Generating Realistic Natural Language Counterfactuals

Marcel Robeer
Netherlands Police Lab AI
Utrecht University
m.j.robeer@uu.nl

Floris Bex
Netherlands Police Lab AI
Utrecht University
Tilburg University
f.j.bex@uu.nl

Ad Feelders
Utrecht University
a.j.feelders@uu.nl

Abstract

Counterfactuals are a valuable means for understanding decisions made by ML systems. However, the counterfactuals generated by the methods currently available for natural language text are either unrealistic or introduce imperceptible changes. We propose *CounterfactualGAN*: a method that combines a conditional GAN and the embeddings of a pretrained BERT encoder to model-agnostically generate realistic natural language text counterfactuals for explaining regression and classification tasks. Experimental results show that our method produces perceptibly distinguishable counterfactuals, while outperforming four baseline methods on fidelity and human judgments of naturalness, across multiple datasets and multiple predictive models.

Original	+	I loved how the police helped me solve my case.
Unrealistic	-	how the police helped me solve my.
	-	the police me solve my case.
	-	I how the helped me to solve my case.
	-	I how the solve case.
Realistic	-	I **hated** how the police helped me solve my case.
	-	I **didn't like** how the police helped me solve my case.
	-	I loved how the **men** helped me solve my case.
	-	I loved how the police helped me solve **nothing**.

Figure 1: Example for a sentence predicted with positive (+) or negative (-) sentiment, with example (un)realistic counterfactuals and their sentiments.

1 Introduction

The increase of machine learning (ML) applications in high-stakes domains has led to a proliferation of Explainable AI (XAI) and Interpretable ML approaches, aimed at making models (global explanations) or individual decisions (local explanations) more understandable (Doshi-Velez and Kim, 2017; Tomsett et al., 2018). Output explanations explain individual decisions by understanding the (local) behavior around the output (Guidotti et al., 2019). However, in practice individuals may not always have access to the models they want explained (e.g. because of intellectual property) (Edwards and Veale, 2017). To overcome this access problem, model-agnostic approaches (sometimes called *post-hoc* approaches (Lipton, 2016)) only require access to the model outputs for provided instances, with the added benefit of being applicable to explain any model for a type of ML task (Ribeiro et al., 2016a). Prominent model-agnostic output explanations are local surrogate models (Ribeiro et al., 2016b), feature importances (Lundberg and Lee, 2017; Fong and Vedaldi, 2017), example-based explanations (Kim et al., 2016) and counterfactual explanations (Wachter et al., 2018).

Counterfactual explanations express *what might have happened instead* (Roese and Olson, 1995): certain values in an input instance are perturbed (e.g. the age of a defendant) while keeping other values the same, in order to observe how that influences the output (e.g. they would not have been convicted). Each of the output-changing perturbations is a counterfactual, where the difference between the counterfactual and the original instance provides insights into how the inputs affect the outputs, and can be used to pinpoint fairness issues and to reach a desired output (Wachter et al., 2018). As these counterfactuals are a valuable means to understand the behavior of a system, in recent years the same technique has been applied for explaining ML decisions—mainly for structured data (e.g. (Russell, 2019; Ustun et al., 2019; Wachter et al., 2018)) and image data (e.g. (Dhurandhar et al., 2018; Guidotti et al., 2019; Poyiadzi et al., 2020)).

Unlike structured and image data, counterfactuals for natural language text data have largely been disregarded. For text classification, Martens and Provost (2014) proposed the removal of words as a means to measure their contribution to the output. This paradigm was later adopted for constructing model-agnostic local surrogate models (Ribeiro et al., 2016a) and to determine which words have to be necessarily present for a classification decision (Ribeiro et al., 2018). Yet, this paradigm fails to create *realistic* counterfactuals (as illustrated in Figure 1 for sentiment analysis). Realisticness is an important property for how humans create and accept counterfactuals (Byrne, 2019; Miller, 2018) and may help prevent misleading explanations produced by model-agnostic explanation methods (Slack et al., 2020). Recently, human-in-the-loop approaches were proposed (Ribeiro et al., 2020; Wu et al., 2021) to support explainees in forming realistic counterfactuals.

In this work, we propose *CounterfactualGAN*: a method able to model-agnostically generate realistic, targeted counterfactuals for natural language text regression and classification without explainee intervention. Our method (i) generates realistic counterfactuals for the text-domain—ensuring dataset-specific realisticness by adversarially training on a training set—, (ii) uses a single model to provide counterfactuals for any instance and any (classification or regression) target of a black-box model, (iii) generates counterfactuals with a single pass after training, and (iv) does not require explainee intervention to do so.

2 Related Research

2.1 Counterfactuals for machine learning

In the literature, several properties of counterfactual generation methods for ML classification and/or regression models have been suggested. First, the generation of counterfactuals is either *targeted* (i.e. to a specific class or regression output) or *untargeted* (any target other than the original) (Zhang et al., 2020). Second, while generally viewed as being part of the post-hoc XAI approaches, some methods assume *white-box* access to the model rather than viewing it as a *black-box* (e.g. (Russell, 2019; Ustun et al., 2019)). For example, if the model is a linear classifier and its weights are accessible, these weights can be used to effectively find (targeted) counterfactuals. Third, for each original instance *one* or *multiple* counterfactuals can be found, providing either a single explanation or elucidating the various ways in which decisions may change (Wachter et al., 2018). When selecting multiple counterfactuals, approaches are typically concerned with the *diversity* of the counterfactuals to ensure maximal coverage with a sparse counterfactual set (Russell, 2019; Karimi et al., 2020).

While these properties impact the approach for obtaining counterfactuals, further strategies are required to confine the search space of possible counterfactuals to the ones that hold the best explanatory value. Some approaches select the *nearest* counterfactuals (e.g. (Wachter et al., 2018)), such that minimal changes are required to change to the counterfactual. However, more recently authors have addressed the issue of *implausibility*: "[…] the counterfactuals generated may not be valid data-points in the domain or they may suggest feature-changes that are difficult-to-impossible" (Keane and Smyth, 2020, pp. 166–167). Implausibility has been tackled with various strategies: either enforcing user-imposed *feasibility* constraints (e.g. excluding explanations where one needs to lower their age to lower the risk of recidivism) (Poyiadzi et al., 2020; Karimi et al., 2020) or using automated methods, such as selection based on closeness to a class prototype (Van Looveren and Klaise, 2021), that on the path from changing from the factual to counterfactual no other outputs are encountered (Laugel et al., 2019) or selecting instances from the training set as counterfactuals instead of generating them (Keane and Smyth, 2020).

2.2 Counterfactuals for text

Many counterfactual generation methods mainly consider structured and image data. Two methods that do support the creation of counterfactuals for natural language text require humans to determine where to apply changes in the instance: CheckList (Ribeiro et al., 2020) suggests input perturbations to a user, who can then choose from these perturbations and test how they affect the output, while PolyJuice (Wu et al., 2021) uses control codes in a finetuned GPT-2 model to form counterfactuals. MiCE (Ross et al., 2021) can generate counterfactuals using a finetuned T5 model with white-box access to a predictive model.

However, optimizing textual perturbations to find counterfactuals with black-box access in a fully automated manner poses specific problems, as encountered in the related areas of adversarial ML (seeking *semantically imperceptible* changes to text that change the black-box label) and style transfer (changing linguistic attributes of *ground-truth* texts, while retaining content). First, it is non-trivial how to define distance measures between the original instance and its perturbations, as they typically are discrete objects (Belinkov and Glass, 2019). Second, minimizing this distance cannot easily be formulated as an optimization problem, as this requires computing gradients on discrete inputs (Belinkov and Glass, 2019). The adversarial attack and style transfer literature tackles these problems by either (i) leveraging a combination of NLP algorithms and knowledge-engineered perturbation rules (e.g. (Li et al., 2019; Jin et al., 2020)), or (ii) finding perturbations in a latent space of an autoencoder model and decoding these into natural language instances (Melnyk et al., 2017; Wang et al., 2019).

We draw from insights in these areas, especially adversarial attacks—able to craft instances changing a black-box prediction—, but observe that their goal of *imperceptibility* of the perturbation (Zhang et al., 2020) (i.e. humans being unable to tell the difference between two instances and assigning the same label) is at odds with the goal of counterfactuals used in explanations to be human-understandable (Wachter et al., 2018). Counterfactuals provide explainees (e.g. developers, lay-users or examiners (Tomsett et al., 2018)) with meaningful information regarding a models' prediction, without necessitating technical know-how (Wachter et al., 2018). This requires an explainee to *perceive* the difference between the original instance and counterfactual (e.g. which words have changed), potentially even with limited technical or domain expertise. We note this perceptibility does not have to align with human judgments of distinguishing factors in a task: changing from 'him' to 'her' in sentiment analysis may be equally perceptible as changing from 'good' to 'bad'.

3 Realistic Counterfactuals

For natural language text, we propose a new strategy to create plausible counterfactuals in an automated manner, by applying a *realisticness constraint* to the generated counterfactuals. This approach has several benefits over the strategies tackling implausibility in Section 2.1: (i) no domain-specific assumptions have to be made by users, (ii) the perturbation path between an original instance and its counterfactual is not constrained and (iii) counterfactuals are not restricted to only training instances. What realisticness entails may depend on the type of language expected within a certain context, e.g. realistic movie reviews typically differ in use of language and grammatical correctness from Tweets. Therefore, we deem an instance realistic if it is indistinguishable from other instances (expected) in a dataset from a specific domain.

Definitions. Let us assume we are able to provide inputs to a black-box ML model $f\colon \mathcal{X} \to \mathcal{Y}$ and get the corresponding predictions, and have an instance $\mathbf{x} \in \mathcal{X}$ for which to find counterfactuals. $f(\cdot)$ was trained on a dataset containing N labeled or unlabeled instances (represented by $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{N}$ or $(\mathbf{X}, \mathbf{Y}) = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, respectively). Instance $\mathbf{x}$ is represented as a feature vector $\mathbf{x} = (x_1, x_2, \dots, x_n)$, where each x denotes a feature value. In our work, x_i is either a word-level token or is absent. A second index j ($x_{i,j}$) is used sometimes when tokens are represented by a probability distribution over a vocabulary of M tokens (including a special token indicating absence).

A perturbed instance $\widetilde{\mathbf{x}} = \mathbf{x} + \delta$ is formed by applying one or more valid perturbations δ to $\mathbf{x}$ (Dhurandhar et al., 2018). A valid perturbation transforms the feature values in $\mathbf{x}$ such that $\widetilde{\mathbf{x}} \in \mathcal{X}$. For example, valid perturbations are word replacements or removing a word by setting it to absent. We estimate instance realisticness by determining if it is indistinguishable from the original data distribution $p(\mathbf{x})$. In practice we estimate this indistinguishability with a discriminator model $g_{\mathbf{X}} : \mathcal{X} \to [0, 1]$ (trained on $\mathbf{X}$) indicating the likelihood that an instance $\widetilde{\mathbf{x}}$ could have come from $p(\mathbf{x})$. A low score indicates out-of-distribution instances.

We define the set of counterfactuals (CFs) of the instance $\mathbf{x}$ as $\mathrm{CF}_f(\mathbf{x}) = \{\widetilde{\mathbf{x}} \in \mathcal{X} \mid f(\mathbf{x}) \neq f(\widetilde{\mathbf{x}})\}$ (Karimi et al., 2020), i.e. all instances with an output different to $\mathbf{x}$. Targeted counterfactuals (TCFs) are instances that change the *fact* (the current output $y = f(\mathbf{x})$), to the *foil*, an output of interest $y' \in \mathcal{Y}$. Instead of requiring the output $f(\widetilde{\mathbf{x}})$ of perturbed instance $\widetilde{\mathbf{x}}$ to be exactly equal to y', we generalize the assumption $f(\widetilde{\mathbf{x}}) = y'$ and obtain:

$$\mathrm{TCF}_f(\mathbf{x},\ y') = \{\widetilde{\mathbf{x}} \in \mathcal{X} \mid d_f(f(\widetilde{\mathbf{x}}),\ y') \leq \epsilon\},$$

where $d_f : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ is a distance function indicating how similar the output $f(\mathbf{x})$ is to output of interest y' and $\epsilon \geq 0$ a user-defined threshold for the strictness in including instances.[1] A realistic (targeted) counterfactual is a (targeted) counterfactual that also maximizes realisticness model $g_{\mathbf{X}}(\cdot)$.

[1]For example, in regression analysis $d_f(y_1, y_2)$ may be defined as the squared error $(y_2 - y_1)^2$, while for a (multi-class) classifier a definition could be an indicator function $\mathbb{1}[y_1 \neq y_2]$ evaluating to 0 for target y_2.

4 Method: CounterfactualGAN

We operationalize model-agnostic, realistic, targeted counterfactuals for text regression and classification with our proposed method CounterfactualGAN. Our method works across predictive models by crafting counterfactuals in the form of a string. To ensure counterfactual realisticness, we combine insights from pretrained language models (LMs) and generative adversarial networks (GANs).

Generative Adversarial Networks (GANs). GANs use a generator G and discriminator D to learn latent representations or mappings in an unsupervised or semi-supervised manner (Creswell et al., 2018). By letting these networks compete, they are forced to jointly improve their performance. After training, generator G is used to generate realistic synthetic instances.

Rather than having G unconditionally generating realistic instances, Conditional GANs aim to create realistic mappings for specific inputs (Creswell et al., 2018). For example, an unconditional GAN may be able to create sentences with positive sentiment, while a Conditional GAN can turn the positive sentiment of an input sentence into a negative sentiment counterpart. First popularized in the image domain, approaches such as *pix2pix* (Isola et al., 2017) are able to e.g. convert grayscale photographs to full color and convert day scenes to night scenes. *CycleGAN* (Zhu et al., 2017) is able to do this without ground-truth pairs with example mappings between domains by jointly training two generators G_{ab} and G_{ba}—where G_{ab} learns a mapping from domain a (e.g. positive sentiment) to b (e.g. negative sentiment), while G_{ba} learns a mapping from domain b to a. An important contribution of CycleGAN is ensuring a minimal *reconstruction loss* of the network $G_{ba}(G_{ab}(\mathbf{x})) \approx \mathbf{x}$, helping the network to preserve relevant input features.

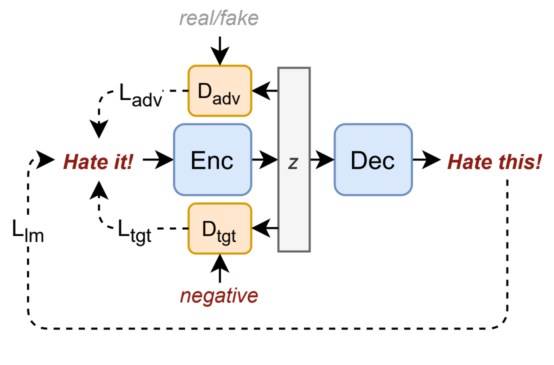
The downside of using CycleGAN is that in the multi-domain case (e.g. multi-class or a continuous domain) conditional generation requires training many separate generators and discriminators. *StarGAN* (Choi et al., 2018) mitigates this shortcoming of CycleGAN by using one generator G that takes both the input instance and a target domain (e.g. positive/negative sentiment) as inputs, and a single discriminator D that predicts (1) if the instance is real [D_{adv}] and (2) which target domain it belongs to [D_{tgt}].

Language models (LMs). While GANs have shown promising results, their application to natural language text has been limited. The discrete nature of text makes propagating the gradient from the discriminator back to the generator infeasible. We therefore opt to use the approach of finding a mapping in latent (embedding) space $\mathcal{Z}$ (in similar vein to e.g. (Melnyk et al., 2017)), but in our case use a pretrained LM for the autoencoder. Pretrained LMs have proven to greatly improve the state-of-the-art performance on a plethora of down-stream tasks (e.g semantic similarity, reading comprehension and commonsense reasoning) (Radford and Salimans, 2018). By using a *pretrained* LM, we leverage its adeptness in encoding syntax and semantic content—even beyond the training data.

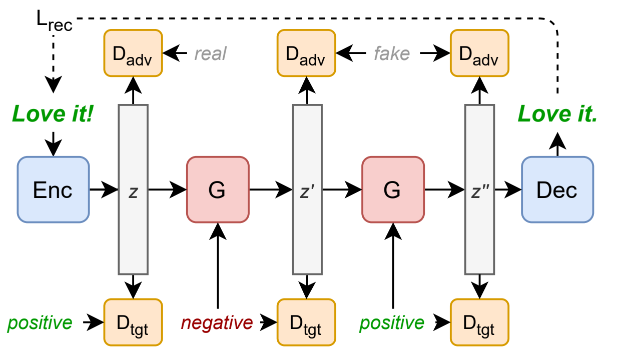
CounterfactualGAN. Our method combines the encoder-decoder architecture of an LM and the generator-discriminator architecture of a GAN for finding counterfactuals. Discriminator D is responsible for determining the realisticness of an instance $\widetilde{\mathbf{x}}$, while we use the predictions of black-box model $f(\cdot)$ to determine if a counterfactual is of output y'.

In practice, we use *BERT* (Devlin et al., 2019) as an LM encoder to create an embedding $\mathbf{z}$. Decoder $Dec(\cdot)$ is then tasked with mapping $\mathbf{z}$ back to original instance $\mathbf{x}$. For the GAN, we use a *StarGAN* (Choi et al., 2018), with a single generator G—that is provided with a target y'—, and one discriminator with two heads, tasked with determining whether the instance was real or fake (D_{adv}) and how well the instance corresponds to the target (D_{tgt}). To ensure that the output is similar after mapping to the target domain and back, the reconstruction loss is not only calculated on the embeddings $\mathbf{z}$ and $\mathbf{z}'' = G(G(\mathbf{z}, y'), y)$, but also on $\mathbf{x}$ and the token predictions according to the decoder $\mathbf{x}'' = Dec(\mathbf{z}'')$.

As our goal is to provide counterfactual explanations, unlike the original StarGAN, D_{tgt} is trained on the predicted labels $y = f(\mathbf{x})$ of the black-box decision function we aim to explain rather than ground-truth labels. Because black-box $f(\cdot)$ uses instances $\mathbf{x} \in \mathcal{X}$ to make its predictions rather than embedding $\mathbf{z} \in \mathcal{Z}$, D_{tgt} is first trained to distinguish target outcomes using embedding $\mathbf{z}$. In addition, as our method relies on a highly accurate mapping of the encoder-decoder part of the model, the encoder and decoder are pretrained on the training data as well. To incorporate these two requirements, we propose to train Counterfactual-



(a) *Phase 1*: finetuning the encoder-decoder language model on the dataset, while pretraining the discriminator.



(b) *Phase 2*: using generator G to adversarially find a targeted counterfactual mapping in embedding space $\mathcal{Z}$.

Figure 2: Two-phase training of CounterfactualGAN.

GAN in two phases, both illustrated in Figure 2.

Phase 1 (Figure 2a) starts with a pretrained LM encoder and decoder, and has the goal of (i) ensuring that the decoder accurately reconstructs embedding $\mathbf{z}$ into the encoded instance $\mathbf{x}$, and (ii) ensuring that D_{tgt} accurately mimics black-box $f(\cdot)$. Next, Phase 2 (Figure 2b) fixes the encoder and decoder weights, and introduces generator G to find the mapping in embedding space $\mathcal{Z}$. Generator G first maps $\mathbf{z}$ to $\mathbf{z}'$ with target y', and then back to original outcome y—resulting in $\mathbf{z}''$. For the discriminator, $\mathbf{z}$ is marked as a real instance of target y, $\mathbf{z}'$ as a fake instance of target y' and $\mathbf{z}''$ as a fake instance of original target y. CounterfactualGAN uses a three-layer Transformer decoder (Vaswani et al., 2017) for generator G, while discriminator D uses a two-layer GRU to combine embedding $\mathbf{z}$ into a single low-dimensional embedding for the entire input, used by the two heads D_{adv} and D_{tgt}. After both phases, a counterfactual is generated from $\mathbf{x}$ with target y' by running its encoding through generator G and decoding the generated embedding: $\widetilde{\mathbf{x}} = Dec(G(Enc(\mathbf{x}), y'))$. During this generation, a top-k of counterfactuals is generated for each instance, from which one string (i.e. the counterfactual) that is most similar to target y' according to $f(\cdot)$ is returned to the end-user.

Training objectives. To generate instances that are indistinguishable from real instances, discriminator D uses an adversarial loss

$$\mathcal{L}_{adv} = \mathbb{E}_{\mathbf{z}',r'}[(D_{adv}(\mathbf{z}') - r')^2],$$

where $\mathbf{z}'$ is a (generated) embedding and r' a value indicating whether the instance was real (1) or fake (0). G tries to minimize this objective, while D tries to maximize it. Next to the adversarial loss we also include a target loss $\mathcal{L}_{tgt}$ to ensure that the generated instances (embedded as $\mathbf{v} = \mathbf{z}'$) resemble the target domain, while the original instances (embedded as $\mathbf{v} = \mathbf{z}$) resemble the original domain (Choi et al., 2018). To handle multiple types of black-box methods, we further distinguish between classification and regression targets:

$$\mathcal{L}_{tgt} = \begin{cases} \mathbb{E}_{\mathbf{v},w}[(D_{tgt}(\mathbf{v}) - w)^2] & \text{for regression,} \\ \mathbb{E}_{\mathbf{v},w}[-\log D_{tgt}(w \mid \mathbf{v})] & \text{otherwise.} \end{cases}$$

Here, w is either the original label y (in case of $\mathbf{v} = \mathbf{z}$) or the target label y' (in case of $\mathbf{v} = \mathbf{z}'$) corresponding to that respective instance. We indicate the version used by the discriminator (with $\mathbf{z}$ and y) as $\mathcal{L}^D_{tgt}$, while we indicate the version used by the generator (with generated embedding $\mathbf{z}' = G(\mathbf{z}, y')$ and y' as target label) as $\mathcal{L}^G_{tgt}$. Both G and D try to minimize this objective.

Lastly, the reconstruction loss (Zhu et al., 2017) $\mathcal{L}_{rec} = \frac{1}{2}\mathcal{L}_{rec,x} + \frac{1}{2}\mathcal{L}_{rec,z}$ ensures that only domain-relevant parts of the inputs are changed when constructing a counterfactual. Here, we use a cross-entropy loss $\mathcal{L}_{rec,x}$ between original instance $\mathbf{x}$ and the cycle-reconstructed instance $\mathbf{x}'' = Dec(\mathbf{z}'')$ and the L_2-norm $\mathcal{L}_{rec,z}$ of their respective embeddings $\mathbf{z}$ and $\mathbf{z}'' = G(G(Enc(\mathbf{x}), y'), y)$:

$$\mathcal{L}_{rec,x} = \mathbb{E}_{\mathbf{x}}\left[\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{M} x_{i,j} \log x''_{i,j}\right],$$

$$\mathcal{L}_{rec,z} = \mathbb{E}_{\mathbf{x}}[||\mathbf{z} - \mathbf{z}''||^2],$$

where y is the original label, y' the target label, and x_i and x''_i (with the j-th token in a vocabulary) are corresponding elements of sequence $\mathbf{x}$ and $\mathbf{x}''$, respectively. G aims to minimize this objective.

In Phase 1 (finetuning) we train the encoder-decoder part with a language modelling loss $\mathcal{L}_{lm}$ and pretrain discriminator D by jointly training D_{adv} ($\mathcal{L}_{adv}$) and D_{tgt} ($\mathcal{L}^D_{tgt}$). To increase the informativeness of instances during finetuning, in some instances words are randomly swapped or commonly used tokens belonging to ground-truth instances with the approximate target values are inserted. The goal for Phase 1 is to minimize $\mathcal{L}_{finetune}$ using the aforementioned loss functions:

$$\mathcal{L}_D = \mathcal{L}_{adv} + \lambda^D_{tgt}\mathcal{L}^D_{tgt},$$
$$\mathcal{L}_{finetune} = \mathcal{L}_D + \lambda_{lm}\mathcal{L}_{lm},$$

where $\mathcal{L}_{adv}$ is the adversarial loss, $\mathcal{L}^D_{tgt}$ the discriminator target loss and $\mathcal{L}_{lm}$ the language modelling loss. λ^D_{tgt} and λ_{lm} are user-defined hyperparameters indicating the objectives' relative weights.

In Phase 2, the generator and discriminator are jointly optimized in an adversarial setting. They are trained using objective $\mathcal{L}_{GAN}$:

$$\mathcal{L}_G = \mathcal{L}_{adv} + \lambda^G_{tgt}\mathcal{L}^G_{tgt} + \lambda_{rec}\mathcal{L}_{rec},$$
$$\mathcal{L}_{GAN} = \mathcal{L}_G + \mathcal{L}_D,$$

where G tries to minimize objective $\mathcal{L}_{GAN}$ and D tries to maximize it. Generator loss $\mathcal{L}_G$ comprises the adversarial loss $\mathcal{L}_{adv}$, the target loss $\mathcal{L}^G_{tgt}$ for the generator, and reconstruction loss $\mathcal{L}_{rec}$ (responsible for ensuring minimal change when mapping to the target label and back). Again, λ^G_{tgt} and λ_{rec} are user-defined hyperparameters. The implementation details for our method are included in Appendix A.

5 Experiments

CounterfactualGAN was evaluated against four baselines using a quantitative validation and a human evaluation in the form of a user experiment.

5.1 Predictive models and datasets

We evaluated the generation methods using three task-specific datasets: one regression analysis task, one binary classification task and one multi-class classification task. To assess model-agnosticism, for each of these tasks three models were devised.

Datasets & tasks. We used three well-known NLP datasets for the training and evaluation of the predictive models and generation methods. These datasets cover various domains of NLP regression and classification tasks. HATESPEECH (Davidson et al., 2017) is a Twitter dataset used for hatespeech identification, where for our purposes the three class labels hatespeech (1.0), offensive language (.4) or neither (.0) were recoded to be used in a regression analysis of hatespeech severity. During preprocessing, @mentions in Tweets were anonymized with a string '`@user`'. The Stanford Sentiment Treebank (Socher et al., 2013) (SST-2) contains movie reviews with either positive or negative sentiment. SNLI (Bowman et al., 2015) is a textual entailment dataset, where the goal is to determine whether a hypothesis entails, contradicts or is neutral to a premise.

Each of these datasets was split into a training set (used for predictive/counterfactual generation model training), development set (used for hyperparameter optimization) and test set (used for evaluation). An overview of each dataset is provided in Table 1, including the task description, size of the dataset and its mean number of words.[2]

Predictive models. The predictive methods result in a model $f(\cdot)$, which gives predictive values (regression values or class probabilities) for each instance. First, we include a hand-crafted white-box model (`WB`) where ground-truth counterfactuals can be deduced.[3] In addition, we used two recent popular approaches that have shown competitive performance on several text regression and classification tasks as black-box models: InferSent (`IS`) (Conneau et al., 2017) and BERT (`BE`) (Devlin et al., 2019). Both models were finetuned on the specific dataset and corresponding task. The performance of each method on each tasks is shown in Table 1, where the performance is measured with *MSE* (lower is better) for HATESPEECH and macro-averaged F_1 (higher is better) for SST-2 and SNLI.

5.2 Counterfactual generation methods

We compared CounterfactualGAN to four baseline model-agnostic counterfactual generation methods. Each method creates a single counterfactual $\widetilde{\mathbf{x}} = \text{TCF}(\mathbf{x}, y')$ for each instance $\mathbf{x} \in \mathbf{X}_{\text{test}}$.

SEDC. *Search for Explanations for Document Classification* (Martens and Provost, 2014) aims to find the minimal set of words so that removing these words changes the decision from the current

[2]The reported mean length for SNLI is the total for premises (12.9) and hypotheses (7.4).

[3]The white-box for SST-2 (using a sentiment lexicon) was taken from the `AFINN` Python package (Nielsen, 2011) and for HATESPEECH (combining a lexicon, sentiment and text features such as readability) from Davidson et al. (2017). For SNLI, TF-IDF vectorized features were used by a logistic regression model to predict the labels.

		Instances				Performance			
Dataset	Task	Train	Dev.	Test	Mean length	Measure	`WB`	`IS`	`BE`
HATESPEECH	Regression analysis	17,391	2,429	4,363	19.1	$\downarrow$ *MSE*	.0877	.1265	.1223
SST-2	Classification (2 classes)	6,920	872	1,331	19.2	$\uparrow$ F_1-score	.6791	.8031	.8868
SNLI	Classification (3 classes)	550,152	10,000	9,997	20.3	$\uparrow$ F_1-score	.6662	.7658	.6248

Table 1: Dataset descriptives and corresponding predictive model performance.

output to a target output. SEDC iteratively deletes tokens in order to find a counterfactual.

PWWS+. *Probability Weight Word Saliency* (PWWS) (Ren et al., 2019) uses synonym substitutions from WordNet (Miller, 1995) to craft untargeted adversarial examples for classification. We extend this method for targeted counterfactual generation by (i) also allowing for antonym substitutions, (ii) including support for regression analysis and (iii) returning an instance close to the target.

Masked-LM. In Masked-LM, a baseline also used in Wu et al. (2021), words are deleted or replaced based on the most probable substitute at a `[MASK]` position according to a pretrained LM (`bert-base-uncased`). Variations of the original sentence are formed, in which up to five random words are replaced by a `[MASK]` token while retaining all other input words. Combinations of the top-2 predicted tokens (excluding the original) at each mask position are used as replacements.

TextFooler. TextFooler (Jin et al., 2020) provides a competitive baseline for semantic adversarial attacks for text classification and entailment. It replaces the most sensitive words with synonyms with an equal part-of-speech to craft adversarial instances, while ensuring maximal semantic similarity. For our purposes, we extend TextFooler to regression analysis by making two predictions y and y' approximately equal when $|y - y'| \leq 0.2$.

5.3 Evaluation

A realistic, targeted counterfactual generation method should produce counterfactuals that (i) accurately mimic black-box $f(\cdot)$, (ii) are realistic for a given dataset (see Section 3) and (iii) are perceptibly distinguishable from the original instance (see Section 2.2). To capture these aspects, we evaluated the generation methods on each predictive model and task using three metrics: *fidelity*, *naturalness* and *perceptibility*. Fidelity determines how accurately the method captures $f(\cdot)$, high naturalness indicates realisticness and perceptibility quantitatively estimates if the difference between $\mathbf{x}$ and $\widetilde{\mathbf{x}}$ is sufficient to be used for forming explanations. The quantitative metrics fidelity and perceptibility were evaluated on test set $\mathbf{X}_{\text{test}}$[4], while we selected a representative subset of 30 instances in $\mathbf{X}_{\text{test}}$ to evaluate naturalness in a human experiment. We assigned a random target y' to each instance $\mathbf{x} \in \mathbf{X}_{\text{test}}$, and used this to generate a corresponding targeted counterfactual $\widetilde{\mathbf{x}} = \text{TCF}_f(\mathbf{x}, y')$ using each generation method. This procedure was repeated for five random targets per instance, resulting in five counterfactuals for each instance. For CounterfactualGAN, the counterfactual for each instance was selected by generating the top-5 counterfactuals and selecting the one where $f(\widetilde{\mathbf{x}})$ was closest to target y'.

Table 2 shows example targeted counterfactuals. Additional examples are included in Appendix B.

Fidelity. Fidelity evaluates how well the generation method estimates the true behavior of the black-box predictive model (Ribeiro et al., 2016b). A generation method accurately mimicking its black-box will be able to produce counterfactuals that are of target class y' according to predictive model $f(\cdot)$. For classification, the fidelity is oftentimes captured using the label flip score (Wu et al., 2021), i.e. how often the predicted label 'flips' to the target label. To generalize this notion beyond classification, for each instance we compared output $f(\widetilde{\mathbf{x}})$ to target y' and measure its performance. The measures used are the same as for overall predictive model performance for each task. Table 3 reports on the results for the fidelity evaluation, showing that our method outperforms the baselines on 6 out of 9 model–dataset pairs. A one-way ANOVA shows a significant difference between methods for HATESPEECH [$F(4, 70) = 26.33$, $p < .01$], SST-2 [$F(4, 70) = 222.69$, $p < .01$] and SNLI [$F(4, 70) = 93.47$, $p < .01$]. A Tukey's post-hoc test ($\alpha = .05$) indicates that our method significantly outperforms Masked-LM and TextFooler on

[4]TextFooler on SNLI was tested on the first 1000 instances of $\mathbf{X}_{\text{test}}$, due to its long inference time (see Appendix A).

Method	Generated counterfactual	Prediction (WB)	Correct?
SEDC	~~**a**~~ blond ~~**woman**~~ in a black shirt ~~**is**~~ standing ~~**behind**~~ a counter ~~.~~	*entailment*	✗
PWWS+	a blond woman in a ~~**black**~~ **white** shirt ~~**is**~~ **differ** standing behind ~~**a counter**~~.	*contradiction*	✓
Masked-LM	a ~~**blond**~~ **young** woman in a ~~**black**~~ **blue** shirt is ~~**standing being a counter**~~ **seenly seated, in a large chair**.	*neutral*	✗
TextFooler	a ~~**blond**~~ **blonds** ~~**woman**~~ **lady** in a ~~**black**~~ **negra** ~~**shirt**~~ **jumper** is ~~**standing**~~ **stands** behind a ~~**counter**~~ **counteract**.	*contradiction*	✓
CounterfactualGAN	a ~~**blond**~~ **big** ~~**woman**~~ **man** in a black shirt is standing behind a counter.	*contradiction*	✓

Table 2: Targeted counterfactuals for '**Premise**: A blond woman in a black shirt is standing behind a counter. **Hypothesis**: The woman has her hair pulled back in a bun.' (SNLI), converting the prediction of white-box (WB) from *neutral* to *contradiction*. Methods that correctly flip the label are marked with a check mark (✓).

HATESPEECH (but not PWWS+ and SEDC), while outperforming all four baselines on SST-2 and SNLI. The improvement is most prominent for SST-2 (improving .104 to .255 percentage points over the best baselines), while also showing considerable improvement for the WB and IS models of the other datasets. Baseline methods showing poor fidelity (e.g. Masked-LM on SST-2 and SNLI) are unable to produce counterfactuals for most instances.

Perceptibility. To be used in explanations, the generation methods should produce counterfactuals $\tilde{\mathbf{x}}$ that are perceptibly distinguishable from their corresponding instances $\mathbf{x}$ (see Section 2.2). We quantitatively estimate perceptibility by taking semantic similarity estimated by the Universal Sentence Encoder (USE) (Cer et al., 2018), where we measure perceptibility with semantic distance $1 - \text{USE}(\mathbf{x}, \tilde{\mathbf{x}})$. Unlike semantic adversarial examples, which have the goal of minimizing this semantic distance, we aim to have a higher score such that the difference (e.g. positive words in a review to negative ones) can be easily perceived—while being far enough from a completely unrelated counterfactual (score of 1). Table 4 shows that our method has the highest perceptibility for 8 out of 9 model–dataset pairs. A one-way ANOVA shows a significant difference in perceptibility between methods for HATESPEECH [$F(4, 70) = 41.25, p < .01$], SST-2 [$F(4, 70) = 275.12, p < .01$] and SNLI [$F(4, 70) = 48.47, p < .01$]. A Tukey's post-hoc test indicates that our method scores significantly better than all baseline methods on perceptibility.

Human experiment: naturalness. We qualitatively determined which of the generation methods produces the most natural counterfactuals according to 196 native English speakers sampled from crowdsourcing platform *Prolific*[5]. Naturalness indicates how realistic an utterance is for a given context ('movie reviews', 'Tweets' or 'reading comprehension'). Participants were provided with pairs of counterfactuals generated for the same instance and predictive model, and asked which utterance was more natural in that context. A natural instance is one that could have been produced by a human (Novikova et al., 2017). Note that unlike other experiments humans were not asked to judge if the counterfactual correctly belongs to the target (e.g. positive/negative reviews), as the counterfactual explains the model behavior on the data—which may not correspond with human interpretation of the distinguishing factors.

Each participant received a random subset of 50 pairs in which they chose whether they prefer the first utterance (generated by one counterfactual generation method), the second (generated by another), or had no preference. The participants were urged to choose between the utterances even with a slight preference. All instances for each predictive model, dataset and generation method were shown at least five times to varying participants. Participants had excellent inter-rater reliability [Krippendorff's $\alpha = .84$]. Appendix C expands further on the experimental procedure. The results, reported in Table 5, show that our method is preferred (wins) regarding naturalness across all datasets and predictive models.

6 Conclusion

In this paper, we proposed CounterfactualGAN: a counterfactual generation method providing realistic counterfactuals to explain natural language text regression and classification black-box models, using a combination of pretrained LMs and a StarGAN to craft counterfactuals. Experimental results showed that our counterfactual generation method outperforms baselines across predictive

[5]https://prolific.co

	HATESPEECH (↓MSE)			SST-2 (↑F_1)			SNLI (↑F_1)		
Method	WB	IS	BE	WB	IS	BE	WB	IS	BE
SEDC	.166±.027	.116±.001	**.121±.002**	.629±.009	.677±.016	.647±.018	.407±.001	.398±.003	.477±.003
PWWS+	.168±.002	.124±.002	.129±.002	.694±.010	.697±.019	.634±.020	.400±.002	.410±.001	**.493±.002**
Masked-LM	.227±.002	.239±.002	.243±.001	.465±.010	.470±.016	.432±.024	.330±.004	.313±.001	.313±.002
TextFooler	**.132±.002**	.223±.002	.235±.002	.643±.014	.645±.012	.574±.019	.322±.015	.244±.006	.271±.008
Ours	.136±.002	**.097±.031**	.154±.044	**.798±.015**	**.890±.010**	**.902±.020**	**.487±.049**	**.534±.028**	.462±.008

Table 3: Mean (± standard deviation) fidelity of each counterfactual generation method, per dataset and predictive model (5 run average). The best score for each column is highlighted in **bold**.

	HATESPEECH			SST-2			SNLI		
Method	WB	IS	BE	WB	IS	BE	WB	IS	BE
SEDC	.20	.19	.19	.20	.20	.21	.10	.09	.09
PWWS+	.17	.18	.18	.18	.17	.18	.11	.12	.10
Masked-LM	.12	.12	.12	.12	.12	.12	.06	.08	.06
TextFooler	.21	.07	.07	.21	.26	.20	**.22**	.24	.27
Ours	**.32**	**.47**	**.21**	**.32**	**.37**	**.41**	.11	**.36**	**.37**

Table 4: Mean perceptibility of each counterfactual generation method (5 run average). The best score (↑) for each column is highlighted in **bold**.

Ours vs.		SEDC			PWWS+			Mask.-LM			TextFool.		
		W	T	L	W	T	L	W	T	L	W	T	L
Dataset	HATE.	**63**	11	26	**63**	15	22	**71**	9	20	**51**	24	26
	SST-2	**66**	2	32	**58**	8	35	**55**	10	36	**71**	13	16
	SNLI	**63**	7	30	**67**	10	23	**53**	21	27	**84**	9	6
Model	WB	**65**	5	29	**64**	5	31	**56**	13	31	**81**	12	8
	IS	**59**	6	35	**65**	11	23	**64**	13	24	**65**	16	20
	BE	**68**	8	25	**59**	16	25	**59**	13	28	**61**	19	20

Table 5: Percentage (%) of wins (W), ties (T) and losses (L) of our method against four baseline methods according to human judgments of naturalness. The methods deemed most natural are highlighted in **bold**.

models and datasets in finding human-perceptible, targeted counterfactuals, which remained natural according to human judgments.

CounterfactualGAN greatly improves natural language counterfactuals' quality, potentially having a profound effect on the explanation quality of model-agnostic XAI methods using perturbations to form explanations (e.g. local surrogates (Ribeiro et al., 2016b) and counterfactual explanations (Wachter et al., 2018)). For future work, we intend to (i) assess for which XAI methods and in which contexts realisticness is most beneficial, (ii) determine what level of perceptibility is optimal for human-understandable explanations, and (iii) extend CounterfactualGAN to other languages than English and more ML task types (e.g. multi-label classification).

Acknowledgements

This study has been partially supported by the Dutch National Police. The authors would like to thank Henry Prakken, Dong Nguyen and Arthur van Bunningen for their insightful discussions and valuable reviews of earlier versions of this work.

References

Yonatan Belinkov and James Glass. 2019. Analysis Methods in Neural Language Processing: A Survey. *Transactions of the Association for Computational Linguistics*, 7:49–72.

Samuel Bowman, Gabor Angeli, Christopher Potts, and Christopher Manning. 2015. A Large Annotated Corpus for Learning Natural Language Inference. In *Proceedings of the Empirical Methods in Natural Language Processing (EMNLP)*, pages 632–642.

Ruth M.J. Byrne. 2019. Counterfactuals in Explainable Artificial Intelligence (XAI): Evidence from Human Reasoning. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 6276–6282.

Daniel Cer, Yinfei Yang, Sheng yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Céspedes, Steve Yuan, Chris Tar, Yun Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal Sentence Encoder for English. In *Conference on Empirical Methods in Natural Language Processing (EMNLP): System Demonstrations*, pages 169–174.

Yunjey Choi, Minje Choi, Munyoung Kim, Jung Woo Ha, Sunghun Kim, and Jaegul Choo. 2018. StarGAN: Unified Generative Adversarial Networks for Multi-domain Image-to-Image Translation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8789–8797.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised Learning of Universal Sentence Representations From Natural Language Inference Data. In *Proceedings of the Empirical Methods in Natural Language Processing (EMNLP)*, pages 670–680.

Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A. Bharath. 2018. Generative Adversarial Networks: An Overview. *IEEE Signal Processing Magazine*, 35(1):53–65.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated Hate Speech Detection and the Problem of Offensive Language. In *Proceedings of the Conference on Web and Social Media (ICWSM)*, pages 512–515.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Amit Dhurandhar, Pin-Yu Chen, Ronny Luss, Chun-Chen Tu, Paishun Ting, Karthikeyan Shanmugam, and Payel Das. 2018. Explanations based on the Missing: Towards Contrastive Explanations with Pertinent Negatives. *Advances in Neural Information Processing Systems (NeurIPS 2018)*, pages 590–601.

Finale Doshi-Velez and Been Kim. 2017. Towards A Rigorous Science of Interpretable Machine Learning. *arXiv:1702.08608*.

Lilian Edwards and Michael Veale. 2017. Slave to the Algorithm? Why a Right to Explanation is Probably Not the Remedy You are Looking for. *Duke Law & Technology Review*, 16(1):18–84.

Ruth Fong and Andrea Vedaldi. 2017. Interpretable Explanations of Black Boxes by Meaningful Perturbation. In *Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV)*.

Riccardo Guidotti, Anna Monreale, Stan Matwin, and Dino Pedreschi. 2019. Black Box Explanation by Learning Image Exemplars in the Latent Feature Space. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECMLPKDD)*.

Ari Holtzman, Jan Buys, Leo Du, Maxwell Forbes, and Yejin Choi. 2020. The Curious Case of Neural Text Degeneration. In *International Conference on Learning Representations (ICLR 2020)*.

Phillip Isola, Jun Yan Zhu, Tinghui Zhou, and Alexei A. Efros. 2017. Image-to-Image Translation with Conditional Adversarial Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2017)*, pages 5967–5976.

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. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8018–8025.

Amir-Hossein Karimi, Gilles Barthe, Borja Balle, and Isabel Valera. 2020. Model-Agnostic Counterfactual Explanations for Consequential Decisions. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 895–905. PMLR.

Mark Keane and Barry Smyth. 2020. Good Counterfactuals and Where to Find Them: A Case-Based Technique for Generating Counterfactuals for Explainable AI (XAI). In *Case-Based Reasoning Research and Development*, pages 163–178. Springer International Publishing.

Been Kim, Rajiv Khanna, and Oluwasanmi Koyejo. 2016. Examples are not Enough, Learn to Criticize! Criticism for Interpretability. In *Advances in Neural Information Processing Systems (NIPS 2016)*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR 2015)*.

Thibault Laugel, Marie Jeanne Lesot, Christophe Marsala, Xavier Renard, and Marcin Detyniecki. 2019. The Dangers of Post-Hoc Interpretability: Unjustified Counterfactual Explanations. *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2801–2807.

Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. 2019. TextBugger: Generating Adversarial Text Against Real-world Applications. In *Network and Distributed Systems Security (NDSS) Symposium*, February.

Zachary C. Lipton. 2016. The Mythos of Model Interpretability. In *2016 ICML Workshop on Human Interpretability in Machine Learning (WHI 2016)*.

Scott M. Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. In *Advances in Neural Information Processing Systems (NIPS 2017)*, pages 4765–4774.

David Martens and Foster Provost. 2014. Explaining Data-Driven Document Classifications. *MIS Quarterly*, 38(1):73–99.

Igor Melnyk, Cicero Nogueira dos Santos, Kahini Wadhawan, Inkit Padhi, and Abhishek Kumar. 2017. Improved Neural Text Attribute Transfer with Non-Parallel Data. In *NIPS 2017 Workshop on Learning Disentangled Representations*.

George A. Miller. 1995. WordNet: A Lexical Database for English. *Communications of the ACM*, 38(11):39–41.

Tim Miller. 2018. Contrastive Explanation: A Structural-Model Approach. *arXiv:1811.03163*.

Finn Årup Nielsen. 2011. A New ANEW: Evaluation of a Word List for Sentiment Analysis in Microblogs. *CEUR Workshop Proceedings*, 718:93–98.

Jekaterina Novikova, Ondřej Dušek, Amanda Cercas Curry, and Verena Rieser. 2017. Why We Need New Evaluation Metrics for NLG. In *Proceedings of the Empirical Methods in Natural Language Processing (EMNLP)*, pages 2241–2252.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Rafael Poyiadzi, Kacper Sokol, Raul Santos-Rodriguez, Tijl De Bie, and Peter Flach. 2020. FACE: Feasible and Actionable Counterfactual Explanations. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society (AIES)*, AIES '20, page 344–350, New York, NY, USA. Association for Computing Machinery.

Alec Radford and Tim Salimans. 2018. Improving Language Understanding by Generative Pre-Training. *OpenAI*.

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.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016a. Model-Agnostic Interpretability of Machine Learning. In *2016 ICML Workshop on Human Interpretability in Machine Learning (WHI 2016)*, pages 91–95.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016b. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery in Data Mining (KDD'16)*, pages 1135–1144.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Anchors: High-Precision Model-Agnostic Explanations. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond Accuracy: Behavioral Testing of NLP Models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912, Online. Association for Computational Linguistics.

Neal J. Roese and James M. Olson. 1995. Counterfactual Thinking: A Critical Overview. In *What Might Have Been: The Social Psychology of Counterfactual Thinking*, pages 1–55. Lawrence Erlbaum Associates, Hillsdale, NJ.

Alexis Ross, Ana Marasović, and Matthew Peters. 2021. Explaining NLP Models via Minimal Contrastive Editing (MiCE). In *Findings of the Association for Computational Linguistics (ACL-IJCNLP 2021)*, pages 3840–3852, Online. Association for Computational Linguistics.

Chris Russell. 2019. Efficient Search for Diverse Coherent Explanations. In *Proceedings of the 2019 Conference on Fairness, Accountability, and Transparency (FAT* 2019)*, pages 20–28.

Dylan Slack, Sophie Hilgard, Emily Jia, Sameer Singh, and Himabindu Lakkaraju. 2020. Fooling LIME and SHAP: Adversarial Attacks on Post Hoc Explanation Methods. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society (AIES)*, pages 180–186.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher Manning, Andrew Ng, and Christopher Potts. 2013. Recursive Deep Models for Semantic Compositionality over a Sentiment Treebank. In *Proceedings of the Empirical Methods in Natural Language Processing (EMNLP)*, pages 1631–1642.

Richard Tomsett, Dave Braines, Dan Harborne, Alun Preece, and Supriyo Chakraborty. 2018. Interpretable to Whom? A Role-based Model for Analyzing Interpretable Machine Learning Systems. In *2018 ICML Workshop on Human Interpretability in Machine Learning (WHI 2018)*.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-Read Students Learn Better: On the Importance of Pre-training Compact Models. *arXiv:1908.08962v2*.

Berk Ustun, Alexander Spangher, and Yang Liu. 2019. Actionable Recourse in Linear Classification. In *Proceedings of the 2019 Conference on Fairness, Accountability, and Transparency (FAT* 2019)*, pages 10–19.

Arnaud Van Looveren and Janis Klaise. 2021. Interpretable Counterfactual Explanations Guided by Prototypes. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECMLPKDD)*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *Advances in Neural Information Processing Systems (NIPS 2017)*.

Sandra Wachter, Brent Mittelstadt, and Chris Russell. 2018. Counterfactual Explanations Without Opening the Black Box: Automated Decisions and the GDPR. *Harvard Journal of Law and Technology*, 31(2):841–887.

Ke Wang, Hang Hua, and Xiaojun Wan. 2019. Controllable Unsupervised Text Attribute Transfer via Editing Entangled Latent Representation. In *Advances* *in Neural Information Processing Systems (NeurIPS 2019)*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Tongshuang Wu, Marco Tulio Ribeiro, Jeffrey Heer, and Daniel Weld. 2021. PolyJuice: Automated, General-Purpose Counterfactual Generation. *arXiv:2101.00288*.

Wei Emma Zhang, Quan Z. Sheng, Ahoud Alhazmi, and Chenliang Li. 2020. Adversarial Attacks on Deep-Learning Models in Natural Language Processing: A Survey. *ACM Transactions on Intelligent Systems and Technology*, 11(3).

Jun Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. 2017. Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2242–2251.

A Implementation details

CounterfactualGAN is implemented in PyTorch (Python 3.7.5) and trained on a Tesla V100 GPU with CUDA 10.2. BERT is used as a pretrained language model (LM) encoder $Enc(\cdot)$, implemented using `prajjwal1/bert-small` of the Transformers package (Wolf et al., 2020)—allowing more effective training of the method parts than a larger model. `bert-small` is a small, uncased version of the BERT model from the official Google repository[6] with a hidden dimension size of 512, 4 attention heads and 4 layers that total to 29.1M parameters (Turc et al., 2019).

The encoder transforms the instance into an embedding of size $t \times h$, where t are the maximum number of tokens (where before the first token we place the special `[CLS]` token and `[SEP]` after the last token, and fill the remainder with `[PAD]` tokens) and h the hidden dimension size. Decoder $Dec(\cdot)$ is a fully-connected linear layer with bias, transforming an embedding $\mathbf{z}$ into a tensor $t \times v$ containing logits for each token in a vocabulary of size v. We extract k token sequences from these using nucleus sampling with $p = .9$ (Holtzman et al., 2020). Nucleus sampling selects the top logits for each t such that their softmax probabilities sum to p. The chosen tokens are then recombined into strings using the Penn Treebank detokenizer in `NLTK`[7] for each of the top-k counterfactuals. These were then fed back into $f(\cdot)$ to calculate their true target, after which the one most similar to the provided target y' was selected as the counterfactual.

During Phase 1, we increase instance informativeness by including copies of a batch where (i) in each instance 15% of the words are replaced by a `[MASK]` token, (ii) in each instance 15% of the tokens are randomly swapped and (iii) for 15% of the tokens random tokens belonging to the target (regression value bin or class) are inserted into the token sequence. In each case, this is done non-destructively to ensure that the special tokens `[CLS]` and `[SEP]` are not replaced.

Generator G is a three-layer Transformer decoder with three attention heads, which receives the embedding $Enc(\mathbf{x})$ as a shared input. The target tokens are the same embedding, except that the first token is replaced with an embedding of size h that contains the target for that instance. Discriminator D is a two-layer gated recurrent unit (GRU) with 10% dropout, which transforms an embedding $t \times h$ into an embedding $1 \times h$ that is used by the D_{tgt} and D_{adv} heads. Both D_{adv} (determining realisticness of instances) and D_{tgt} (predicting what the value of black-box $f(\cdot)$ for the embedding is) are single layer feed-forward neural networks.

Hyperparameters. All networks are optimized with the Adam optimizer (Kingma and Ba, 2014) with a learning rate of .0002, $\beta_1 = .9$ and $\beta_2 = .999$. During Phase 2 (GAN training), for each time G performs one step, D performs two. We use hyperparameter optimization to find the relative weights λ of parts of the loss functions, the number of epochs in Phase 1 (finetune epochs) e_1, and the number of epochs in Phase 2 e_2. Optimization was performed for 30 trials for each model–dataset pair using random combinations formed by Python package `optuna`.[8] The goal of this optimization was maximal fidelity (minimal MSE for HATESPEECH, maximal macro-averaged F_1 for SST-2 and SNLI). The search ranges for the hyperparameters were as follows: (i) e_1 (Phase 1 epochs) in range $[7, 10]$; (ii) e_2 (Phase 1 epochs) in range $[30, 100]$; (iii) λ_{lm} a choice of value from $(1, 3, 5, 10, 15, 25, 50, 100)$, and; (iv) $\lambda^D_{tgt,1}$ (during Phase 1), $\lambda^D_{tgt,2}$ (during Phase 2), λ^G_{tgt} and λ_{rec} a choice of value from $(1, 3, 5, 10, 15, 25, 50)$.

For reproducibility purposes, the hyperparameters of the model with the highest fidelity for each predictive model–dataset are reported in Table 6.

Dataset	Model	e_1	e_2	λ_{lm}	$\lambda^D_{tgt,1}$	$\lambda^D_{tgt,2}$	λ^G_{tgt}	λ_{rec}
HATES.	WB	7	90	1	50	1	15	1
	IS	10	83	5	10	25	50	10
	BE	7	76	10	50	10	15	25
SST-2	WB	7	88	10	25	25	25	10
	IS	7	38	1	25	25	3	1
	BE	9	53	15	50	3	1	3
SNLI	WB	9	63	1	15	5	25	5
	IS	8	83	5	50	3	50	1
	BE	10	49	10	25	50	25	15

Table 6: Hyperparameters for the highest fidelity runs of CounterfactualGAN.

Time usage. The training and inference time vary by dataset due to their different sizes (see Table 1). For the hardware setup that was previously reported, PyTorch on a single core of the Tesla V100 GPU, we report the mean wall time for train-

[6] https://github.com/google-research/bert

[7] https://www.nltk.org/

[8] https://optuna.org/

ing and the wall time for crafting counterfactuals for the whole test set (inference). Table 7 shows the results. Note that for the baseline methods SEDC, PWWS+, Masked-LM and TextFooler only inference is required and therefore only these times are reported. Since TextFooler on SNLI was only run for 1000 samples, where time taken for inference averaged 24.36 minutes, its value is estimated for all 9997 samples in the test set. These times show that while requiring training time to ensure counterfactual realisticness, our method is faster when generating counterfactuals for all datasets.

Generation method		HATESPEECH	SST-2	SNLI
SEDC		.92	.21	2.52
PWWS+		.98	.24	2.69
Masked-LM		5.37	.78	8.93
TextFooler		24.69	24.36	243.54*
Ours	*training*	31.01	17.03	134.55
	inference	.32	.14	1.01

Table 7: Mean wall times in minutes for inference (and training for CounterfactualGAN) per dataset.

Predictive models. InferSent (Conneau et al., 2017) (IS) uses a bi-directional LSTM on word-level GloVe embeddings (Pennington et al., 2014) to create semantic representations of sentences. In BE the [CLS] token in the final layer is used as a sentence representation. In both cases, these were then input into a linear layer to produce the prediction for the black-box predictive model.

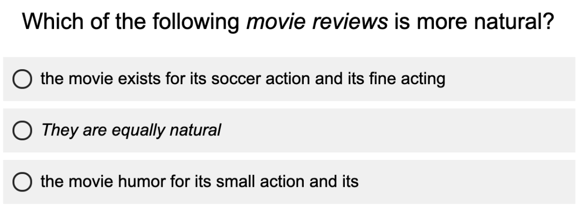
B Example Counterfactuals

Table 8 compares illustrative examples from all datasets for all counterfactual generation methods. In addition, we first include the original instance, and described its original prediction and the counterfactual target. Note that all instances and generated counterfactuals have been lowercased. Moreover, in HATESPEECH users in mentions are replaced by '@user' to ensure anonymity.

C Human Experiment

For our human experiment, we recruited 199 participants from crowd-sourcing platform Prolific. All users remained anonymous. Their Prolific ID was only used for participant pay-out (they were awarded £2.50 for 20 minutes of their time, providing a *good* hourly rate according to https://prolific.co in January 2021), after which it was discarded in further processing. Participants were randomly assigned to us, where the selection criteria we provided was that their first spoken language is English (self-reported) and had an approval rate of at least 80%. First, participants were introduced to the task to determine the naturalness of two utterances. We defined naturalness in our study as "[...] an utterance is more natural if it is more likely that it was produced by a human. Aspects you could consider are the type of language used in a context, grammatical correctness and semantically meaningful sentences." Next, they were asked to agree to a GDPR-compliant informed consent form before continuing their participation.

We generated pairwise comparisons by sampling 30 instances from each test set, and for each explanation method picking the corresponding counterfactuals from the run with the best fidelity score (highest F_1 or lowest MSE). Participants were provided with 50 pairwise comparisons, where for each question they were asked "Which of the following {context} is more natural?" The provided text in the {context} placeholder depended on the dataset these counterfactuals were generated for, namely *Tweets* for HATESPEECH, *movie reviews* for SST-2 and *reading comprehension sentences* for SNLI. Figure 3 provides an example question.


Which of the following *movie reviews* is more natural?

○ the movie exists for its soccer action and its fine acting

○ *They are equally natural*

○ the movie humor for its small action and its

Figure 3: Example naturalness preference question.

The 50 questions were randomly drawn from all pairwise comparisons, and shown in a random order. To check the quality of each submission, we included two quality control mechanisms: (i) we recorded the time of each survey completion and (ii) we included two control pairwise comparisons before and after the pairwise comparisons. The estimated completion time of the survey was 20 minutes, with a true average completion time of 14 minutes and 42 seconds. The options in the control questions compared true instances to ones with the lowest word-level edit distance by any generation method, one for HATESPEECH and one for SST-2. Excluding participants with completion times ≤ 5 minutes ($n = 2$) and participants choosing the non-natural answer for both control questions ($n = 1$), the final sample size was $n = 196$.

Dataset	Method	Original instance and generated counterfactuals
HATESPEECH (black-box IS, higher to lower hatespeech score)	*Original*	rt @user: i aint gonna text first cus pride b*tch
	SEDC	@user i aint cus
	PWWS+	rt user: i gonna text first cus humility
	Masked-LM	rt @user: i destroyedt gonna costumes & natural turner promising b*tch bree
	TextFooler	rt @user: i aint gonna text first cus pride bitch
	Ours (top-1)	rt @user: i aint na text first cus pride
	Ours (top-3)	rt @user: i aint gonna text first cus my pride
	Ours (top-5)	rt @user: i aint gonna text first cus my pride
HATESPEECH (black-box WB, medium to higher hatespeech score)	*Original*	rt @user: sonia never criticises kejriwal. kejriwal, who trashes every other leader, never criticises sonia. touching.
	SEDC	: sonia kejriwal, who .other leader criticises sonia. .
	PWWS+	rt @user: sonia ever criticises kejriwal ., who trashes every other leader, never criticises touching.
	Masked-LM	rt @user: load defend extensionss ke auditionsri ore .ld charged willy rod, honest trashes prevented liu
	TextFooler	rt @subscriptions: sonia never criticises kejriwal. kejriwal, who trashes each other executives, never criticises sonia. touching.
	Ours (top-1)	rt @user: sonia never criticises kejriwal,îÄ kejriwal, who trashes every other leader, never criticises sonia. touching.
	Ours (top-3)	rt @user: sonia criticises kej wall. kejriwal, who trashes every other leader, never criticises sonia. touching b*tch.
	Ours (top-5)	rt @user: sonia criticises kejriwal. kejriwal, who trashes every b*tch leader, never criticises sonia. b*tch touching b*tch.
SST-2 (black-box BE, from positive to negative)	*Original*	the movie has lots of dancing and fabulous music
	SEDC	movie lots and music
	PWWS+	the movie lack lots of fabulous music
	Masked-LM	the movie has 295 of dancing andial music
	TextFooler	the photo possesses parcel of cheer and amazing symphonic
	Ours (top-1)	the movie has loads of dancing and fabulous music
	Ours (top-3)	the movie has loads of dancing and terrible music
	Ours (top-5)	the movie has a loss of music and terrible music
SST-2 (black-box WB, from positive to negative)	*Original*	the problem with concept films is that if the concept is a poor one, there's no saving the movie
	SEDC	the with concept films is that if concept is a, there no saving the
	PWWS+	the problem with concept films is that if the concept is a rich one, there saving the movie
	Masked-LM	the problem with concept films is that if the concept is a $\geq$ one, there's no saving the movie
	TextFooler	the matters with concepts movie is that if the concepts is a poorer one, there's no save the film
	Ours (top-1)	the problem with concept films is that if the concept is a good one, there's no
	Ours (top-3)	the problem with films is that if the concept is a good one, there's no saving
	Ours (top-5)	the thing with concept films is that if the concept is a good one, there's saving
SNLI (black-box WB, from neutral to contradicting where edits are only applied to the premise)	*Original*	**Premise**: a man is posing on a ski board with snow in the background.
		Hypothesis: a naked man is posing on a ski board with snow in the background.
	SEDC	a man is on ski board snow in.
	PWWS+	a man is a ski board snow in the play up.
	Masked-LM	a man is posing on a ski board with snow in the background.
	TextFooler	a friend is parading on a slalom juries with blizzards in the wellspring.
	Ours (top-1)	a man is posing on a ski board with snow in the background.
	Ours (top-3)	a girl is sitting on a ski board with snow in the background.
	Ours (top-5)	a girl is posing on a ski with snow in the background.

Table 8: Illustrative examples comparing counterfactual generation methods.