Counterfactuals of Counterfactuals: a back-translation-inspired approach to analyse counterfactual editors

Giorgos Filandrianos[1] **Edmund Dervakos**[1] **Orfeas Menis-Mastromichalakis**[1]
Chrysoula Zerva[2,3] and **Giorgos Stamou**[1]
[1]National Technical University of Athens
[2]Instituto de Telecomunicações
[3]Instituto Superior Técnico & LUMLIS (Lisbon ELLIS Unit)
{geofila, eddiedervakos}@islab.ntua.gr, menorf@ails.ece.ntua.gr
chrysoula.zerva@tecnico.ulisboa.pt, gstam@cs.ntua.gr

Abstract

In the wake of responsible AI, interpretability methods, which attempt to provide an explanation for the predictions of neural models have seen rapid progress. In this work, we are concerned with explanations that are applicable to natural language processing (NLP) models and tasks, and we focus specifically on the analysis of counterfactual, contrastive explanations. We note that while there have been several explainers proposed to produce counterfactual explanations, their behaviour can vary significantly and the lack of a universal ground truth for the counterfactual edits imposes an insuperable barrier on their evaluation. We propose a new back translation-inspired evaluation methodology that utilises earlier outputs of the explainer as ground truth proxies to investigate the consistency of explainers. We show that by iteratively feeding the counterfactual to the explainer we can obtain valuable insights into the behaviour of both the predictor and the explainer models, and infer patterns that would be otherwise obscured. Using this methodology, we conduct a thorough analysis and propose a novel metric to evaluate the consistency of counterfactual generation approaches with different characteristics across available performance indicators.[1]

1 Introduction

The eXplainable AI (XAI) field has risen to prominence in recent years, spurred on by the success of opaque (black-box) deep learning models, which despite their impressive performance cannot be used in practice in many cases due to ethical and legal (Goodman and Flaxman, 2017) concerns. To mitigate this, multiple explanation methodologies have been proposed for different tasks, data domains and use-cases (Bodria et al., 2021). One family of methods for explaining classifiers are *counterfactual* or *contrastive* explanations (Verma et al., 2020; Stepin et al., 2021; Balkir et al., 2022). These answer the question "What should change in an input sample for it to be classified to class B instead of class A" and can be especially useful in real-world applications as they offer recourse to a user. For example someone who was declined a loan by a bank's AI could ask "what is the smallest change I should do for my loan to be approved"

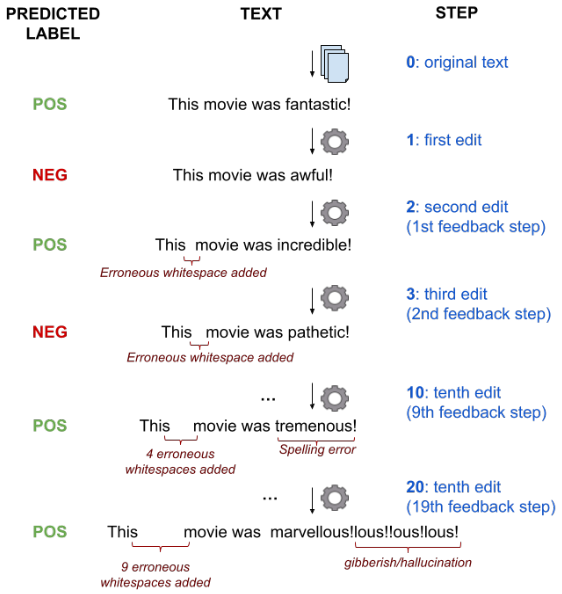

PREDICTED LABEL
TEXT
STEP
0: original text
POS
This movie was fantastic!
1: first edit
NEG
This movie was awful!
2: second edit (1st feedback step)
POS
This movie was incredible!
Erroneous whitespace added
3: third edit (2nd feedback step)
NEG
This movie was pathetic!
Erroneous whitespace added
...
10: tenth edit (9th feedback step)
POS
This movie was tremenous!
4 erroneous whitespaces added
Spelling error
...
20: tenth edit (19th feedback step)
POS
This movie was marvellous!lous!!ous!lous!
9 erroneous whitespaces added
gibberish/hallucination

Figure 1: Using the back-translation framework to feed back the edited text to MiCE: We see the evolution of edits (centre) and predicted labels (left) through multiple feedback steps (right). As feedback steps increase, we observe an amplification of erroneous edits.

[1] Data and Code available at: https://github.com/geofila/Counterfactuals-of-Counterfactuals

Findings of the Association for Computational Linguistics: ACL 2023, pages 9507–9525
July 9-14, 2023 ©2023 Association for Computational Linguistics

(Wachter et al., 2017).

However, many XAI methods have come under scrutiny, as they can often be misleading and cause more problems than they solve (Rudin, 2019). Specifically counterfactual explainers have been shown to suffer from issues of robustness (Slack et al., 2021; Rawal et al., 2020; Delaney et al., 2021), and while there have been attempts to formalise notions of interpretability and its evaluation (Doshi-Velez and Kim, 2017), there is no agreement on what constitutes a good explanation (Lipton, 2018). This ambiguity regarding the desiderata for a good explanation and the fact that they can vary according to the use-case, transcends to the chosen evaluation methods used, which vary accordingly and offer a limited perspective of the editor(s) behaviour. Thus it is imperative that we develop further ways to evaluate and compare XAI methods in more depth.

For counterfactual explanations, existing metrics enable comparisons between different methods, however the absence of a ground truth does not allow us to assess the quality of a single explainer in a standalone mode and evaluate its output on how close it is to a (theoretically attainable) ideal explanation. Finding such an ideal explanation in practice is not easy, but we can expand on the idea of evaluation by comparison, and compare a counterfactual system's performance against itself. Following this rationale, we propose a methodology that is inspired by back-translation, which has been used for evaluating and improving machine translation systems (Tyupa, 2011; Edunov et al., 2018). Specifically, by feeding the output of the system to itself (a counterfactual of a counterfactual), we would expect the result to be "at least as good as" the original input since we know that the original input exists, is actionable, and feasible, thus it constitutes a "lower bound" for the generated edit, and a proxy for ground truth.

This methodology can be applied to obtain a lower bound on several metrics; in this work we focus on its application to *minimality* since it is the metric that most of the editors attempt to minimise as a primary criterion (Guidotti, 2022). One of the desired characteristics of counterfactual explanations is that they constitute minimal changes to the input sample, and minimality is the metric used to measure the distance between the original and the edited samples. Due to the lack of an ideal explanation to be used as ground truth, there is no way to know if a specific value of minimality is optimal (and generally good or bad). Given a ground truth explanation, it is possible to calculate the optimal minimality to be obtained, but without it, it is only possible to compare minimality values across different edits/editors.

Using our proposed methodology we introduce a metric (which we call *inconsistency*) that uses the editor's previous outputs as reference points to evaluate an editor's capability to produce minimal edits. We feed the output of the editor back to it as input to produce a new edit, and we expect the new edit to be at least as good as the edit of the previous step. For example, in Figure 1 we see the outputs through different steps of our feedback loop approach, so when we feed the first edit ("This movie was awful!") back to the counterfactual system, we expect the generated edit to be at least as good as the original text ("This movie was fantastic!"). However, we see that the editor adds an erroneous whitespace to the generated edit (see 2:second edit in Figure 1) so we know that there exists a better output that the system was not able to find, thus for sure the system did not produce the optimal output. Note that a counterfactual system with a non-zero value of inconsistency is guaranteed to be sub-optimal, however a zero value does not indicate that it is optimal. This approach essentially sets a lower bound for the editor, but we do not have access to the higher bound which could be impossible to define automatically. In the rest of the paper we provide a detailed description of our proposed methodology and novel metric and demonstrate its application on several frequently used editors with different characteristics.

2 Background

In this paper we are concerned with systems that attempt to minimally edit a given textual input so that they change the prediction of a classifier; we will henceforth refer to such systems as *counterfactual editors*, or simply *editors*. Below, we provide a categorisation of such systems, along with related literature, in addition to an overview of how they are being evaluated in related works.

2.1 Counterfactual Editors

The intended use-case of editors varies, as do the methodologies they use for achieving their goal. For example, approaches such as MiCE (Ross et al., 2020), and DoCoGen (Calderon et al., 2022) optimise their edits directly on the output of a given predictor, $g()$ by pseudo-randomly masking words in the text and optimising the proposed replacement to change the output of g. On the contrary, editors such as Polyjuice (Wu et al., 2021) aim to identify generic text perturbations that can change the semantics of a sentence, without targeting a specific predictor. They frame this as *general purpose counterfactuals* since they can be used for a wider range of purposes, from data-augmentation to producing counterfactual explanations or conditioned to a specific task/dataset. Finally, a large family of editors aim to generate *adversarial examples*, whose intended use-case is to identify vulnerabilities of a classifier and expose them. Adversarial models may differ from other counterfactual editors in that they do not necessarily aim to generate a minimal or fluent edit of the original input, hence the edits might include addition of noise, etc. A collection of adversarial example generators for NLP including TextFooler (Jin et al., 2020) and Bert-Attack (Li et al., 2020), is implemented in the TextAttack framework (Morris et al., 2020). The intuitively simpler form of such methods concerns using gradient-descent on the instance to generate examples that alter the predictor's class while simultaneously optimising the value of one or more metrics (Mothilal et al., 2020). Instead of attempting random permutations to generate counterexamples, other editors alter only the important features of each text. This importance is calculated in a variety of ways, including training a classifier to extract the correlation of each term with the task (Wang and Culotta, 2021), measuring the effect of a feature deletion on the prediction of the classifier (Jin et al., 2020), or using the predictor's attention (Ross et al., 2020). Then the important terms can be replaced with synonyms, antonyms, important terms from other tasks or using pre-trained seq2seq models (Madaan et al., 2021; Ross et al., 2021, 2020; Wu et al., 2021; Fern and Pope, 2021). In our experiments, we employed editors with different intended use-cases and internal logic, namely MiCE, Polyjuice and TextFooler.

2.2 Evaluation of Counterfactual Editors

A practical criterion for evaluating editors measures how often the output of a predictor flips to the desired class, referred to as flip-rate, validity, fidelity or attack success rate. Other metrics relate to the quality of generated text and include fluency, as used for example in MiCE, Polyjuice , and CAT (Chemmengath et al., 2021), grammaticality and semantics as defined in Textattack and perceptibility as defined in counterfactualGAN (Robeer et al., 2021). These metrics rely on the use of language models, either comparing masked language model loss between original and edited text, or computing a semantic similarity between original and edited text. Proximity as described in (Verma et al., 2020; Keane et al., 2021) refers to generic distance measures between edited and original data. For natural language processing, the distance metric used is typically the word level Levenshtein edit distance (Levenshtein et al., 1966), referred to also as minimality (MiCE), closeness (Polyjuice) and edit distance (CAT). There are more criteria that have been used for the evaluation of counterfactuals, such as sparcity (Keane et al., 2021) referring to the number of features being changed, and closeness to the training data (Verma et al., 2020) or relative distance (Keane et al., 2021) involve comparing the explanations with instances from the training data. Finally, a more recent approach for evaluating counterfactual explanations involves measuring the degree to which the explanations help a student model (Pruthi et al., 2022) or a human (Doshi-Velez and Kim, 2017; Treviso and Martins, 2020) to learn to simulate a black-box teacher model. This provides a measure of informativeness of the explanations.

In this work we focus on evaluation with automated metrics that do not require human input or external data, and are most frequently used in editor evaluation, namely, minimality, flip-rate and fluency. These metrics aim to quantify different aspects of editors' behaviour in a comparative fashion. Instead, we propose an inconsistency metric which allows to set a lower bound on the editor's ability to reach an optimal counterfactual and as such can be useful without the need to compare to other editors. For instance, a minimality value for a given editor carries no information on its own regarding optimality. However, if an editor has a value $inc@1 = x$ for the inconsistency of minimality, it means that on a given test-set it missed the optimal counterfactual solution at least by an average of x tokens.

3 Back-translation for analyzing editors

We formalise our problem as follows. We assume access to a classifier g such that $g : \mathcal{L} \rightarrow [0, 1]^C$, where $\mathcal{L}$ the set of text for a specific language and C

is the number of different classes. We then consider the counterfactual editors for g as functions $f : \mathcal{L} \rightarrow \mathcal{L}$, and we assume that the goal of the editor f is threefold:

1. The edited text is classified to a different class $\arg\max g(f(x)) \neq \arg\max g(x)$.

2. The edits are minimal with respect to some distance metric d: $f = \arg\min_{h \in \mathcal{F}} d(x, h(x))$, where $\mathcal{F}$ is the set of functions for which $\arg\max g(f(x)) \neq \arg\max g(x)$.

3. The edited text $f(x)$ is fluent and within the distribution of $\mathcal{L}$.

To examine the degree to which these criteria hold, we analyse the behaviour of editors when they are iteratively fed back with their output, i.e., we are studying the function $f(f(f(...f(x))))$, and evaluating the three criteria described above after n applications of the editor. Specifically, we first define a novel evaluation metric to quantify the second criterion based on the iterative feedback approach, and then we discuss how the first and third criteria can be more thoroughly checked by measuring performance metrics after n steps of feedback (notated as metric@n).

3.1 (In)consistency of minimality

Intuitively, since the edits are ideally minimal, if a sentence A is edited into sentence B and their distance is $d(A, B)$, then feeding back sentence B to the editor should yield a sentence C for which $d(B, C) \leq d(A, B)$, otherwise C is not the result of a minimal edit. This inequality holds based on that (a) we know that A exists, (b) we assume all textual edits to be reversible, hense A is reachable from B and (c) d is symmetric, meaning $d(A, B) = d(B, A)$. Thus, in this case, A can be used as a proxy to a ground truth, to be compared with C. Given a distance metric d (such as Levenshtein distance, embedding cosine similarity, etc.), we can measure how consistent the counterfactual editor is w.r.t d by iteratively feeding back the edited text to the editor and measuring the change in the value of d. Specifically, given an editor $f : \mathcal{L} \rightarrow \mathcal{L}$, a text $x \in \mathcal{L}$ and a distance $d : \mathcal{L} \times \mathcal{L} \rightarrow \mathbb{R}^+$ we define the *inconsistency of f with respect to d, for x as:*.

$$\mathsf{inc}(f, x) = \mathsf{relu}[d(f(f(x)), f(x)) - d(f(x), x)]. \quad (1)$$

The difference $d(f(f(x)), f(x)) - d(f(x), x)$ shows how much the distance d changes between consecutive applications of the editor f and the relu function allows to take into account only the increase of the distance. This is important, because a decrease in the distance, which would correspond to a negative difference, is not necessarily an indicator of a good set of edits. It could, for example, indicate that not enough changes were made, and there is no way to know if that is the case, or if a better, more minimal set of edits was found. Contrarily, when the value is positive, we have a *guarantee* that a better set of edits exists, namely, the one of the previous feedback step. Equation 1 counts the difference in d after a single feedback iteration though the editor, but as with other metrics in this work, we can keep feeding back the output of the editor to itself, and compute $\mathsf{inc}(f, f(x))$ to get more information about the editor's inconsistency. When we do this, we measure the average inconsistency after n steps of feedback as:

$$\mathsf{inc@}n(f, x) = \frac{1}{n} \sum_{i=0}^{n-1} \mathsf{inc}(f_{i+1}(x), f_i(x)), \quad (2)$$

where $f_0(x) = x$ and $f_i(x) = f(f_{i-1}(x))$.

3.2 Diverging from the distribution

Of the desiderata for counterfactual editors, the constraint that $f(x)$ is fluent and within distribution can be hard to verify, as the true distribution of texts $\mathcal{L}$ may be inaccessible, and fluency is hard to evaluate in a systematic and automated way. The token-level perplexity of a large language model is a frequently used proxy to the fluency estimation, employed in multiple NLP tasks (John et al., 2018; Wang and Cho, 2019; Vasilakes et al., 2022). It involves using a language model $\mathcal{M}_\mathcal{D}$ trained on a large dataset $\mathcal{D}$ and computing the averaged perplexity over a given sequence of text $x = x_1, x_2, ..., x_T$ as follows:

$$\mathrm{PPL}(x) = \exp\left\{\frac{1}{T}\sum_{t=1}^{T} \log p_{\mathcal{M}_\mathcal{D}}(x_t | x_{1:t-1})\right\}. \quad (3)$$

Note that for the fluency estimation $\mathcal{M}_\mathcal{D}$ in equation 3 is not finetuned on $\mathcal{L}$. Assuming $\mathcal{L}$ is accessible, we can also fine-tune $\mathcal{M}_\mathcal{D}$ on it, obtaining a model $\mathcal{M}_\mathcal{L}$ that we can use to detect out-of-distribution (OOD) cases (Arora et al., 2021) using the same PPL formula described in equation

3, since in this case we can assume that the probability over each token predicted by $\mathcal{M}_{\mathcal{L}}$ reflects the probability under the distribution of $\mathcal{L}$. We employ and compare both PPL over $\mathcal{M}_{\mathcal{D}}$ (PPL$_{\mathcal{D}}$) and PPL over $\mathcal{M}_{\mathcal{L}}$ (PPL$_{\mathcal{L}}$) in our experiments. Furthermore, PPL$_{\mathcal{L}}$@n and PPL$_{\mathcal{D}}$@n could more clearly show the divergence from the distribution of generated samples, and the deterioration of fluency.

3.3 Flip rate

So far we have not taken under consideration the ability of the editor to change the predicted class, which is typically measured as flip rate. Many recently proposed counterfactual editors achieve flip rates above 95%, leaving a small margin to confidently compare editors with respect to this metric. Using the proposed feedback methodology, we can measure the flip rate after n loops of feedback to get more detailed information about the ability of the editor to consistently change the predicted class after several perturbations of the sentence.

4 Experimental Setup

We evaluate our approach on two different datasets and classifiers trained on them. Specifically, we use a binary classifier trained for sentiment classification on the IMDb dataset (Maas et al., 2011) and a multi-class short-document classifier trained on Newsgroups (Lang, 1995). We provide details and statistics for each dataset in Appendix A.

Using these classifiers, we apply our methodology on the three counterfactual editors described in §4.2 and examine the metrics described in §3 by generating edits, testing the classifiers on them and feeding back the edited text to the editor for $n = 10$ steps. For each editor we apply our methodology as follows: for a given input text x, at step n we select from the pool of generated edited texts the one with the minimum minimality that alters the prediction, if such an output exists, otherwise we select the minimum minimality output. We repeat this process until $n = 10$ and discuss the behaviour of each editor across metrics in §5 and §6.

We also study the impact of test-set size on the observed differences between results and the statistical significance of findings. We present detailed results in Appendix E. We found that for a test set size greater than 200 texts results converge on both datasets and we obtain statistically significant differences. Based on these findings we randomly sample 500 texts from the IMDb dataset for our experiments, to reduce the computational load. Since NewsGroups is smaller, we use the full dataset.

4.1 Evaluation

The metrics that we use with our methodology are:
Minimality: The word-level Levenshtein distance between original and edited text.
inc@n: Inconsistency of word-level Levenshtein distance as per equation 2.
Flip rate: The ratio $\frac{\mathsf{n_{flipped}}}{\mathsf{n_{all}}}$, where $\mathsf{n_{all}}$ is the size of the dataset, and $\mathsf{n_{flipped}}$ are the samples for which the prediction changes after applying the editor.
ppl-base: Language model perplexity of GPT-2, a large, general-domain language model (Radford et al., 2019), as per equation 3.
ppl- fine: Language model perplexity of GPT-2, fine-tuned on IMDB [2] and on Newsgroups [3]. Used to examine how "unexpected" the edited text is with respect to each dataset.

We also compute the above metrics after n steps of feedback, with the exception of inc@n, which uses the feedback steps by definition.

4.2 Editors

We experimented with three editors with different characteristics. Brief descriptions of each editor and the main differences between them are presented below, and more details in Appendix B.

Polyjuice Polyjuice is a general-purpose counterfactual generator that produces perturbations based on predefined control types. This editor employs a GPT-2 model that was fine-tuned on various datasets of paired sentences (input-counterfactual), including the IMDb dataset. Polyjuice does not use the classifier predictions during the generation of the counterfactual texts but rather focuses on the variety of the edits based on a learned set of control codes such as "negation" or "delete".

MiCE MiCE is a two-step approach to generating counterfactual edits. It uses a T5 deep neural network to fill the blanks on a text, which is fine-tuned in the first step to better fit the dataset distribution. In the second step, the input text is masked either randomly or by using predictors attention in a white box manner, and the fine-tuned model fills these blanks. This step aims to learn the minimum edits that will alter the classifier's prediction. In our experiments, we employed MiCE in a white box

[2] https://huggingface.co/lvwerra/gpt2-imdb
[3] https://huggingface.co/QianWeiTech/GPT2-News

manner, meaning that the fine-tuning is done by using the predictor's outputs (and not the ground-truth label), and for selecting the masks' locations, the classifier's attention is used in order to compare its result with Polyjuice which also utilised a deep neural network for the generation, but uses the predictor in a black box manner.

TextFooler TextFooler is a method for generating adversarial examples for a black-box classifier. The generation process differs from other editors since it does not employ a deep neural network, such as GPT2 or T5, to construct the produced counterfactual. Instead consists of finding the words that are influential to the predicted class and replacing them in a more deterministic way. The influence of each word is calculated by measuring how its removal alters the predictor's output (Liang et al., 2017). The alternatives that can replace a word are predefined to be the closest match in the embedding space and are thus independent of the rest of the sentence and the classifier. Hence, TextFooler chooses single word replacements that are synonyms with the removed word, with the added constraint of having the same part-of-speech.

5 Interpreting the inc@n metric

In Table 1 we show the results of the proposed inc@n metric. It is important to mention that there is an intuitive interpretation of the inc@n metric. Since we use Levenstein distance in our experiments, inconsistency corresponds to the mean number of tokens that the editor is altering on top of those that were needed to produce a valid counterfactual. The reasons behind inconsistency could vary depending on the mechanism of selecting important parts of the source text, the generation procedure, or the algorithm for locating the best edits.

We can observe significant differences for inc@n between the editors, reflecting the differences in the underlying approach. TextFooler is the most consistent editor, with low inc@n values that imply very rare increases in minimality between steps. This shows that the controlled nature of TextFooler's approach for selecting replacements is beneficial to the generation of consistent explanations. MiCE and Polyjuice on the other hand are less consistent, which could be attributed to the use of large language models in the generation process, which are more sensitive to small perturbations that can alter their output (Jin et al., 2020).

Table 1: Inconsistency (inc@n) computed on the IMDb and Newsgroups datasets.

	MiCE	Polyjuice	TextFooler
		IMDb	
inc@1 ↓	0.86	6.21	0.01
inc@2 ↓	5.95	4.65	0.33
inc@3 ↓	4.65	3.98	0.36
inc@5 ↓	4.87	2.9	0.47
inc@9 ↓	4.73	2.22	0.49
		Newsgroups	
inc@1 ↓	11.11	0.99	0.04
inc@2 ↓	7.97	1.29	0.55
inc@3 ↓	7.89	1.35	0.46
inc@5 ↓	6.92	1.3	0.49
inc@9 ↓	6.11	1.21	0.46

The comparatively high value of inconsistency for Polyjuice at the first steps for IMDb, can be explained by the fact that it has to "guess" the locations it should change in the text without access to the predictor. Especially for longer inputs, the search space of Polyjuice is exponentially larger. This forces it to make more aggressive edits to achieve the same result, often deleting a large portion of the source text and seems to contribute to Polyjuice's reported robustness issue (Madsen et al., 2022). For example, Polyjuice erased over 70% of the original text for 83% of the first two steps of edits on the IMDb dataset (candidates with lower minimality may be produced but failed to change the class and were rejected). An extensive analysis of this tendency across all editors and datasets is presented in the Appendix D. This "extreme erasure" pattern disappeared in the next steps, where the input length was significantly smaller (for instance original texts have on average 204 tokens, while the edited ones produced by Polyjuice have 29). On the other hand, for shorter texts, for which the search space is smaller, Polyjuice is more consistent, without radical changes on the original input. We hence attribute the differences between the first and later steps of the $inc@n$ metric to this tendency of Polyjuice to reduce the length of long texts. After the first step since the length of the input texts have already been significantly reduced, its behaviour is more consistent. This pattern is invariant between the two datasets but not visible on the first steps of Newsgroups results since it contains texts with 43% fewer tokens than IMDb on average.

To better understand what these values of inc@n represent, in Figures 3a, 3b we show box-plots of minimality and inc@n after each step of feedback for the IMDb dataset (box-plots for Newsgroups appear in Appendix C). While the tendency is for minimality to decrease, implying that the editors tend to perform fewer edits after each feedback step, there are cases where it increases, implying inconsistency.

In Figure 3b, what stands out is the higher value of inc@n for MiCE when n is even. Since even steps correspond to the attempt to move from the original class to a different one, higher inc@n indicates that it is easier to transition to the original class than from it. This could be attributed to remnants of the original input text pushing the classifier towards the original prediction, and thus requiring fewer edits. For instance in Figure 2 we can see an example of an edit produced by MiCE, where clearly parts of the original text indicating positive sentiment (marked in bold) were left unchanged. On Newsgroups, MiCE is significantly more inconsistent especially in the first steps, probably due to its requirement to select a specific target class in contrast to the other editors (see also Appendix C).

These observations highlight the need for feedforward evaluations of such systems since the minimality@1 reveals only a limited, dataset-dependent aspect of editors' capabilities and performance. Furthermore, they show the effectiveness of additional feedback steps to more accurately quantify the difference between the samples produced by an editor, and obtain a proxy for a global minimum. In Table 1 we see that from inc@3 on the obtained inc@n values start to converge for both datasets.

It is important to mention that the inconsistency of minimality captures different attributes of the editor than the minimality itself. High minimality means that the editor made more edits in order to alter the label of the input text. This may be due to either a weakness of the editor or the input itself requiring more edits to change class. To exclude the latter hypothesis, it is necessary to find counterfactual examples with lower minimality than the one produced, to confirm that there are better states that the editor could not explore. However, these states must meet the exact same conditions that the editor takes into account. The three editors analysed in this study have a secondary objective of producing realistic counterexamples; hence, for instance, the addition of random characters in the middle of the text is not a desirable goal state, despite the fact that it may result to a label flip with lower minimality. Along the same lines, TextFooler aims to replace every word with a synonym; therefore, a counterexample that replaces a word with an antonym (e.g., 'love' with 'hate') is not an acceptable goal state for it. To our knowledge, there have been no efficient or impartial methods for finding counterexamples with a lower value of a specified metric (such as minimality) that also meet exactly the same requirements as the editor. The proposed methodology comes to fill this gap, and the inconsistency metric can quantify the weaknesses of the editor in terms of the studied metric, in this case, minimality. In short, a positive inconsistency proves that there are goal states with a lower value of the corresponding metric that the editor should, but did not explore.

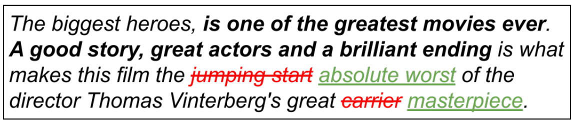

*The biggest heroes, **is one of the greatest movies ever**. **A good story, great actors and a brilliant ending** is what makes this film the ~~jumping start~~ absolute worst of the director Thomas Vinterberg's great ~~carrier~~ masterpiece.*

Figure 2: MiCE example of an IMDb dataset sample.

Table 2: Flip-rate after feeding the original text to the editor once (@1), and after 9 steps of feedback (@9) for the IMDb and Newsgroups dataset.

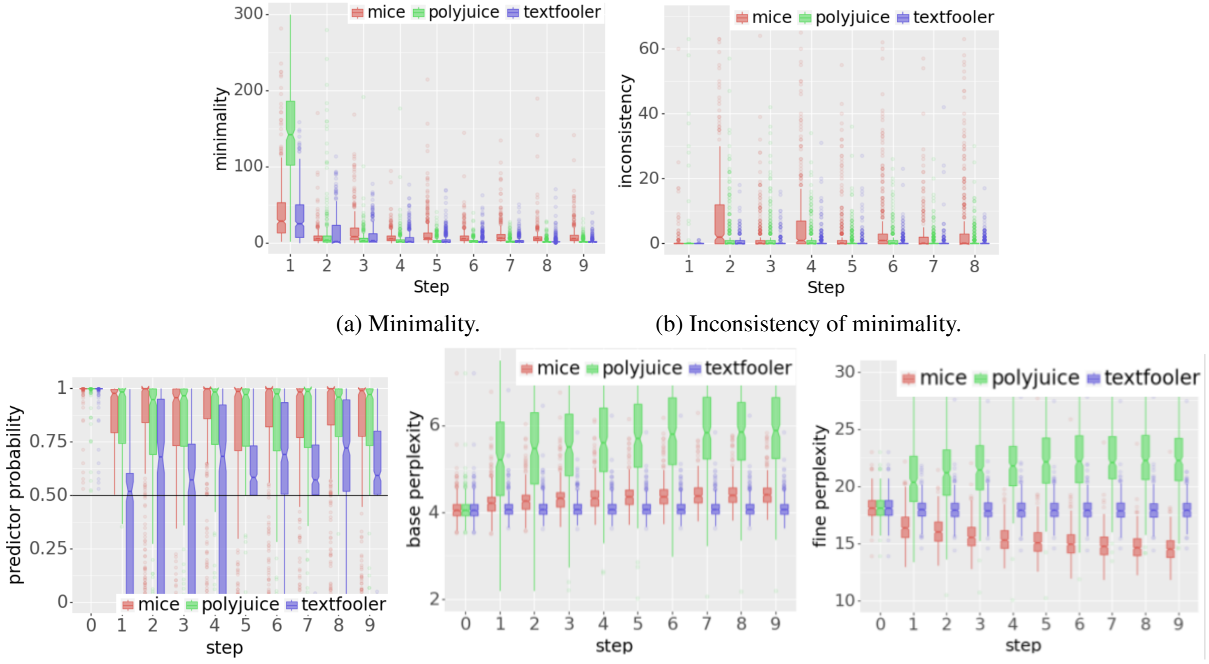
	MiCE	Polyjuice	TextFooler
		IMDb	
Flip Rate@1 ↑	**1.000**	0.8747	0.6195
Flip Rate @9 ↑	0.8561	**0.9675**	0.7865
		Newsgroups	
Flip Rate@1 ↑	**0.87**	0.77	0.79
Flip Rate @9 ↑	0.836	**0.968**	0.89

6 Additional insights from counterfactuals of counterfactuals

Besides measuring minimality and inc@n, we also investigated how the feedback approach can give us additional insights for the other two desiderata for editors, flip-rate and fluency.

6.1 Flip Rate

In Table 2 we show flip-rate measured after applying the feedback methodology. At the first step MiCE has a perfect flip rate; if analysed in solitude this observation might lead to the erroneous conclusion that the model can always alter the class of any text. However, this is a test-set-dependent result and does not apply in general since the flip rate reduces significantly in the following steps. Hence, there are instances closer to its distribution (Section 6.1) in which MiCE could not alter the predicted class. Conversely, the flip-rate of Polyjuice and TextFooler increases for later feedback steps.



(a) Minimality.

(b) Inconsistency of minimality.

(c) Probability of the target class.

(d) PPL of base GPT-2.

(e) PPL of fine GPT-2.

Figure 3: Minimality, inc@n, and predictor probability, base-ppl and fine-ppl, after each step of feedback and for each editor on the IMDb dataset.

To investigate this, in Figure 3c we show the prediction probabilities for the target class after applying each editor and after each step of feedback, where a sample is flipped if it has a target prediction probability greater than 0.5. Comparing this figure with the corresponding figure of the inconsistency of minimality (Figure 3b), we can observe the same patterns, such as differences in even and odd steps for MiCE. Thus this figure seems to corroborate the difficulty of the editor to return to the original class from the counterfactual.

6.2 Fluency

We use two metrics to approximate the fluency of the generated texts, shown in Table 3. For the ppl-base indicator, TextFooler has the lowest value, indicating fluent text, while the value does not change after several feedback steps, which further supports the editor's consistency. In contrast, MiCE's fluency slightly deteriorates after feedback, which coincides with the editor's inconsistency (compared to TextFooler). Furthermore, in Figures 3d, 3e we show the evolution of the two fluency related metrics for each feedback step. TextFooler's fluency is relatively stable across both metrics, on par with its low inconsistency. On the other hand, Polyjuice appears to deteriorate in fluency, as both the base-PPL and fine-PPL indicators have an increasing trend. The most striking difference in the perplexity patterns lies between MiCE and Polyjuice, where for the base model perplexity is consistently increasing for both editors, while for the fine-tuned model, MiCE's perplexity decreases and Polyjuice's continues to increase. We can thus deduce that while both editors produce changes that have a negative impact in the overall fluency, in the case of MiCE which is trained exclusively on IMDb data, the edited texts are closer to the IMDb distribution, hinting at an "overfitting" behaviour of sorts. Instead, Polyjuice takes advantage of different datasets during training and produces more diverse edits.

7 Conclusion

In this work we introduced a methodology for analysing different aspects of counterfactual editors and obtaining an approximate ground truth by iteratively feeding back their output. Combined with evaluation metrics from related literature, our proposed approach provides new ways to understand the counterfactual editors' behaviour and performance based on their intended use case, and thus help develop better editors. We proposed inc@n, a metric to measure the consistency of editors, and we showed how the proposed approach can help diagnose and analyse a diverse set of existing editors

Table 3: Metrics for measuring fluency computed for three counterfactual editors, of the IMDb and Newsgroups datasets, after feeding the original text to the editor once (@1), and after 8 additional steps of feedback (@9)

	MiCE	Polyjuice	TextFooler
		IMDb	
ppl-base@1 ↓	4.2546	7.4525	**4.1178**
ppl-base@9 ↓	4.4512	7.3825	**4.1161**
ppl-imdb@1 ↓	**16.5315**	33.4798	18.0662
ppl-imdb@9 ↓	**14.6069**	27.8074	17.9917
		Newsgroups	
ppl-base@1 ↓	5.164	8.926	**4.801**
ppl-base@9 ↓	5.36	7.878	**4.776**
ppl-newsgroup@1 ↓	4.27	6.67	**3.99**
ppl-newsgroup@9 ↓	4.4	5.90	**3.98**

and gain new insights on their behaviour, such as those made apparent by observing the discrepancies of odd and even steps of feedback.

Our findings allow for a more interpretable evaluation of editors that goes beyond mere comparisons between them. The results motivate further research in this direction including experiments on additional evaluation metrics, editors and tasks. Apart from expanding the scope of our experiments, we intend to look into using the feedback information to automatically address the weaknesses and inconsistencies of editors during fine-tuning, to obtain more robust and interpretable counterfactual edits. Along these lines, we also plan to investigate the benefit of integrating the feedback rationale into the training of counterfactual generation algorithms; for example a back-translation inspired objective could help alleviate the problematic behaviour and boost performance.

Limitations

This work focused on experiments on English datasets and did not explore other languages. While we expect that our assumptions hold across languages and the proposed methods and metrics can be applied without any further modifications to other languages this has not been explicitly verified. Additionally, we ensured to experiment with counterfactual editors that are representative of the main counterfactual editing methodologies, however we did not exhaustively cover all publicly available editors as our main goal was to demonstrate that our proposed method is widely applicable rather than to exhaustively compare editors.

Acknowledgements

This work was supported by the European Research Council (ERC StG DeepSPIN 758969) and by the Fundação para a Ciência e Tecnologia through contract UIDB/50008/2020.

References

Udit Arora, William Huang, and He He. 2021. Types of out-of-distribution texts and how to detect them. *arXiv preprint arXiv:2109.06827.*

Esma Balkir, Svetlana Kiritchenko, Isar Nejadgholi, and Kathleen C Fraser. 2022. Challenges in applying explainability methods to improve the fairness of nlp models. *arXiv preprint arXiv:2206.03945.*

Francesco Bodria, Fosca Giannotti, Riccardo Guidotti, Francesca Naretto, Dino Pedreschi, and Salvatore Rinzivillo. 2021. Benchmarking and survey of explanation methods for black box models. *arXiv preprint arXiv:2102.13076.*

Nitay Calderon, Eyal Ben-David, Amir Feder, and Roi Reichart. 2022. Docogen: Domain counterfactual generation for low resource domain adaptation. *arXiv preprint arXiv:2202.12350.*

Saneem Chemmengath, Amar Prakash Azad, Ronny Luss, and Amit Dhurandhar. 2021. Let the cat out of the bag: Contrastive attributed explanations for text. *arXiv preprint arXiv:2109.07983.*

Eoin Delaney, Derek Greene, and Mark T Keane. 2021. Uncertainty estimation and out-of-distribution detection for counterfactual explanations: Pitfalls and solutions. *arXiv preprint arXiv:2107.09734.*

Finale Doshi-Velez and Been Kim. 2017. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608.*

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. *arXiv preprint arXiv:1808.09381.*

Xiaoli Fern and Quintin Pope. 2021. Text counterfactuals via latent optimization and shapley-guided search. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5578–5593.

Bryce Goodman and Seth Flaxman. 2017. European union regulations on algorithmic decision-making and a “right to explanation”. *AI magazine*, 38(3):50–57.

Riccardo Guidotti. 2022. Counterfactual explanations and how to find them: literature review and benchmarking. *Data Mining and Knowledge Discovery*, pages 1–55.

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. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8018–8025.

Vineet John, Lili Mou, Hareesh Bahuleyan, and Olga Vechtomova. 2018. Disentangled representation learning for non-parallel text style transfer. *arXiv preprint arXiv:1808.04339*.

Mark T Keane, Eoin M Kenny, Eoin Delaney, and Barry Smyth. 2021. If only we had better counterfactual explanations: Five key deficits to rectify in the evaluation of counterfactual xai techniques. *arXiv preprint arXiv:2103.01035*.

Ken Lang. 1995. Newsweeder: Learning to filter netnews. In *Machine Learning Proceedings 1995*, pages 331–339. Elsevier.

Vladimir I Levenshtein et al. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union.

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

Bin Liang, Hongcheng Li, Miaoqiang Su, Pan Bian, Xirong Li, and Wenchang Shi. 2017. Deep text classification can be fooled. *arXiv preprint arXiv:1704.08006*.

Zachary C Lipton. 2018. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3):31–57.

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

Nishtha Madaan, Inkit Padhi, Naveen Panwar, and Diptikalyan Saha. 2021. Generate your counterfactuals: Towards controlled counterfactual generation for text. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 13516–13524.

Andreas Madsen, Siva Reddy, and Sarath Chandar. 2022. Post-hoc interpretability for neural nlp: A survey. *ACM Computing Surveys*, 55(8):1–42.

John X Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp. *arXiv preprint arXiv:2005.05909*.

Ramaravind K Mothilal, Amit Sharma, and Chenhao Tan. 2020. Explaining machine learning classifiers through diverse counterfactual explanations. In *Proceedings of the 2020 conference on fairness, accountability, and transparency*, pages 607–617.

Danish Pruthi, Rachit Bansal, Bhuwan Dhingra, Livio Baldini Soares, Michael Collins, Zachary C Lipton, Graham Neubig, and William W Cohen. 2022. Evaluating explanations: How much do explanations from the teacher aid students? *Transactions of the Association for Computational Linguistics*, 10:359–375.

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

Kaivalya Rawal, Ece Kamar, and Himabindu Lakkaraju. 2020. Can i still trust you?: Understanding the impact of distribution shifts on algorithmic recourses. *arXiv e-prints*, pages arXiv–2012.

Marcel Robeer, Floris Bex, and Ad Feelders. 2021. Generating realistic natural language counterfactuals. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3611–3625.

Alexis Ross, Ana Marasović, and Matthew E Peters. 2020. Explaining nlp models via minimal contrastive editing (mice). *arXiv preprint arXiv:2012.13985*.

Alexis Ross, Tongshuang Wu, Hao Peng, Matthew E Peters, and Matt Gardner. 2021. Tailor: Generating and perturbing text with semantic controls. *arXiv preprint arXiv:2107.07150*.

Cynthia Rudin. 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215.

Dylan Slack, Anna Hilgard, Himabindu Lakkaraju, and Sameer Singh. 2021. Counterfactual explanations can be manipulated. In *Advances in Neural Information Processing Systems*, volume 34, pages 62–75. Curran Associates, Inc.

Ilia Stepin, José Maria Alonso, Alejandro Catalá, and Martin Pereira-Fariña. 2021. A survey of contrastive and counterfactual explanation generation methods for explainable artificial intelligence. *IEEE Access*, 9:11974–12001.

Marcos V Treviso and André FT Martins. 2020. The explanation game: Towards prediction explainability through sparse communication. *arXiv preprint arXiv:2004.13876*.

Sergiy Tyupa. 2011. A theoretical framework for back-translation as a quality assessment tool. *New Voices in Translation Studies*, 7(1):35–46.

Jake Vasilakes, Chrysoula Zerva, Makoto Miwa, and Sophia Ananiadou. 2022. Learning disentangled representations of negation and uncertainty. *arXiv preprint arXiv:2204.00511*.

Sahil Verma, John P. Dickerson, and Keegan Hines. 2020. Counterfactual explanations for machine learning: A review. *CoRR*, abs/2010.10596.

Sandra Wachter, Brent D. Mittelstadt, and Chris Russell. 2017. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *CoRR*, abs/1711.00399.

Alex Wang and Kyunghyun Cho. 2019. Bert has a mouth, and it must speak: Bert as a markov random field language model. In *Proceedings of the Workshop on Methods for Optimizing and Evaluating Neural Language Generation*, pages 30–36.

Zhao Wang and Aron Culotta. 2021. Robustness to spurious correlations in text classification via automatically generated counterfactuals. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 14024–14031.

Tongshuang Wu, Marco Tulio Ribeiro, Jeffrey Heer, and Daniel S Weld. 2021. Polyjuice: Generating counterfactuals for explaining, evaluating, and improving models. *arXiv preprint arXiv:2101.00288*.

A Dataset Analytics

A.1 IMDb

The original IMDb dataset consists of 50K movie reviews split evenly between positive and negative ones (binary classification). We randomly sample 500 documents from the dataset to generate a test-set for our experiments.

In the sampled test-set, the mean number of tokens and characters of the selected comments are 204 and 1000, with a standard deviation of 112 and 562, respectively. In addition, 52% of these comments are classified as "positive" while 48% as "negative". The mean number of characters and tokens for inputs that are classified with "positive" sentiment is 990 and 530, with a standard deviation of 204, and 108, respectively. The distribution for texts that are classified with "negative" sentiment is similar, where the mean number of characters and tokens is 1006 ± 589 and 204 ± 115, respectively.

A.2 Newsgroups

The original Newsgroups dataset consists of 20K short documents split evenly between 20 newsgroup classes, representing the document topic. We use the test-set partition which consists of 7K documents for our experiments, as it is provided from scikit-learn library [4], since the train set has already used for fine-tuning some of the editors. The mean number of characters and tokens for this dataset is 603 and 207, with a standard deviation of 495 and 103, respectively.

The list of the 20 classes present in the dataset is: [comp.graphics, comp.os.ms-windows.misc, comp.sys.ibm.pc.hardware, comp.sys.mac.hardware, comp.windows.x, rec.autos, rec.motorcycles, rec.sport.baseball, rec.sport.hockey sci.crypt, sci.electronics, sci.med, sci.space, misc.forsale, talk.politics.misc, talk.politics.guns, talk.politics.mideast, talk.religion.misc, alt.atheism, soc.religion.christian]

B Experimental Setup

For both of our experiments, we used the predictors that are used in MiCE (since MiCE requires white box access to the predictor, and we wanted to intervene as little as possible in editors' code). These predictors were built based on ROBERTA-LARGE and were fixed during the evaluation. The predictors' accuracy is the same as stated in the proposed paper, 95.9% for IMDb and 85.3% for the Newsgroups.

We compare three counterfactual editors (MiCE, Polyjuice, and TextFooler) using the same classifier by making changes, testing the classifier on them, and feeding back the modified text to the editor ten times. Editors create numerous altered versions at each feedback stage and for each input text. We choose the output with the lowest minimality that changes the prediction (counterfactual goal), if such an output exists; otherwise, we choose the output with the lowest minimality. This design choice was made because there were cases in which an editor may not alter the prediction of the produced text by it made this transformation in the following steps.

MiCE For the editor of mice, we used the pretrained T5 model that the authors provided[5] [6]. This model was fine-tuned on the same data as the predictor. For the generation procedure, we left the default arguments for each one of the datasets as the authors supplied on its page [7], where we also got the code for our experiments. The only addition that was made to this code is integrating our data as an input to generate counterfactuals at each step.

Polyjuice We use Polyjuice through this module[8]. For the generation procedure, we searched in all the control codes ('resemantic', 'restructure', 'negation', 'insert', 'lexical', 'shuffle', 'quantifier', 'delete'), and we produce as many perturbations as it is possible for each instance. We did this by setting the $num\_perturbations = 1000$. In none of our experiments, Polyjuice had returned this plethora of results.

TextFooler We utilised TextFooler through TextAttack module[9]. For the sake of fair comparison, we chose the same parameters as those presented in the paper by the authors. The constraints concern the disallowing of the modification of stopwords and already modified words. Also, the threshold of the word embedding distance that two words are considered synonyms is defined as 0.5, and we force the replacements to be based on pos tagging.

[4] https://scikit-learn.org/0.19/datasets/twenty_newsgroups.html

[5] https://storage.googleapis.com/allennlp-public-models/mice-imdb-predictor.tar.gz

[6] https://storage.googleapis.com/allennlp-public-models/mice-newsgroups-editor.pth

[7] https://github.com/allenai/mice

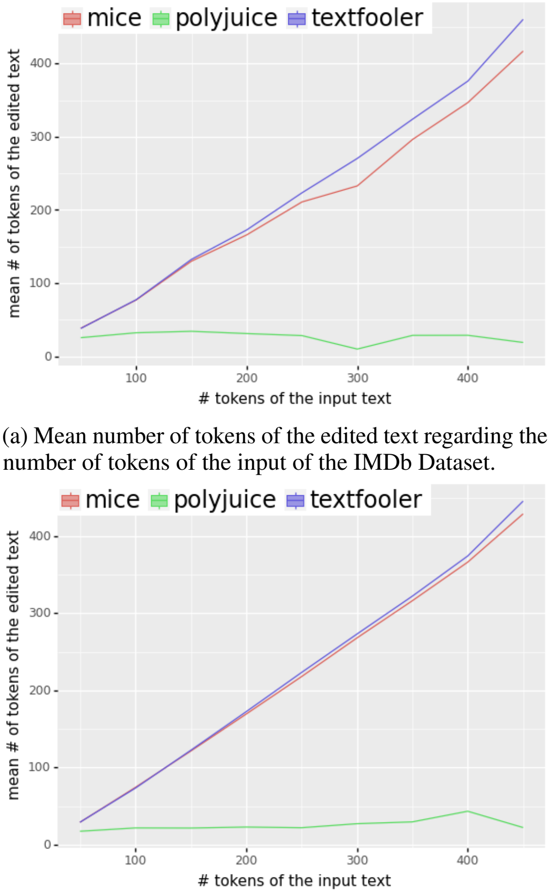
[8] https://github.com/tongshuangwu/polyjuice

[9] https://textattack.readthedocs.io/en/latest/

C Newsgroups further analysis

Figure 6 depicts the inconsistency of minimality, the perplexity of base GPT-2, and the perplexity of fine GPT-2 for each editor on the Newsgroups dataset. As this task is not binary, there is no pattern between odd and even steps that we observed on the IMDb dataset, but there is consistent behaviour. More specifically, since the editor does not have to return to the original class, but to any other class at each feedback step, the difficulty of flipping labels is similar between even and odd steps. In fact, if we isolate the cases where the editor returns to the original class, the behaviours observed on IMDb still hold. Furthermore, the fact that Newsgroups is a multi-class dataset, seems to make MiCE struggle more than the other editors (see also Table 1) due to the fact that MiCE requires a target class to be specified, and edits the text accordingly to flip to that class, while Polyjuice and TextFooler allow the option to perform edits just to change the class of the input, to *any* other class. We address this requirement by defining as a target class for each step, the second class of the prediction, which is also the default methodology that the editor's creators follow in their study. So the task that MiCE performs is harder than the others (editing a text to be classified from class A to class B is at least as hard as editing the text to be classified from class A to any other class), and could lead to the observed higher inconsistency values for MiCE and the different behaviour compared to the IMDb dataset.

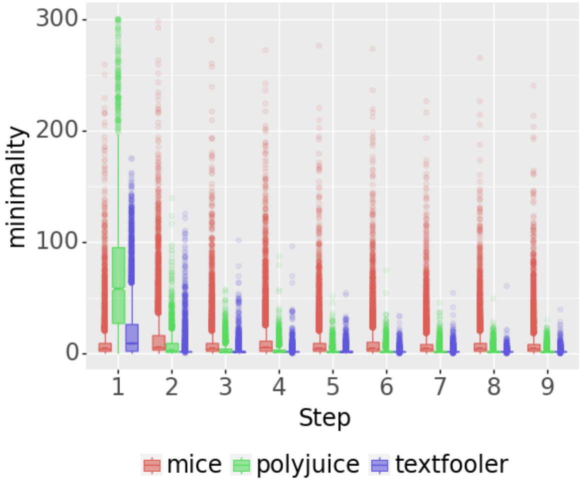
Based on the figures, we can conclude that the proposed method produces consistent results for the behaviour of each editor even with fewer steps. We can thus significantly reduce the computational cost of the method, as just two or three steps are enough for drawing reliable conclusions.



(a) Mean number of tokens of the edited text regarding the number of tokens of the input of the IMDb Dataset.

(b) Mean number of tokens of the edited text regarding the number of tokens of the input of the Newsgroups Dataset.

Figure 4: Mean number of tokens of the edited text regarding the number of tokens of the input.



(a) Minimality@n for the Newsgroups dataset.

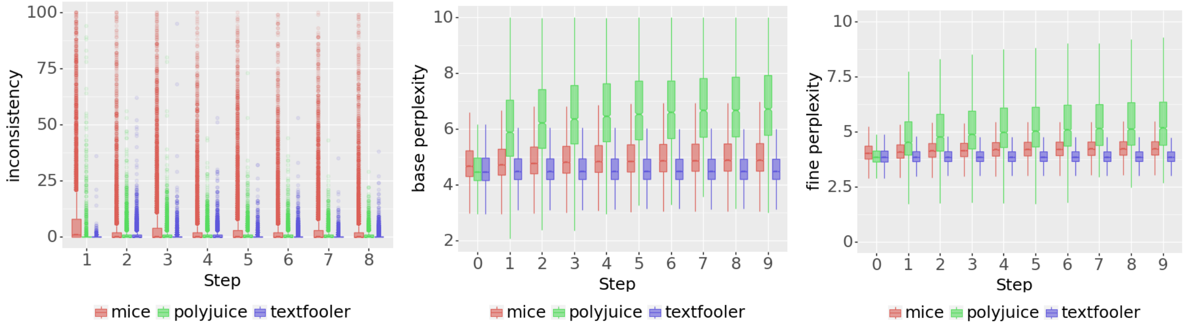
Figure 5: Minimality@n for the Newsgroups Dataset.

D Length of Counterfactual texts

In order to further investigate the behaviour of each editor regarding the number of tokens of the input text, we present Figure 4, which depicts the mean number of tokens of the edited texts relative to the number of input tokens. The output texts of MiCE and Textfooler are distributed equally with the input text for both of the studied datasets. However, Polyjuice produces text with a limited length. There are also cases where the produced text of Polyjuice is longer, but they are not representative. It is worth mentioning that this may be caused due by the inner mechanism (e.g. GPT-2) of this method but also due to the evaluation method. As Polyjuice is a method for counterfactual generation, for the evaluation procedure, we preferred a text that has a different label than the original text (achieve the counterfactual goal) instead of one closest one that is classified on the same class (Madsen et al., 2022). This, combined with the task-agnostic nature of Polyjuice, forced it to make more aggressive edits by pruning a significant portion of the original text, which is a constant behaviour along the datasets.

E Size of test set

In order to investigate the effect of the sample size on the results of the proposed metric, we conducted multiple t-tests for different sample sizes, feedback steps, and datasets. In particular, we selected four subsets of 10, 50, 100, and 200 samples, and we performed t-tests between the values of their inconsistencies of every pair of the editors in order to find out at which point their values are significantly different from each other. The p-values of these experiments are shown in Table 4 for the IMDb dataset and Table 5 for the Newsgroups. In these tables, the values that are consistently (for each feedback step) less than 0.05 are shown in bold. For every feedback step in the IMDb dataset, p is less than 0.05 for sample sizes greater than **100**, while for Newsgroups, the same holds true for sample sizes greater than **200**.



(a) Inconsistency of minimality.

(b) Perplexity of base GPT-2.

(c) Perplexity of fine GPT-2.

Figure 6: Inc@n, Perplexity of base GPT-2 and Perplexity of fine GPT-2 for the Newsgroups Dataset.

MiCE								
Sample Size: 10								
	1	2	3	4	5	6	7	8
Polyjuice	0.2812	0.588	0.3563	0.3219	0.1093	0.3376	0.3039	0.133
TextFooler	0.4788	0.4853	0.2538	0.2107	0.2014	0.6249	0.0695	0.1658
Sample Size: 50								
Polyjuice	0.0383	0.342	0.0266	0.0073	0.1714	0.0377	0.0852	0.1184
TextFooler	0.2805	1.232e-05	0.2646	0.004	0.03	0.0054	0.1028	0.0063
Sample Size: 100								
Polyjuice	**0.0252**	**0.0168**	**0.0001**	**0.0001**	**0.0048**	**0.0091**	**0.0081**	**0.0003**
TextFooler	**0.0495**	**6e-08**	**0.0104**	**0.0001**	**0.0016**	**0.0003**	**0.0032**	**0.0001**
Sample Size: 200								
Polyjuice	**0.0084**	**0.0036**	**1e-08**	**4.02e-05**	**2.72e-05**	**0.0012**	**0.0007**	**5.66e-06**
TextFooler	**0.0461**	**2.73e-14**	**0.0013**	**1.3e-10**	**0.0006**	**4.89e-07**	**2.2e-05**	**4.2e-08**
Sample Size: 500								
Polyjuice	**0.00043**	**0.0**	**0.036**	**0.0**	**0.0006**	**1e-08**	**0.001**	**0.0**
TextFooler	**0.0368**	**0.0**	**1.76e-06**	**0.0**	**1.86e-06**	**0.0**	**4.2e-05**	**0.0**

Polyjuice								
Sample Size: 10								
	1	2	3	4	5	6	7	8
mice	0.2812	0.588	0.3563	0.3219	0.1093	0.3376	0.3039	0.133
textfooler	0.2831	0.3649	0.3056	0.2198	0.073	0.3337	0.1573	0.047
Sample Size: 50								
mice	0.0383	0.342	0.0266	0.0073	0.1714	0.0377	0.0852	0.1184
textfooler	0.0378	0.0015	0.021	0.0011	0.11	0.021	0.0199	0.0261
Sample Size: 100								
mice	**0.0252**	**0.0168**	**0.0001**	**0.0001**	**0.0048**	**0.0091**	**0.0081**	**0.0003**
textfooler	**0.0246**	**4.733e-05**	**4.351e-05**	**9e-06**	**0.0026**	**0.0038**	**0.0003**	**4.258e-05**
Sample Size: 200								
mice	**0.0084**	**0.0036**	**1e-08**	**4.028e-05**	**2.723e-05**	**0.0012**	**0.0007**	**5.66e-06**
textfooler	**0.0082**	**0.0**	**0.0**	**6e-08**	**1.367e-05**	**0.0002**	**1.6e-07**	**1.3e-07**
Sample Size: 500								
mice	**0.0004**	**0.0**	**0.036**	**0.0**	**0.0007**	**1e-08**	**0.0011**	**0.0**
textfooler	**2.2e-05**	**0.0001**	**3.88e-05**	**0.0016**	**0.0058**	**0.0009**	**0.0192**	**0.0007**

TextFooler								
Sample Size: 10								
	1	2	3	4	5	6	7	8
polyjuice	0.2831	0.3649	0.3056	0.2198	0.073	0.3337	0.1573	0.047
mice	0.4788	0.4853	0.2538	0.2107	0.2014	0.6249	0.0695	0.1658
Sample Size: 50								
polyjuice	0.0378	0.0015	0.021	0.0011	0.11	0.021	0.0199	0.0261
mice	0.2805	1.232e-05	0.2646	0.004	0.03	0.0054	0.1028	0.0063
Sample Size: 100								
polyjuice	**0.0246**	**4.733e-05**	**4.351e-05**	**9e-06**	**0.0026**	**0.0038**	**0.0003**	**4.258e-05**
mice	**0.0495**	**6e-08**	**0.0104**	**0.0001**	**0.0016**	**0.0003**	**0.0032**	**0.0001**
Sample Size: 200								
polyjuice	**0.0082**	**0.0**	**0.0**	**6e-08**	**1.367e-05**	**0.0002**	**1.6e-07**	**1.3e-07**
mice	**0.0461**	**0.0**	**0.0013**	**0.0**	**0.0006**	**4.9e-07**	**2.225e-05**	**4e-08**
Sample Size: 500								
polyjuice	**2.28e-05**	**0.0001**	**3.882e-05**	**0.0016**	**0.0058**	**0.0009**	**0.0192**	**0.0007**
mice	**0.0369**	**0.0**	**1.76e-06**	**0.0**	**1.86e-06**	**0.0**	**4.251e-05**	**0.0**

Table 4: P-value of the inconsistency of different sample sizes of the IMDb dataset.

MiCE								
Sample Size: 10								
polyjuice	0.1311	0.0963	0.4378	0.0042	0.1069	0.2044	0.1384	0.1809
textfooler	0.0717	0.2283	0.0924	0.3264	0.165	0.2194	0.5411	0.0453
Sample Size: 50								
polyjuice	0.1311	0.0963	0.4378	0.0042	0.1069	0.2044	0.1384	0.1809
textfooler	0.0006	0.0064	0.0338	0.1265	0.008	0.1015	0.0995	0.0101
Sample Size: 100								
polyjuice	0.0033	2.64e-06	0.0177	1e-08	0.0017	2.878e-05	0.081	0.0049
textfooler	**1.29e-06**	**0.0004**	**0.0034**	**0.0001**	**0.0009**	**0.0104**	**0.0344**	**0.0007**
Sample Size: 200								
polyjuice	**0.0**	**0.0005**	**4.937e-05**	**0.0002**	**2.13e-06**	**0.0003**	**0.006**	**0.0041**
textfooler	**1.513e-05**	**0.0**	**2.8e-07**	**0.0**	**3.5e-07**	**0.0**	**0.0043**	**2e-08**
Sample Size: 500								
polyjuice	**0.0**	**2.32e-06**	**0.0**	**1.09e-06**	**0.0**	**5e-08**	**0.0**	**1e-08**
textfooler	**0.0**	**3.3e-07**	**0.0**	**3e-08**	**0.0**	**0.0**	**0.0**	**0.0**

Polyjuice								
Sample Size: 10								
	1	2	3	4	5	6	7	8
mice	0.3306	0.0995	0.1407	0.0043	0.7421	0.4028	0.1387	0.2846
textfooler	0.1311	0.0963	0.4378	0.0042	0.1069	0.2044	0.1384	0.1809
Sample Size: 50								
mice	0.9654	0.0065	0.6376	8e-08	0.7342	0.3373	0.1692	0.9168
textfooler	0.0033	2.64e-06	0.0177	1e-08	0.0017	2.878e-05	0.081	0.0049
Sample Size: 100								
mice	0.3659	1e-08	0.719	0.0	0.5959	0.173	0.1055	0.7947
textfooler	0.0004	0.0	0.0002	0.0	0.0001	1e-08	0.0358	4.421e-05
Sample Size: 200								
mice	**0.0468**	**0.0**	**0.5025**	**0.0**	**0.8023**	**0.0176**	**0.0384**	**0.2746**
textfooler	**1.513e-05**	**0.0**	**2.8e-07**	**0.0**	**3.5e-07**	**0.0**	**0.0043**	**2e-08**
Sample Size: 500								
mice	**0.0**	**2.32e-06**	**0.0**	**1.09e-06**	**0.0**	**5e-08**	**0.0**	**1e-08**
textfooler	**0.0005**	**0.026**	**5.3e-07**	**0.0643**	**1.8e-07**	**0.0037**	**0.0**	**0.0019**

TextFooler								
Sample Size: 10								
	1	2	3	4	5	6	7	8
polyjuice	0.1311	0.0963	0.4378	0.0042	0.1069	0.2044	0.1384	0.1809
mice	0.0717	0.2283	0.0924	0.3264	0.165	0.2194	0.5411	0.0453
Sample Size: 50								
polyjuice	0.0033	2.64e-06	0.0177	1e-08	0.0017	2.878e-05	0.081	0.0049
mice	0.0006	0.0064	0.0338	0.1265	0.008	0.1015	0.0995	0.0101
Sample Size: 100								
polyjuice	**0.0004**	**0.0**	**0.0002**	**0.0**	**0.0001**	**1e-08**	**0.0358**	**4.421e-05**
mice	**1.29e-06**	**0.0004**	**0.0034**	**0.0001**	**0.0009**	**0.0104**	**0.0344**	**0.0007**
Sample Size: 200								
polyjuice	**1.513e-05**	**0.0**	**2.8e-07**	**0.0**	**3.5e-07**	**0.0**	**0.0043**	**2e-08**
mice	**0.0**	**0.0005**	**4.937e-05**	**0.0002**	**2.13e-06**	**0.0003**	**0.006**	**0.0041**
Sample Size: 500								
polyjuice	**0.0005**	**0.0213**	**5.3e-07**	**0.0643**	**1.8e-07**	**0.0037**	**0.0**	**0.0019**
mice	**0.0**	**3.3e-07**	**0.0**	**3e-08**	**0.0**	**0.0**	**0.0**	**0.0**

Table 5: P-value of the inconsistency of different sample sizes of the Newsgroup dataset.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Limitations

□ A2. Did you discuss any potential risks of your work?
Not applicable. Left blank.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Abstract and Introduction (1)

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Experimental Setup (4), Interpreting the inc@n metric (5), Additional insights from counterfactuals of counterfactuals (6), and supplementary material

☑ B1. Did you cite the creators of artifacts you used?
Experimental Setup (4), Interpreting the inc@n metric (5), Additional insights from counterfactuals of counterfactuals (6)

☒ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
We provide references and links where the reader can find the license or terms. We used everything according to the license and terms.

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Experimental Setup (4), Interpreting the inc@n metric (5), Additional insights from counterfactuals of counterfactuals (6)

☒ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
We used publicly available datasets and we do not display any sensitive information in the paper.

□ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Not applicable. Left blank.

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Please see section 4, and appendix

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

C ☑ Did you run computational experiments?

Sections 5 and 6

☒ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
We provide information regarding the models used and details/references for reproducibility purposes, but we did not manage to calculate the computational budget, but since we only used publicly available models, the computational budget of their training/inference can be retrieved from the respective references.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Section 4

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Sections 5 and 6

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Appendix

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

□ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

□ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

□ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

□ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

□ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.