***K*eywords** Counterfactual Explanations • Explainable AI

1 Abstract

Counterfactual Explanations (CE) have emerged as a popular method to explain the predictions made by opaque machine learning models in a post-hoc fashion. We propose a novel approach that leverages counterfactuals during the training phase of models.

2 Introduction

3 Related Literature

3.1 Background on Counterfactual Explanations

(Wachter, Mittelstadt, and Russell 2017; Joshi et al. 2019; Altmeyer et al. 2024)

3.2 Learning Representations

For example, joint-energy models

3.3 Generalization and Robustness

Sauer and Geiger (2021) generate counterfactual images for MNIST and ImageNet through independent mechanisms (IM): each IM learns class-conditional input distributions over a specific lower-dimensional, semantically meaningful factor, such as *texture*, *shape* and *background*. The demonstrate that using these generated counterfactuals during classifier training improves model robustness. Similarly, Abbasnejad et al. (2020) argue that counterfactuals represent potentially useful training data in machine learning, especially in supervised settings where inputs may be reasonably mapped to multiple outputs. They, too, demonstrate the augmenting the training data of image classifiers can improve generalization.

Teney, Abbasnedjad, and Hengel (2020) propose an approach using counterfactuals in training that does not rely on data augmentation: they argue that counterfactual pairs typically already exist in training datasets. Specifically, their approach relies on, firstly, identifying similar input samples with different annotations and, secondly, ensuring that the gradient of the classifier aligns with the vector between pairs of counterfactual inputs using the cosine distance as a loss function (referred to as *gradient supervision*) (***this might be useful for our task as well***). In the natural language processing (NLP) domain, counterfactuals have similarly been used to improve models through data augmentation: Wu et al. (2021), propose POLYJUICE, a general-purpose counterfactual generator for language models. They demonstrate empirically that augmenting training data through POLYJUICE counterfactuals improves robustness in a number of NLP tasks.

3.4 Link to Adversarial Training

Freiesleben (2022) propose two definitional differences between Adversarial Examples (AE) and Counterfactual Explanations (CE): firstly, and more importantly according to the authors, the term AE implies missclassification, which is not the case for CE (***this might be a useful notion for use to distinguish between adversarials and explanations during training***); secondly, they argue that closeness plays a more critical role in the context of CE but confess that even counterfactuals that are not close might be relevant explanations. Pawelczyk et al. (2022) show that CE and AE are equivalent under certain conditions and derive upper bounds on the distances between them.

3.5 Closely Related

Guo, Nguyen, and Yadav (2023) are the first to propose end-to-end training pipeline that includes counterfactual explanations as part of the training prodeduce. In particular, they propose a specific network architecture that includes a predictor and CE generator network (***akin a GAN?***), where the parameters of the CE generator network are learnable. Counterfactuals are generated during each training iteration and fed back to the predictor network (***here we are aligned***). In contrast, we impose no restrictions on the neural network architecture at all. (***to ensure the one-hot encoding of categorical features is maintained, they simple use softmax (might be interesting for CE.jl)***) Interestingly, the authors find that their approach is sensitive to the choice of the loss function: only MSE seems to lead to good performance. They also demonstrate theoretically, that the objective function is difficult to optimize due to divergent gradients and suffers from poor adversarial robustness. (***because partial gradients with respect to the classification loss component and the counterfactual validity component point in opposite directions***). To mitigate these issues, the authors use block-wise gradient descent: they first update with respect to classification loss and then use a second update with respect to the other loss components (***this might be useful for our task as well***). Ross, Lakkaraju, and Bastani (2024) propose a way to train models that are guaranteed to provide recourse for individuals with high probability. The approach builds on adversarial training (***here we are aligned***), where in this context adversarial examples are actively encouraged to exist, but only target attacks with respect to the positive class. The proposed method allows for imposing a set of actionable recourse ex-ante: for example, users can impose mutability constraints for features (***here we are aligned***). (***To solve their objective function more efficiently, they use a first-order Taylor approximation to approximate the recourse loss component (might be applicable in our case)***)

Luu and Inoue (2023) introduce Counterfactual Adversarial Training (CAT) with intention of improving generalization and robustness of language models. Specifically, they propose to proceed as follows: firstly, identify training samples that are subject to high predictive uncertainty (entropy); secondly, generate counterfactual explanations for those samples; and, finally, finetune the model on the augmented dataset that includes the generated counterfactuals.

4 Counterfactual Training

5 Experiments

5.1 Experimental Setup

5.2 Experimental Results

6 Discussion

7 Conclusion

8 Appendix

8.1 Training Details

8.2 Initial Grid Search

For the initial round of experiments we

8.2.1 Generator Params

- **generator_params**:
  - **lambda_cost**: 0.0, 0.001, 0.1
  - **lambda_energy**: 0.01, 0.05, 0.1, 0.5, 1.0, 5.0, 10.0, 15.0
  - **lr**: 1.0
  - **maxiter**: 20, 50, 100
  - **opt**: sgd
- **generator_type**: ecco, generic, omni, revise
- **training_params**:
  - **objective**: full, vanilla

8.2.1.1 Linearly Separable

- **Energy Penalty** (Table 1): *ECCo* generally does yield better results than *Vanilla* for higher choices of the energy penalty (10,15) during training. *Generic* performs poorly accross the board. *Omni* seems to have an anchoring effect, in that it never performs terribly but also never as good as the best *ECCo* results. *REVISE* performs poorly across the board.
- **Cost (distance penalty)**: Results for all generators (except *Omni*) are quite bad, which can likely be attributed to extremely bad results for some choices of the **Energy Penalty** (results here are averaged). For *ECCo* and *Generic*, higher cost values generally lead to worse results.
- **Maximum Iterations**: No clear patterns recognizable, so it seems that smaller choices are ok.
- **Validity**: *ECCo* almost always valid except for very low values during training and high values at evaluation time. *Generic* often has poor validity.
- **Accuracy**: Seems largely unaffected.

Table 1: Results for Linearly Separable data by energy penalty.

Objective	λ_{div}**(train)**	**Generator**	**Value**	**Std**
full	0.01	*ECCo*	$-9.91 \cdot 10^{11}$	$2.25 \cdot 10^{12}$
full	0.01	*Generic*	$-5.71 \cdot 10^{17}$	$1.3 \cdot 10^{18}$

Continuing table below.

Objective	λ_{div}(train)	Generator	Value	Std
full	**0.01**	**Omniscient**	**-2.54**	**0.116**
full	0.01	*REVISE*	-15.6	13.2
vanilla	0.01	*ECCo*	-4.28	3.52
vanilla	0.01	*Generic*	-4.45	3.47
vanilla	0.01	*Omniscient*	-5.12	4.46
vanilla	0.01	*REVISE*	-4.91	4.24
full	0.05	*ECCo*	$-5.63 \cdot 10^5$	$1.28 \cdot 10^6$
full	0.05	*Generic*	$-8.35 \cdot 10^{17}$	$1.9 \cdot 10^{18}$
full	**0.05**	**Omniscient**	**-2.53**	**0.114**
full	0.05	*REVISE*	-15	12.6
vanilla	0.05	*ECCo*	-4.4	3.66
vanilla	0.05	*Generic*	-4.38	3.48
vanilla	0.05	*Omniscient*	-5.25	4.62
vanilla	0.05	*REVISE*	-4.94	4.22
full	0.1	*ECCo*	$-6.74 \cdot 10^5$	$1.53 \cdot 10^6$
full	0.1	*Generic*	$-1.72 \cdot 10^{11}$	$3.9 \cdot 10^{11}$
full	**0.1**	**Omniscient**	**-2.56**	**0.124**
full	0.1	*REVISE*	-15.6	13.2
vanilla	0.1	*ECCo*	-4.28	3.52
vanilla	0.1	*Generic*	-4.45	3.48
vanilla	0.1	*Omniscient*	-5.12	4.46
vanilla	0.1	*REVISE*	-4.91	4.25
full	0.5	*ECCo*	-11.8	9.83
full	0.5	*Generic*	$-1.06 \cdot 10^{18}$	$2.42 \cdot 10^{18}$
full	**0.5**	**Omniscient**	**-2.54**	**0.123**
full	0.5	*REVISE*	-15	12.6
vanilla	0.5	*ECCo*	-4.4	3.65
vanilla	0.5	*Generic*	-4.38	3.48
vanilla	0.5	*Omniscient*	-5.25	4.61
vanilla	0.5	*REVISE*	-4.95	4.22
full	1	*ECCo*	-11.5	11.1
full	1	*Generic*	$-1.71 \cdot 10^{11}$	$3.88 \cdot 10^{11}$
full	**1**	**Omniscient**	**-2.59**	**0.117**
full	1	*REVISE*	-15.7	13.3
vanilla	1	*ECCo*	-4.28	3.51
vanilla	1	*Generic*	-4.44	3.47
vanilla	1	*Omniscient*	-5.11	4.46
vanilla	1	*REVISE*	-4.91	4.25
full	5	*ECCo*	-3.99	3.12
full	5	*Generic*	$-4.88 \cdot 10^{17}$	$1.11 \cdot 10^{18}$
full	**5**	**Omniscient**	**-2.53**	**0.117**
full	5	*REVISE*	-14.6	12.1
vanilla	5	*ECCo*	-4.4	3.65
vanilla	5	*Generic*	-4.38	3.48
vanilla	5	*Omniscient*	-5.25	4.61
vanilla	5	*REVISE*	-4.95	4.22
full	**10**	**ECCo**	**-2.31**	**0.735**
full	10	*Generic*	$-1.7 \cdot 10^{11}$	$3.86 \cdot 10^{11}$
full	10	*Omniscient*	-2.53	0.117
full	10	*REVISE*	-15.5	13
vanilla	10	*ECCo*	-4.28	3.51
vanilla	10	*Generic*	-4.44	3.47
vanilla	10	*Omniscient*	-5.12	4.46
vanilla	10	*REVISE*	-4.91	4.24
full	**15**	**ECCo**	**-2.01**	**0.488**

Continuing table below.

Objective	λ_{div}(train)	Generator	Value	Std
full	15	*Generic*	$-4.91 \cdot 10^{17}$	$1.12 \cdot 10^{18}$
full	15	*Omniscient*	-2.53	0.116
full	15	*REVISE*	-14.4	11.7
vanilla	15	*ECCo*	-4.4	3.65
vanilla	15	*Generic*	-4.38	3.48
vanilla	15	*Omniscient*	-5.25	4.6
vanilla	15	*REVISE*	-4.95	4.23

8.2.1.2 Moons

- **Energy Penalty** (Table 2): *ECCo* consistently yields better results than *Vanilla*, except for very low choices of the energy penalty during training for which it performs abismal. *Generic* performs quite badly across the board for high enough choices of the energy penalty at evaluation time. *Omni* has small positive effect. *REVISE* performs poorly across the board.
- **Cost (distance penalty)**: *Generic* generally does better for higher values, while *ECCo* does better for lower values.
- **Maximum Iterations**: No clear patterns recognizable, so it seems that smaller choices are ok.
- **Validity**: *ECCo* generally achieves full validity except for very low choices the energy penalty during training and high choices at evaluation time. *Generic* performs poorly for high choices of the energy penalty during evaluation.
- **Accuracy**: Largely unaffected although *ECCo* suffers a bit for very low choices the energy penalty during training. *REVISE* suffers a lot in general (around 10 percentage points).

Table 2: Results for Moons data by energy penalty.

Objective	λ_{div}(train)	Generator	Value	Std
full	0.01	*ECCo*	$-2.8 \cdot 10^{22}$	$6.39 \cdot 10^{22}$
full	0.01	*Generic*	$-4.89 \cdot 10^{30}$	$1.11 \cdot 10^{31}$
full	**0.01**	**Omniscient**	**-4.74**	**5.08**
full	0.01	*REVISE*	-572	$1.25 \cdot 10^{3}$
vanilla	0.01	*ECCo*	-15.5	17.3
vanilla	0.01	*Generic*	-10.9	11.9
vanilla	0.01	*Omniscient*	-12.7	14.4
vanilla	0.01	*REVISE*	-11.2	13
full	0.05	*ECCo*	$-1.55 \cdot 10^{16}$	$3.52 \cdot 10^{16}$
full	0.05	*Generic*	$-2.22 \cdot 10^{20}$	$5 \cdot 10^{20}$
full	**0.05**	**Omniscient**	**-4.41**	**4.48**
full	0.05	*REVISE*	$-1.04 \cdot 10^{3}$	$2.3 \cdot 10^{3}$
vanilla	0.05	*ECCo*	-15.5	17.2
vanilla	0.05	*Generic*	-11.7	12.8
vanilla	0.05	*Omniscient*	-12.4	14.1
vanilla	0.05	*REVISE*	-11.3	13.1
full	0.1	*ECCo*	$-3.41 \cdot 10^{3}$	$7.73 \cdot 10^{3}$
full	0.1	*Generic*	$-5.22 \cdot 10^{30}$	$1.19 \cdot 10^{31}$
full	**0.1**	**Omniscient**	**-4.78**	**5.12**
full	0.1	*REVISE*	-288	594
vanilla	0.1	*ECCo*	-15.5	17.2
vanilla	0.1	*Generic*	-10.9	11.9
vanilla	0.1	*Omniscient*	-12.7	14.4
vanilla	0.1	*REVISE*	-11.3	13.1
full	0.5	*ECCo*	-7.09	7.51
full	0.5	*Generic*	$-1.11 \cdot 10^{31}$	$2.53 \cdot 10^{31}$
full	**0.5**	**Omniscient**	**-4.58**	**4.83**
full	0.5	*REVISE*	$-1.19 \cdot 10^{3}$	$2.64 \cdot 10^{3}$

Continuing table below.

Objective	λ_{div}(train)	Generator	Value	Std
vanilla	0.5	*ECCo*	-15.5	17.2
vanilla	0.5	*Generic*	-11.7	12.8
vanilla	0.5	*Omniscient*	-12.4	14.1
vanilla	0.5	*REVISE*	-11.3	13.1
full	1	*ECCo*	-6.06	6.33
full	1	*Generic*	$-1.58 \cdot 10^{33}$	$3.59 \cdot 10^{33}$
full	**1**	**Omniscient**	**-4.66**	**4.89**
full	1	*REVISE*	$-1.16 \cdot 10^{3}$	$2.59 \cdot 10^{3}$
vanilla	1	*ECCo*	-15.5	17.3
vanilla	1	*Generic*	-10.9	11.9
vanilla	1	*Omniscient*	-12.7	14.4
vanilla	1	*REVISE*	-11.3	13.1
full	**5**	**ECCo**	**-2.57**	**2.07**
full	5	*Generic*	$-1.17 \cdot 10^{28}$	$2.66 \cdot 10^{28}$
full	5	*Omniscient*	-4.29	4.31
full	5	*REVISE*	-530	$1.16 \cdot 10^{3}$
vanilla	5	*ECCo*	-15.5	17.2
vanilla	5	*Generic*	-11.7	12.7
vanilla	5	*Omniscient*	-12.4	14.1
vanilla	5	*REVISE*	-11.3	13.1
full	**10**	**ECCo**	**-1.76**	**0.974**
full	10	*Generic*	$-1.54 \cdot 10^{33}$	$3.51 \cdot 10^{33}$
full	10	*Omniscient*	-4.44	4.56
full	10	*REVISE*	$-1.52 \cdot 10^{3}$	$3.4 \cdot 10^{3}$
vanilla	10	*ECCo*	-15.5	17.3
vanilla	10	*Generic*	-10.9	11.9
vanilla	10	*Omniscient*	-12.7	14.4
vanilla	10	*REVISE*	-11.3	13.1
full	**15**	**ECCo**	**-1.37**	**0.365**
full	15	*Generic*	$-5.32 \cdot 10^{28}$	$1.21 \cdot 10^{29}$
full	15	*Omniscient*	-4.34	4.38
full	15	*REVISE*	-473	$1.03 \cdot 10^{3}$
vanilla	15	*ECCo*	-15.5	17.2
vanilla	15	*Generic*	-11.7	12.8
vanilla	15	*Omniscient*	-12.4	14.1
vanilla	15	*REVISE*	-11.3	13.1

8.2.1.3 Circles

- **Energy Penalty** (Table 3): *ECCo* consistently yields better results than *Vanilla*, though primarily for low to medium choices of the energy penalty (<=5) during training. The same goes for *Generic*, which sometimes outperforms *ECCo* (for small energy penalty at evaluation time). *Omni* does alright for lower energy penalty at evaluation time, but loses out for higher choices. *REVISE* performs poorly across the board (except very low choices at evaluation time).
- **Cost (distance penalty)**: *ECCo* and *Generic* generally achieve the best results when no cost penalty is used during training. Both *Omni* and *REVISE* are largely unaffected.
- **Maximum Iterations**: *ECCo* consistently yields better results for higher numbers of iterations. *Generic* generally does best for a medium number (50). *Omni* is sometimes invalid (**???**).
- **Validity**: *ECCo* tends to outperform its *Vanilla* counterpart, though primarily for low to medium choices of the energy penalty (<=5) during training and evaluation. *Vanilla* typically worse across the board.
- **Accuracy**: Mostly unaffected, but *REVISE* again consistently some deterioration and *ECCo* deteriorates for high choices of energy penalty during training, reflecting other outcomes above.

Table 3: Results for Circles data by energy penalty.

Objective	λ_{div}(**train**)	**Generator**	**Value**	**Std**
full	**0.01**	**ECCo**	**-1.26**	**0.423**
full	0.01	*Generic*	-1.49	0.71
full	0.01	*Omniscient*	-5.21	5.25
full	0.01	*REVISE*	$-2.71 \cdot 10^{26}$	$6.37 \cdot 10^{26}$
vanilla	0.01	*ECCo*	-9.33	7.34
vanilla	0.01	*Generic*	-8.89	6.88
vanilla	0.01	*Omniscient*	-8.67	6.87
vanilla	0.01	*REVISE*	-8.65	6.8
full	0.05	*ECCo*	-1.29	0.397
full	**0.05**	**Generic**	**-1.21**	**0.356**
full	0.05	*Omniscient*	-5.08	5.09
full	0.05	*REVISE*	$-5.91 \cdot 10^{27}$	$1.36 \cdot 10^{28}$
vanilla	0.05	*ECCo*	-9.35	7.32
vanilla	0.05	*Generic*	-8.85	6.87
vanilla	0.05	*Omniscient*	-8.7	6.96
vanilla	0.05	*REVISE*	-8.52	6.76
full	**0.1**	**ECCo**	**-1.2**	**0.383**
full	0.1	*Generic*	-1.5	0.735
full	0.1	*Omniscient*	-5.17	5.23
full	0.1	*REVISE*	$-3.06 \cdot 10^{26}$	$7.7 \cdot 10^{26}$
vanilla	0.1	*ECCo*	-9.33	7.32
vanilla	0.1	*Generic*	-8.88	6.86
vanilla	0.1	*Omniscient*	-8.69	6.9
vanilla	0.1	*REVISE*	-8.68	6.81
full	**0.5**	**ECCo**	**-1.12**	**0.217**
full	0.5	*Generic*	-1.21	0.352
full	0.5	*Omniscient*	-5.09	5.12
full	0.5	*REVISE*	$-5.97 \cdot 10^{27}$	$1.37 \cdot 10^{28}$
vanilla	0.5	*ECCo*	-9.35	7.3
vanilla	0.5	*Generic*	-8.89	6.92
vanilla	0.5	*Omniscient*	-8.68	6.93
vanilla	0.5	*REVISE*	-8.53	6.75
full	**1**	**ECCo**	**-1.1**	**0.163**
full	1	*Generic*	-1.49	0.726
full	1	*Omniscient*	-5.16	5.2
full	1	*REVISE*	$-3.09 \cdot 10^{26}$	$7.22 \cdot 10^{26}$
vanilla	1	*ECCo*	-9.34	7.36
vanilla	1	*Generic*	-8.86	6.85
vanilla	1	*Omniscient*	-8.7	6.9
vanilla	1	*REVISE*	-8.69	6.85
full	5	*ECCo*	-1.75	0.154
full	**5**	**Generic**	**-1.21**	**0.363**
full	5	*Omniscient*	-5.14	5.16
full	5	*REVISE*	$-1.1 \cdot 10^{28}$	$2.5 \cdot 10^{28}$
vanilla	5	*ECCo*	-9.36	7.32
vanilla	5	*Generic*	-8.88	6.91
vanilla	5	*Omniscient*	-8.7	6.93
vanilla	5	*REVISE*	-8.52	6.73
full	10	*ECCo*	$-1.02 \cdot 10^{6}$	$2.32 \cdot 10^{6}$
full	**10**	**Generic**	**-1.49**	**0.702**
full	10	*Omniscient*	-5.13	5.16
full	10	*REVISE*	$-3.74 \cdot 10^{26}$	$9.09 \cdot 10^{26}$
vanilla	10	*ECCo*	-9.31	7.33
vanilla	10	*Generic*	-8.87	6.86

Continuing table below.

Objective	λ_{div}(train)	Generator	Value	Std
vanilla	10	*Omniscient*	-8.7	6.89
vanilla	10	*REVISE*	-8.69	6.83
full	15	*ECCo*	$-3.31 \cdot 10^{13}$	$7.54 \cdot 10^{13}$
full	**15**	**Generic**	**-1.22**	**0.37**
full	15	*Omniscient*	-5.2	5.23
full	15	*REVISE*	$-9.01 \cdot 10^{27}$	$2.06 \cdot 10^{28}$
vanilla	15	*ECCo*	-9.38	7.34
vanilla	15	*Generic*	-8.86	6.87
vanilla	15	*Omniscient*	-8.69	6.96
vanilla	15	*REVISE*	-8.51	6.73

References

Abbasnejad, Ehsan, Damien Teney, Amin Parvaneh, Javen Shi, and Anton van den Hengel. 2020. "Counterfactual Vision and Language Learning." In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 10041–51. https://doi.org/10.1109/CVPR42600.2020.01006.

Altmeyer, Patrick, Mojtaba Farmanbar, Arie van Deursen, and Cynthia CS Liem. 2024. "Faithful Model Explanations Through Energy-Constrained Conformal Counterfactuals." In *Proceedings of the AAAI Conference on Artificial Intelligence*, 38:10829–37. 10.

Freiesleben, Timo. 2022. "The Intriguing Relation Between Counterfactual Explanations and Adversarial Examples." *Minds and Machines* 32 (1): 77–109.

Guo, Hangzhi, Thanh H. Nguyen, and Amulya Yadav. 2023. "CounterNet: End-to-End Training of Prediction Aware Counterfactual Explanations." In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 577–89. KDD '23. New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/3580305.3599290.

Joshi, Shalmali, Oluwasanmi Koyejo, Warut Vijitbenjaronk, Been Kim, and Joydeep Ghosh. 2019. "Towards Realistic Individual Recourse and Actionable Explanations in Black-Box Decision Making Systems." https://arxiv.org/abs/1907.09615.

Luu, Hoai Linh, and Naoya Inoue. 2023. "Counterfactual Adversarial Training for Improving Robustness of Pre-Trained Language Models." In *Proceedings of the 37th Pacific Asia Conference on Language, Information and Computation*, 881–88.

Pawelczyk, Martin, Chirag Agarwal, Shalmali Joshi, Sohini Upadhyay, and Himabindu Lakkaraju. 2022. "Exploring Counterfactual Explanations Through the Lens of Adversarial Examples: A Theoretical and Empirical Analysis." In *Proceedings of the 25th International Conference on Artificial Intelligence and Statistics*, edited by Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera, 151:4574–94. Proceedings of Machine Learning Research. PMLR. https://proceedings.mlr.press/v151/pawelczyk22a.html.

Ross, Alexis, Himabindu Lakkaraju, and Osbert Bastani. 2024. "Learning Models for Actionable Recourse." In *Proceedings of the 35th International Conference on Neural Information Processing Systems*. NIPS '21. Red Hook, NY, USA: Curran Associates Inc.

Sauer, Axel, and Andreas Geiger. 2021. "Counterfactual Generative Networks." https://arxiv.org/abs/2101.06046.

Teney, Damien, Ehsan Abbasnedjad, and Anton van den Hengel. 2020. "Learning What Makes a Difference from Counterfactual Examples and Gradient Supervision." In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part x 16*, 580–99. Springer.

Wachter, Sandra, Brent Mittelstadt, and Chris Russell. 2017. "Counterfactual Explanations Without Opening the Black Box: Automated Decisions and the GDPR." *Harv. JL & Tech.* 31: 841. https://doi.org/10.2139/ssrn.3063289.

Wu, Tongshuang, Marco Tulio Ribeiro, Jeffrey Heer, and Daniel Weld. 2021. "Polyjuice: Generating Counterfactuals for Explaining, Evaluating, and Improving Models." In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, edited by Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, 6707–23. Online: Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.acl-long.523.