CouRGe: Counterfactual Reviews Generator for Sentiment Analysis

Diego Carraro(✉) and Kenneth N. Brown

Insight Centre for Data Analytics, School of Computer Science and IT, University College Cork, Cork, Ireland
{diego.carraro,ken.brown}@insight-centre.org

Abstract. Past literature in Natural Language Processing (NLP) has demonstrated that counterfactual data points are useful, for example, for increasing model generalisation, enhancing model interpretability, and as a data augmentation approach. However, obtaining counterfactual examples often requires human annotation effort, which is an expensive and highly skilled process. For these reasons, solutions that resort to transformer-based language models have been recently proposed to generate counterfactuals automatically, but such solutions show limitations.

In this paper, we present CouRGe, a language model that, given a movie review (i.e. a seed review) and its sentiment label, generates a counterfactual review that is close (similar) to the seed review but of the opposite sentiment. CouRGe is trained by supervised fine-tuning of GPT-2 on a task-specific dataset of paired movie reviews, and its generation is prompt-based. The model does not require any modification to the network's architecture or the design of a specific new task for fine-tuning.

Experiments show that CouRGe's generation is effective at flipping the seed sentiment and produces counterfactuals reasonably close to the seed review. This proves once again the great flexibility of language models towards downstream tasks as hard as counterfactual reasoning and opens up the use of CouRGe's generated counterfactuals for the applications mentioned above.

Keywords: Natural language processing · Sentiment analysis · Language models · Counterfactual reasoning · Data augmentation

1 Introduction

Under the framework of example-based reasoning [20], *counterfactual examples* are widely-adopted as a proxy for investigating causality relationships between events [16]. Their usefulness is well-established in the machine learning literature as they have been employed in many settings and domains, for example, to boost model generalisation, provide explanations and to enrich datasets (e.g. [7,22,23] respectively). In Sect. 2 we briefly review different types of counterfactuals but, in this work, we focus on counterfactuals in the Natural Language Processing (NLP) domain - specifically in sentiment analysis.

L. Longo and R. O'Reilly (Eds.): AICS 2022, CCIS 1662, pp. 305–317, 2023.
https://doi.org/10.1007/978-3-031-26438-2_24

As a demonstrating and relevant example, consider the four textual movie reviews in Table 1. Literature has proposed approaches to generate counterfactual reviews of types **a**, **b** and **c** from the seed review **s**. Review **a** is a *task-specific* counterfactual because its generation is targeted to apply a specific different *counterfactual label* to the review, i.e. the negative sentiment. Generations of this kind can be found in [8,14], for example. Instead, review **c** is a *general-purpose* counterfactual because its generation isn't tailored to any downstream task, i.e. the sentiment label does not necessarily change[1]. Generations of this kind can be found in [17,27], for example.

Table 1. Example of a seed review **s** with three corresponding counterfactual reviews (**a**, **b**, **c**) where edits are highlighted in blue.

Id	Review	Sentiment	Generation type
s	"Titanic" is a good movie because of the original plot and the fascinating cast.	Positive	–
a	"Titanic" is a bad movie because of the expected plot and the low-performing cast.	Negative	Task-specific
b	"Titanic" is a bad movie because of the expected plot (really, I could predict every single minute of it, not kidding) and the horrible soundtracks.	Negative	Task-specific
c	"Titanic" is a good movie because of the original cast and the fascinating plot	Positive	General-purpose

A counterfactual review should be close to the seed review so that minimal changes allow causality assessments [16]. For example, while review **a** and **b** lead to the same negative sentiment, the former is much closer to **s** than the latter. In this paper, we focus on counterfactual reviews of type **a**, i.e. close to **s** but of different sentiment.

Also, generation can be manual or automatic (or hybrid [27]). When manual, human annotators are required to edit the seed review manually to generate counterfactuals. The editing process is generally accurate but expensive: human annotators are required to be "experts" in the task, and the effort dedicated to each generation can be quite high (e.g. 4–5 min in average [8]). Also, resorting to the manual approach might be a limitation in applications where online single-generation is required rather than offline batch-generation. On the other hand, automatic generation is generally cheaper and is fast enough to be suitable for interactive use, thus being appropriate for many modern data-hungry settings.

Although automatic generation is a way of obtaining a large number of cheap counterfactuals, we believe the approach is still under-investigated in the NLP domain. The most successful applications leverage recent progress on transformer-based [24] language models (LMs). By modification to the model's

[1] When the generation is task-specific but the counterfactual label and the seed label are the same, the generated instance is known as semi-factual, e.g. the counterfactual explanations in [9].

architecture and/or fine-tuning, some works apply a controlled generation to a specific task, e.g. [14,17] and some others to a specific part of the text, e.g. [21,27]. Our solution to automatic counterfactuals generation is inspired in particular by [1,17,27] and targets the sentiment analysis task. Indeed we design a generator, which we name CouRGe, that, given a textual seed review and a counterfactual sentiment, produces a textual counterfactual review close to the seed review and displaying the target sentiment. We implement CouRGe by fine-tuning GPT2 [18] with a task-specific dataset of paired examples, and we leverage a prompt-based generation framework [12]. We run experiments[2] on a movie review dataset where we investigate different training scenarios for CouRGe. Results show that CouRGe can generate counterfactuals that belong to the target sentiment and that are diverse and fairly close to the seed review.

The remainder of the paper is structured as follows: Sect. 2 reviews related work in the literature; Sect. 3 outlines the counterfactuals generation framework we employ and describes how we train CouRGe; Sect. 4 presents the experiments and analyse results; and Sect. 5 draws conclusions and illustrates future plans.

2 Background and Related Work

2.1 Counterfactual Examples: Applications

Counterfactual examples have been used for a variety of goals: to explain the outputs of a model for increasing interpretability and trust for both users and AI practitioners in (e.g. [6,7,25]); to obtain more robust models that (hopefully) capture not only spurious correlation relationships, but also causal relationships between inputs and outputs of a model (e.g. [23,26]); to increase fairness (e.g. [5,10]); or simply for data augmentation purposes (e.g. [13,28]).

Counterfactual and adversarial examples are related but different in nature [3]. Indeed, adversarial examples (also known as adversarial attacks) are test inputs created with the purpose of *fooling* a model to misclassify such inputs. They are designed with the specific goal of testing the robustness of a model to unexpected and out-of-distribution inputs. Also, counterfactuals are used to test a model in some settings (e.g. [4,14]), but their use is more related to the interpretability and the analysis of the causal effects between the inputs and the outputs of the model [3]. Although generation algorithms in the literature work with similar principles for both counterfactuals and adversarials, the former typically hold additional properties such as *plausibility* (i.e. generated examples are realistic and in-distribution) and *human-perceptibility* (i.e. changes on the generated examples need to be perceptible by a human evaluator) [14,28].

2.2 On Generating Counterfactuals for NLP

In the NLP domain, manual approaches to generate counterfactuals have been proposed, for example, in [4,8,17]. Similarly, the authors employ human crowd

[2] The code is available at https://github.com/cdiego89phd/counterfactuals-generation.

workers to generate counterfactual reviews from original textual movie reviews. This editing process instructs workers to apply minimal perturbations to the seed text (i.e. closeness constraint) but at the same time ensure that the generated text remains coherent and fluent (i.e. coherence-fluency constraint) and that the counterfactual label applies (i.e. label-flip constraint, when applicable). Generations of this kind are generally very expensive and often impractical: for this reason, in this paper we propose a cheaper alternative, i.e. automatic generation. In the remainder of this section, we review literature that is closest to and inspired our work.

PPLM [1] and GYC [14] are LM-based tools able to generate text entailed to one or more controllable attributes, such as class labels, for example. In practice, the generation is controlled by specific attribute models that are plugged in on top of the LM so that the generation does not require any further training of the LM. While GYC is designed to produce counterfactuals from a seed text, PPLM is a general-purpose text generator. MiCE is a tool that resorts to a two-stage process to generate counterfactuals as a proxy for interpretability [21]. In the first step, MiCE identifies portions of the seed text that are associated with the example's label; in the second step, such portions are minimally perturbed to obtain a text matching a specific counterfactual label. POLYJUICE [27] is a general-purpose conditional counterfactual generator for text sentences. It is a GPT-2 version fine-tuned on various paired-sentences datasets that allow for control over perturbation types and locations through pre-defined control codes. Finally, Counterfactual Story Rewriting (CSR) is a system able to perform counterfactual narrative reasoning and revision by fine-tuning an LM with a task-specific dataset [17].

CouRGe is inspired by PPLM, GYC and MiCE because generation is controlled towards a specific label; it is close to CSR because the training is performed with a task-specific dataset (and we propose a different training scenario); and it uses prompting, which resembles the use of control codes in POLYJUICE.

3 Training CouRGe

3.1 Framework

Our goal is to build a generator G with parameters θ, i.e. G_θ, able to perform the following task: *given a seed review with its sentiment label and a counterfactual target sentiment, generate a counterfactual review as close as possible to the seed review and of target sentiment.* More formally, given a seed review x of sentiment s and a counterfactual opposite sentiment $\overline{s}$, we require G_θ to learn the function g_θ, that returns the counterfactual review $\hat{x}$, as close as possible[3] to x and of sentiment $\overline{s}$:

$$g_\theta(x, s, \overline{s}) = \hat{x} \tag{1}$$

where a sentiment is either positive ($s, \overline{s} = 1$) or negative ($s, \overline{s} = 0$).

[3] We use different distance metrics to measure the closeness, see Sect. 4.

3.2 Training Scenarios

In this section, we describe different training scenarios for our task. We use two variants of the GPT-2 pre-trained language model [18] as base models, i.e. GPT2 and GPT2-m (124 and 355 million parameters respectively), leading to 12 different trained model versions. However, such training scenarios are general, and other pre-trained models could be used with little modification (e.g. the BERT family [2], the T5 family [19]). In some training scenarios below, we also assume the availability of a dataset of n paired reviews $\mathcal{D} = \{x_i, s_i, \overline{x}_i, \overline{s}_i\}$ where x is a seed review with sentiment s_i and a ground truth counterfactual review $\overline{x}_i$ with sentiment $\overline{s}_i$ (we will use the counterfactually-augmented dataset from [8]).

Zero-Shot (ZS). There is no training in this scenario, i.e. we employ GPT2 and GPT2-m to assess the generation capabilities that these models gained from the pre-training.

Unsupervised Fine-Tuning (UFT). In this scenario, we expose GPT2 and GPT2-m to a movie-specific corpus to drive the models' text generation toward the target domain and vocabulary (sometimes, this type of training is also known as continual pre-training). In this setting, the model is fine-tuned to maximize the log-likelihood of the reviews in the corpus C:

$$\mathcal{L}^{UFT}(\Theta) = \log g_\theta(C) \tag{2}$$

Supervised Fine-Tuning (SFT). We use the task-specific dataset from [8] (and formally described in Sect. 3.1) to fine-tune GPT2 and GPT2-m so that the text generation will be specific to our task. Informally, this setting is equivalent to a supervised scenario where ground-truth counterfactual reviews are the target labels. We perform prompt-based fine-tuning [12], where we design two specific manual prompts. The log-likelihood is the following:

$$\mathcal{L}^{SFT}(\Theta) = \log g_\theta(f_{pt}(x, s, \overline{x}, \overline{s})) \tag{3}$$

and f_{pt} is a function that encapsulates the input into the prompt (Table 2).

Unsupervised and Supervised Fine-Tuning (UFT + SFT). In this scenario, we sequentially combine UFT first (Eq. 2) and SFT afterwards (Eq. 3), in order to leverage the advantages of both training steps.

3.3 Generation Step

At generation time, we feed the models from scenarios ZS and UFT with s, x, $\overline{s}$ (separated by the special separation token [SEP]) and we ask them to generate $\overline{x}$. For scenarios SFT and (UFT + SFT) we apply prompt-base inference so that we query the models with the encapsulated input $f_{pt}(x, s, \overline{s})$ to generate $\overline{x}$.

Table 2. The close prompts used for training and generation. The design of P1 and P2 is inspired by [12]. To note, we fill the sentiments s and $\overline{s}$ with the strings accordingly to the sentiment map reported. Also, we use special tokens in square brackets for the prompts: [SEP] is a separator; [BOS] and [EOS] indicate the beginning and the end of the generation, respectively.

Id	Prompt (f_{pt})	Sentiment map	
		$s, \overline{s} = 1$	$s, \overline{s} = 0$
P1	"[BOS]s review:[SEP]x[SEP] $\overline{s}$ review:[SEP]$\overline{x}$[EOS]"	"Positive"	"Negative"
P2	"[BOS]The movie is s.[SEP]x[SEP] The movie is $\overline{s}$.[SEP]$\overline{x}$[EOS]"	"good"	"bad"

4 Experiments

4.1 Datasets Preprocessing

Because our target domain is the movie domain, for the UFT setting, we use the Rotten Tomatoes movies and critic reviews dataset[4]. We randomly split the dataset into training and validation sets (with 80%-20% ratio).

CAD-IMDb[5] is the movie reviews dataset we employ for the SFT scenario. The dataset accounts for 2440 examples: each example is a pair of reviews where one review is the seed review x and the other is the counterfactual review $\overline{x}$[6]. We randomly split the dataset into training, validation and test sets (with 70%-12%-18% ratio).

4.2 Experimental Methodology

When training the different versions of CouRGe in the various scenarios, we use the validation set to tune the hyperparameters (we optimise for the perplexity metric [18] with early stopping); we consider the tuning of the learning rate, weight decay, adam epsilon, warmup steps and accumulation steps.

After a model is trained, i.e. at test time, we run the generation step (Sect. 3.3) three times, so that the model generates three counterfactuals for each seed review in the test set. Similarly, we perform the generation step for the baseline models (see details in the next section) and obtain three counterfactuals per seed review in the test set. For the baselines and our CouRGe models, we randomize the generation so that, instead of selecting the next token with

[4] https://www.kaggle.com/datasets/stefanoleone992/rotten-tomatoes-movies-and-critic-reviews-dataset.

[5] https://github.com/acmi-lab/counterfactually-augmented-data/tree/master/sentiment.

[6] It is not clear which of the two reviews is the original and which one is the manually-crafted counterfactual: we randomly assign one review to be the seed review and the other to be the counterfactual review.

the highest probability, we select among multiple tokens with the highest probability. After the generation is completed, we assess the performances of each generator, computing the metrics described in Sect. 4.4.

Tuning of Generation's Hyperparameters. At the generation step, LMs can control the generation by setting hyperparameters such as the number of beams, repetition penalty, n-gram repetitions, top-k and top-p. To assess the impact of such hyperparameters, we run further experiments (denoted by SFT*) where we take the models from the SFT scenario and we tune hyperparameters on the validation set before running the generation (and we optimize for BLEU, see Sect. 4.4).

Out-Of-Domain (OOD) Test. To assess the generalisation capabilities of our generator, we evaluate CouRGe on two additional test sets, i.e. movies' reviews[7] from the IMDb website and businesses' reviews[8] from the Yelp website.

4.3 Baselines

Among the generators presented in Sect. 2, we selected two baseline generators to compare the performances of our CouRGe. We resort to the trained models made available in their repositories and do not perform any hyperparameter tuning (we use the default values).

PPLM [1]**:** for each seed review in the test set, PPLM uses a context, a Bag of Words (BoW) and a sentiment discriminator to generate a counterfactual. The context is the first three words of the seed review (similarly to [14]); the BoW is composed of the words in the seed review; and the discriminator guides the generation towards the counterfactual label.

POLYJUICE [27]**:** we run the generator on the full-automatic setting. Thus, for each seed review in the test set, we randomly select k sentences to perturb. Each of the selected sentences is entirely blanked (which means that we randomly select the perturbation type), leaving the rest of the seed review as it is. To note, POLYJUICE has been trained with the same task-specific dataset presented in Sect. 4.1 (including the test set portion), which is a considerable advantage over PPLM and our CouRGe.

We do not employ GYC [14] and MiCE [21] as baselines for our experiments. Regarding the former, there is no open implementation available, and its approach is similar to PPLM. We omit the latter because its generation process would unfairly favour the performances on the LFS metric (see next section).

[7] The polarity dataset v2 at www.cs.cornell.edu/people/pabo/movie-review-data.
[8] https://huggingface.co/datasets/yelp/_polarity.

4.4 Evaluation Metrics

We evaluate each generator by applying a wide range of automatic metrics that measure the generated counterfactuals' effectiveness, closeness and diversity. For each metric below, we first average the metric scores across the three generated counterfactuals and then across all the test instances.

Effectiveness. Ensure that the counterfactual label applies to the generated text. We choose to employ the Label-Flip Score (**LFS**), which scores 1 when the counterfactual sentiment is the opposite of the seed sentiment. To predict each label, we use a version of DistilBERT, a sentiment classifier fine-tuned on the SST-2 sentiment dataset[9] (selected as the most accurate classifier among different candidates through a small experiment run on the CAD-IMDb of [8]).

Closeness. We measure Levenshtein edit distance (**LEV**) [11] and the syntactic closeness with the tree-edit distance (**TED**) [29], and we do that by comparing each counterfactual with its corresponding seed review. Also, we compute corpus-level **BLEU** from Papileni et al. [15], widely-used to measure the performance of translation machines, which calculates the overlap between the generated counterfactuals and their respective reference counterfactuals in the test set.

Diversity. We use the Self-BLEU (**S-BLEU**) proposed by Zhu at al. [30]. For each seed review, we compute the metric between the three corresponding counterfactuals (the lower the metric's value, the better).

4.5 Results

The first set of results is reported in Table 3. POLYJUICE's counterfactuals (when $k = 2$) are close to their seed review (best performance for LEV and BLEU) and diverse, but they are not effective (worst performance for LFS). This is as expected, considering the nature of the generator. Indeed, because POLYJUICE's counterfactual reasoning is applied at a sentence level, then closeness is ensured (perturbations are minimal); at the same time, there is no such reasoning at an inter-sentence level, which makes the label flip difficult to achieve for multi-sentences reviews. For $k \in \{3, 4\}$ we have similar outcomes. When $k = 1$, closeness metrics improve (e.g. LEV= 0.09, TED= 10.1) but LFS drops to 0.19. (Results for $k \in \{1, 3, 4\}$ are not reported due to space constraints.)

PPLM's performances are surprisingly low: despite PPLM being able to control the sentiment and the content of the generated text, it fails to generate good counterfactuals accordingly to all the metrics (except for diversity). A possible explanation is that we do not tune the extensive range of the model's hyperparameters. We leave this task for future work.

[9] https://huggingface.co/distilbert-base-uncased-finetuned-sst-2-english.

Table 3. Results of the evaluation, where the test set is composed by 488 instances. We do not report performances for the ZS scenario, as they are very similar to the ones in UFT. For POLYJUICE, we report results for $k = 2$, being the version with the highest LFS. In bold, we highlight the best-performing value of each metric.

Model	Training scenario	Prompt	LFS ↑	LEV ↓	TED ↓	BLEU ↑	S-BLEU ↓
POLYJUICE-2	–	–	0.27	**0.18**	17.3	**0.71**	0.84
PPLM	–	–	0.44	1	59.2	0.01	**0.07**
CouRGe-GPT2	UFT	–	0.54	1	70.1	<0.01	0.28
CouRGe-GPT2-m	UFT	–	0.53	1	68.8	<0.01	0.26
CouRGe-GPT2	SFT	P1	0.88	0.3	23.6	0.45	0.84
CouRGe-GPT2	SFT	P2	0.88	0.3	25.2	0.44	0.83
CouRGe-GPT2-m	SFT	P1	**0.89**	0.32	23.5	0.43	0.83
CouRGe-GPT2-m	SFT	P2	0.87	0.31	23.2	0.43	0.82
CouRGe-GPT2	UFT+SFT	P1	0.85	0.3	26.7	0.45	0.84
CouRGe-GPT2	UFT+SFT	P2	0.85	0.35	28.3	0.39	0.79
CouRGe-GPT2-m	UFT+SFT	P1	0.88	0.32	25.2	0.43	0.84
CouRGe-GPT2-m	UFT+SFT	P2	0.85	0.35	28.3	0.4	0.88
CouRGe-GPT2	SFT*	P1	0.84	0.2	**15.8**	0.57	0.89
CouRGe-GPT2	SFT*	P2	0.85	0.22	17.3	0.54	0.88
CouRGe-GPT2-m	SFT*	P1	0.87	0.23	16.5	0.54	0.89
CouRGe-GPT2-m	SFT*	P2	0.87	0.23	16.4	0.55	0.85

Results for the training scenarios ZS and UFT of CouRGe (we only report the latter as they are similar to the former) show that counterfactual reasoning is a challenging task that cannot be successfully addressed without proper fine-tuning. In particular, performances are poor accordingly to all metrics, even when the LM is shifted towards the domain-specific distribution (UFT scenario).

For the SFT scenario, CouRGe produces effective and reasonably close counterfactuals (best value for LFS while BLEU is the metric where performance is not outstanding). Disproving what is found in [17], models trained in the (UFT+SFT) do not benefit from the UFT training, as results are very similar to the ones in SFT. As expected, when we optimize for closeness, performances improve for LEV, TED and BLEU, while LFS suffers a small drop. Also, diversity is relatively poor in all scenarios (and it is comparable to POLYJUICE's diversity). As a final remark on Table 3, CouRGe built on GPT2-m does not perform better than the one built on GPT2 and training with the two different prompts also leads to similar performances, contrary to what is found in [17].

We also found that CouRGe can generalise fairly well on unseen and out-of-domain data, see Table 4. This is true in particular for the out-of-domain Yelp test, where performances are comparable to the ones reported in Table 3. For the IMDb test, performance degrades despite the fact that reviews are in the same movie domain used for training CouRGe. A possible cause for this is the average length of the seed review given as input to the generator, which is significantly higher than the one in Yelp or in the training set (i.e. 901 characters).

Table 4. Results of the ODD evaluation, where each test set is composed of 250 instances. We employ the best performing model in terms of LFS, i.e. CouRGe-GPT2-m from SFT. We do not measure BLEU as reference counterfactuals are not available in the datasets.

ODD test	Avg. seed review length	LFS	LEV	TED	S-BLEU
IMDb	3892 chars	0.66	0.84	71.2	0.43
Yelp	723 chars	0.81	0.35	21.0	0.77

Table 5. Average computational time for each model's generation. Experiments were run on a NVIDIA A40 48 GB GPU.

Model	Generation time
POLYJUICE-2	2 s per seed review
PPLM	164 s per seed review
CouRGe-GPT2	11 s per seed review
CouRGe-GPT2-m	13.77 s per seed review

Also, Table 5 reports the average times spent by the models for generating the three counterfactuals from the seed review: PPLM takes the largest amount of time and therefore, its generation can only fit batch/offline settings. Instead, the other three might be suitable for both online and offline settings (in particular, POLYJUICE stands out with 2 s per review).

5 Conclusion and Future Work

In this paper, we have designed and trained CouRGe, a GPT2-based text generator able to generate counterfactual reviews for the sentiment analysis task. We have proven that GPT2 is an excellent learner because it can be fine-tuned to perform counterfactual reasoning with no modifications to the training procedure or the model's architecture. Based on our experiments that compare CouRGe with PPLM and POLYJUICE (two state-of-the-art generators), our model is much more effective (i.e. the counterfactual label applies more often), while closeness and diversity are comparable or better than the ones shown by POLYJUICE (the best baseline for these metrics). One limitation of CouRGe is the computational expense in terms of time. Indeed, despite being an order of magnitude faster than PPLM on average for a single instance generation, our model might not be suited to operate in some online settings but only in offline settings. Also, we are aware that our automatic evaluation should be complemented with a proper manual evaluation, as done in [14,27], for example. We leave the investigation to reduce the computational time and the manual evaluation as future work.

To further improve CouRGe's counterfactual reasoning, a few options are available. For example, we could look into prompt engineering, i.e. design

further manual prompts and automatic prompts [12]. Also, because our training framework enjoys generality, we could employ bigger language models from the GPT family (e.g. GPT3); or employ different families of models such as T5 [19] and BERT [2] in place of GPT2.

This work can be extended in some other ways. For example, we might use CouRGe's counterfactuals to augment the training set of a sentiment classifier and increase generalisation (like in [8,27]); we could reproduce the same study of this paper, but framed for a different downstream task like Natural Language Inference (similarly to what is done in [8] for example).

Acknowledgements. This publication has emanated from research conducted with the financial support of Science Foundation Ireland under Grant number 12/RC/2289-P2 which is co-funded under the European Regional Development Fund. For the purpose of Open Access, the author has applied a CC BY public copyright licence to any Author Accepted Manuscript version arising from this submission.

References

1. Dathathri, S., et al.: Plug and play language models: a simple approach to controlled text generation. In: International Conference on Learning Representations (2020)
2. Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. CoRR abs/1810.04805 (2018)
3. Freiesleben, T.: The intriguing relation between counterfactual explanations and adversarial examples. Minds Mach. **32**(1), 77–109 (2022)
4. Gardner, M., et al.: Evaluating models' local decision boundaries via contrast sets. arXiv preprint arXiv:2004.02709 (2020)
5. Garg, S., Perot, V., Limtiaco, N., Taly, A., Chi, E.H., Beutel, A.: Counterfactual fairness in text classification through robustness. In: Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society, pp. 219–226 (2019)
6. Goyal, Y., Wu, Z., Ernst, J., Batra, D., Parikh, D., Lee, S.: Counterfactual visual explanations. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 97, pp. 2376–2384. PMLR (2019)
7. Guidotti, R.: Counterfactual explanations and how to find them: literature review and benchmarking. In: Data Mining and Knowledge Discovery, pp. 1–55 (2022)
8. Kaushik, D., Hovy, E., Lipton, Z.: Learning the difference that makes a difference with counterfactually-augmented data. In: International Conference on Learning Representations (2019)
9. Kenny, E.M., Keane, M.T.: On generating plausible counterfactual and semi-factual explanations for deep learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, pp. 11575–11585 (2021)
10. Kusner, M.J., Loftus, J., Russell, C., Silva, R.: Counterfactual fairness. In: Guyon, I., et al. (eds.) Advances in Neural Information Processing Systems, vol. 30. Curran Associates, Inc. (2017)
11. Levenshtein, V.I., et al.: Binary codes capable of correcting deletions, insertions, and reversals. In: Soviet Physics Doklady, vol. 10, pp. 707–710. Soviet Union (1966)

12. Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., Neubig, G.: Pre-train, prompt, and predict: a systematic survey of prompting methods in natural language processing. arXiv preprint arXiv:2107.13586 (2021)
13. Liu, Q., Kusner, M., Blunsom, P.: Counterfactual data augmentation for neural machine translation. In: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 187–197 (2021)
14. Madaan, N., Padhi, I., Panwar, N., Saha, D.: Generate your counterfactuals: towards controlled counterfactual generation for text. In: AAAI (2021)
15. Papineni, K., Roukos, S., Ward, T., Zhu, W.J.: Bleu: a method for automatic evaluation of machine translation. In: ACL (2002)
16. Pearl, J., Mackenzie, D.: The Book of Why: The New Science of Cause and Effect, 1st edn. Basic Books Inc., New York (2018)
17. Qin, L., Bosselut, A., Holtzman, A., Bhagavatula, C., Clark, E., Choi, Y.: 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) (2019)
18. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al.: Language models are unsupervised multitask learners. OpenAI blog **1**(8), 9 (2019)
19. Raffel, C., et al.: Exploring the limits of transfer learning with a unified text-to-text transformer. J. Mach. Learn. Res. **21**(140), 1–67 (2020)
20. Rissland, E.L.: Example-based reasoning. In: Informal Reasoning and Education, pp. 205–226. Routledge (2012)
21. Ross, A., Marasović, A., Peters, M.E.: Explaining nlp models via minimal contrastive editing (mice). arXiv preprint arXiv:2012.13985 (2020)
22. Temraz, M., Keane, M.T.: Solving the class imbalance problem using a counterfactual method for data augmentation. In: Machine Learning with Applications (2022)
23. Teney, D., Abbasnedjad, E., van den Hengel, A.: Learning what makes a difference from counterfactual examples and gradient supervision. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M. (eds.) ECCV 2020. LNCS, vol. 12355, pp. 580–599. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58607-2_34
24. Vaswani, A., et al.: Attention is all you need. In: Guyon, I., et al. (eds.) Advances in Neural Information Processing Systems, vol. 30. Curran Associates, Inc. (2017)
25. Verma, S., Dickerson, J., Hines, K.: Counterfactual explanations for machine learning: a review. arXiv preprint arXiv:2010.10596 (2020)
26. Wang, Z., Culotta, A.: Robustness to spurious correlations in text classification via automatically generated counterfactuals. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, pp. 14024–14031 (2021)
27. Wu, T., Ribeiro, M.T., Heer, J., Weld, D.S.: Polyjuice: generating counterfactuals for explaining, evaluating, and improving models. arXiv preprint arXiv:2101.00288 (2021)
28. Yang, F., Liu, N., Du, M., Hu, X.: Generative counterfactuals for neural networks via attribute-informed perturbation. ACM SIGKDD Explor. Newsl. **23**(1), 59–68 (2021)
29. Zhang, K., Shasha, D.: Simple fast algorithms for the editing distance between trees and related problems. SIAM J. Comput. **18**, 1245–1262 (1989)
30. Zhu, Y., et al.: Texygen: a benchmarking platform for text generation models. In: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, pp. 1097–1100 (2018)