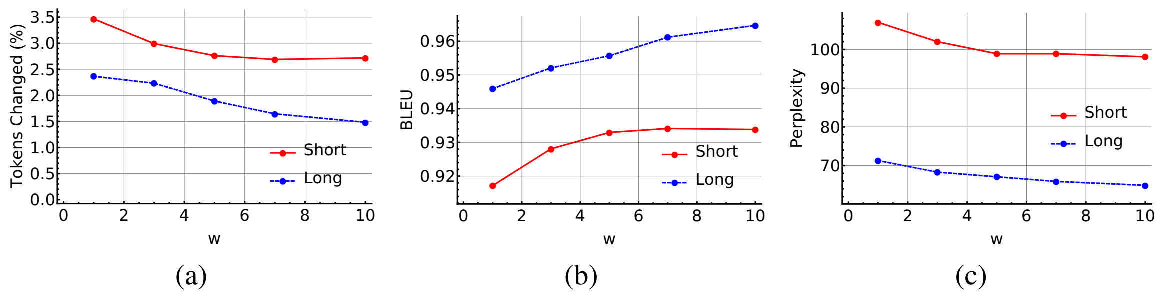


Figure 3: (a) Plot of average percent tokens changed by CLOSS as a function of w for both short and long IMDB. (b) Plot of CLOSS average BLEU score as a function of w for both short and long IMDB. (c) Plot of CLOSS average perplexity as a function of w for both short and long IMDB.

These results suggest failures in the beam search are more of a bottleneck on performance than failing to identify useful substitutions with Shapley values.

However, we can significantly improve performance by increasing both the pool of potential substitutions and the compute spent on estimating Shapley values. This implies many generation failures happen because the pool of potential substitutions the default CLOSS hyperparameters are able to search through does not contain substitutions able to flip the classification.

Description	Text
(a) We can flip the class by changing a small fraction of the sentiment regions.	**Old:** Ruth Gordon is one of the more sympathetic killers that Columbo has ever had to deal with. And, the plot is ingenious all the way around. This is one of the best Columbo episodes ever. Mariette Hartley and G. D. Spradlin are excellent in their supporting roles. And Peter Falk delivers a little something extra in his scenes with Gordon. **New:** Ruth Gordon is one of the more sympathetic killers that Columbo has ever had to deal with. And, the plot is ingenious all the way around. This is one of the worse Columbo episodes ever. Mariette Hartley and G. D. Spradlin are excellent in their supporting roles. And Peter Falk delivers a little something extra in his scenes with Gordon. **Old:** ruth gordon is one of the more sympathetic killers that columbo has ever had to deal with. and, the plot is ingenious all the way around. this is one of the best columbo episodes ever. mariette hartley and g. d. spradlin are excellent in their supporting roles. and peter falk delivers a little something extra in his scenes with gordon. **New:** ruth gordon is one of the more sympathetic killers that columbo has ever had to deal with. and, the plot is ingenious all the way around. this is one of the worst columbo episodes ever. mariette hartley and g. d. spradlin are excellent in their supporting roles. and peter falk delivers a little something extra in his scenes with gordon.
(b) We sometimes see synergistic changes where each change's capacity to influence the classification seems contingent on the other.	**Old:** Excellent documentary that still manages to shock and enlighten. Unfortunately, times haven't changed much since this was made and it is thus an important piece for all freedom-conscious Americans to see. **New:** Very pathetic that still manages to shock and enlighten. Unfortunately, times haven't changed much since this was made and it is thus an important piece for all freedom-conscious Americans to see. **Old:** I love all his work but this looks like nothing.. sorry.. This looks more like a "David Lynch copycat". I think people like it only because "it's from David Lynch". **New:** I love all his work but this hits like everything.. sorry.. This looks more like a "David Lynch copycat". I think people like it only because "it's from David Lynch".

Description	Text
(c) RoBERTa incorrectly classified text as positive. Flipping to negative requires little changes.	**Old:** Some good movies keep you in front of the TV, and you are dying to see the result. This movie does not have highs and lows. It simply describes a young girl's family life in Africa. People come and go, the weather and the background are all the same. **New:** Some decent movies keep you in front of the TV, and you are dying to see the result. This movie does not have highs and lows. It simply describes a young girl's family life in Africa. People come and go, the weather and the background are all the same.
(d) BERT classifies as negative. Greater changes required to flip to positive.	**Old:** some good movies keep you in front of the tv, and you are dying to see the result. this movie does not have highs and lows. it simply describes a young girl's family life in africa. people come and go, the weather and the background are all the same. **New:** some good movies keep you in front of the tv, and you are loving to see the result. this movie does not lack highs and lows. it simply describes a young girl's family life in africa. people come and go, the weather and the background are all the same.
(e) Sometimes distorts words/grammar; Note how CLOSS removes "I loved this" by convering "loved this" into "lovedoo", thereby removing the original's positive sentiment	**Old:** I loved this mini series. Tara Fitzgerald did an incredible job portraying Helen Graham, a beautiful young woman hiding, along with her young son, from a mysterious past. As an anglophile who loves romances... this movie was just my cup of tea and I would recommend it to anyone looking to escape for a few hours into the England of the 1800's. I also must mention that Toby Stephens who portrays the very magnetic Gilbert Markham is reason enough to watch this wonderful production. **New:** I lovedoo mini series. Tara Fitzgerald did an incredible job portraying Helen Graham, a beautiful young woman hiding, along with her young son, from a mysterious past. As an anglophile who loves romances... this movie was just my cup of tea and I would recommend it to anyone looking to escape for a few hours into the England of the 1800's. I also must mention that Toby Stephens who portrays the very magnetic Gilbert Markham does reason enough to watch this dreadful production.
(f) Non-words can significantly change sentiment classification. "thisecrated" doesn't seem particularly sentiment-related, yet it can flip the classification of this otherwise very positive review.	**Old:** absolutely fantastic! whatever i say wouldn't do this underrated movie the justice it deserves. watch it now! fantastic! **New:** absolutely fantastic! whatever i say wouldn't do thisecrated movie the justice it deserves. watch it now! fantastic!

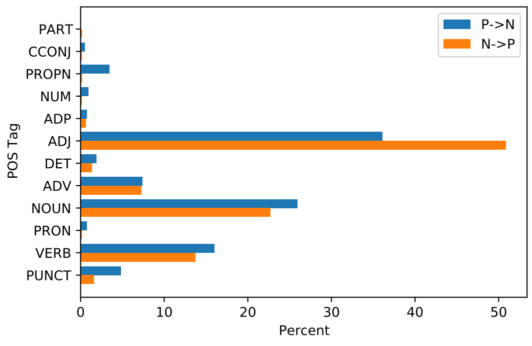
Table 5: Example IMDB counterfactuals generated by CLOSS. Each row demonstrates an interesting pattern of behavior we observed. We use green to highlight words whose changes flip the text to positive and red for changes that flip texts to negative.

Description	Text
(a) CLOSS can often flip entilment to non-entailment by changing a word that appears in both the Question and Context.	**Old:** Question: When was Luther's last sermon? Context : His **last** sermon was delivered at Eisleben, his place of birth, on 15 February 1546, three days before his death. **New:** Question: When was Luther's new sermon? Context : His **last** sermon was delivered at Eisleben, his place of birth, on 15 February 1546, three days before his death. **Old:** Question: when was luther's last sermon? Context : his **last** sermon was delivered at eisleben, his place of birth, on 15 february 1546, three days before his death. **New:** Question: when was luther's traveling sermon? Context : his **last** sermon was delivered at eisleben, his place of birth, on 15 february 1546, three days before his death.
(b) CLOSS can sometimes induce entailment by changing a word in the Question (Context) to match one in the Context (Question).	**Old:** Question: Who were the ESPN Deportes commentators for Super Bowl 50? Context : On December 28, 2015, ESPN Deportes announced that they had reached an **agreement** with CBS and the NFL to be the exclusive Spanish-language broadcaster of the game, marking the third dedicated Spanish-language broadcast of the Super Bowl. **New:** Question: Who were the ESPN Deportes agreements for Super Bowl 50? Context : On December 28, 2015, ESPN Deportes announced that they had reached an **agreement** with CBS and the NFL to be the exclusive Spanish-language broadcaster of the game, marking the third dedicated Spanish-language broadcast of the Super Bowl.
(c) If lexical overlap fails, we often need many edits to change non-entailment to entailment.	**Old:** Question: Who was the number two draft pick for 2011? Context : This was the first Super Bowl to feature a quarterback on both teams who was the #1 pick in their draft classes. **New:** Question: Who was the show two draft pick for Kate? Context : This was the first Super half to feature a Premier on both teams who was the #1 pick in their draft classes.

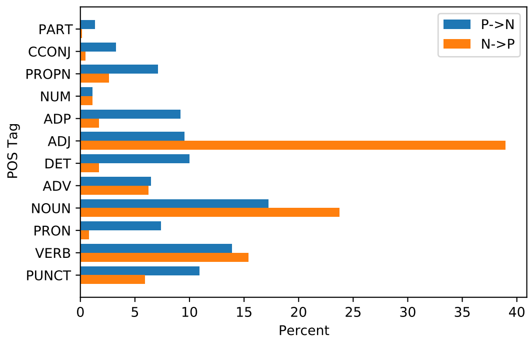
Table 6: Example QNLI counterfactuals generated by CLOSS. Each row demonstrates an interesting pattern of behavior we observe. We use green to highlight words whose changes flip the text to entailment and red for changes that flip texts to non-entailment.

Change	RoBERTa IMDB	BERT IMDB	RoBERTa QNLI	BERT QNLI
CLOSS	4.2	4.1	5.1	3.5
Increase beam width	0.9	0.8	2.6	3.3
Increase w	2.1	3.2	3.9	4.0
Increase K	**0.4**	**0.3**	1.4	2.4
Everything salient, fixed w	6.2	9.3	6.4	4.0
Everything salient, scale w	0.53	0.93	**0.53**	**0.23**

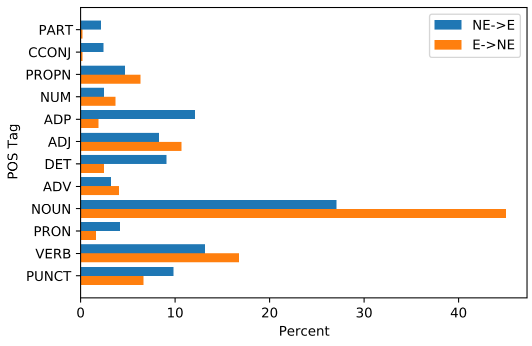
Table 7: Shows the impact on failure rate of significantly increasing CLOSS hyperparameters. Beam wdith increases from 15 to 100, w from 5 to 50, K from 30 to 300.



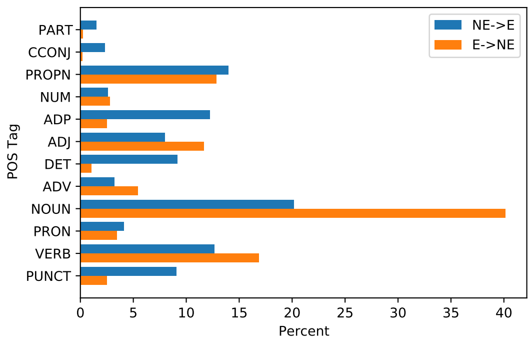
(a) BERT IMDB



(b) RoBERTa IMDB

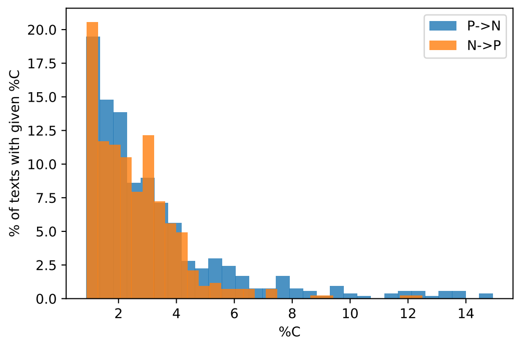


(c) BERT QNLI

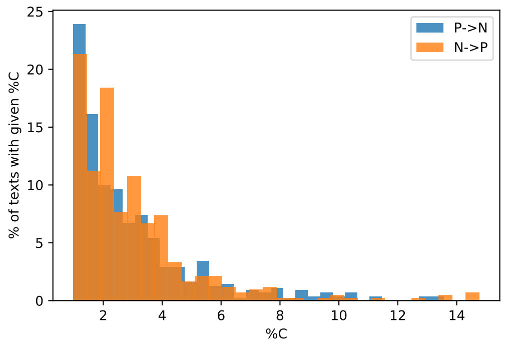


(d) RoBERTa QNLI

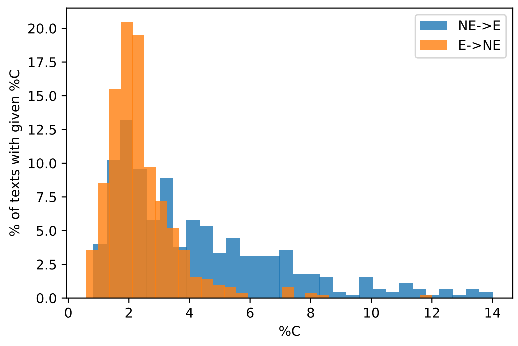
Figure 4: Barcharts showing how the changes CLOSS makes are distributed among each part of speech tag.



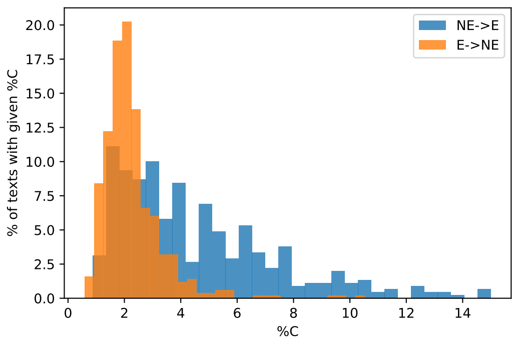
(a) BERT IMDB



(b) RoBERTa IMDB



(c) BERT QNLI



(d) RoBERTa QNLI

Figure 5: Histograms showing the distribution of percent of tokens changed over successfully generated counterfactuals.