Explaining a Black Box without another Black Box

Anonymous ACL submission

Abstract

Although Counterfactual Explanation Methods (CEMs) are popular approaches to explain ML black-box classifiers, they are less widespread in NLP. A counterfactual explanation encodes the smallest changes required in a target document to modify a classifier's output. Most CEMs find those explanations by iteratively perturbing the target document until it is classified differently by the black box. We identify two main families of CEMs in the literature, namely, (a) *transparent* methods that perturb the target by adding, removing, or replacing words, and (b) *opaque* approaches that project the target document onto a latent, non-interpretable space where the perturbation is carried out subsequently. This article offers a comparative study of the performance of these two families of methods on three classical NLP tasks. Our empirical evidence shows that opaque CEMs can be an overkill for downstream applications such as fake news detection or sentiment analysis since they add an additional level of complexity with no significant performance gain.

1 Introduction

The latest advances in ML have revolutionized many downstream NLP tasks such as fake news detection or sentiment analysis. However, the impressive accuracy achieved by modern algorithms comes at the expense of interpretability (Shen et al., 2020). This reliance on black-box models has, in turn, raised an increasing interest in ML explainability, the task of providing appropriate explanations for the answers of ML algorithms (Jacovi, 2023). Unless our ML model is a white box, explaining the outcomes of an ML agent requires to deploy an explanation layer that interprets the black box's inner-workings *a posteriori*.

There are several ways to explain the outcomes of an ML model. Among the different approaches, counterfactual explanations (CEMs) have gained notable popularity in the last 5 years (Miller, 2019). Consider a classifier for sentiment analysis applied to the book review "This is a really interesting book" – classified as positive. A counterfactual explanation is a counter-example that is similar to the original text, but that elicits an opposite outcome in the black box (Wachter et al., 2018). In this toy example, a counterfactual could be the phrase "This is a really boring book". Through this explanation, the CEM is conveying that the adjective "interesting" was the main reason this sentence was classified as positive, and changing the polarity of that adjective may change the classifier's response. CEMs in the literature operate by increasingly perturbing the target text until the classifier's answer changes. These methods lie in a spectrum spanning from fully transparent to fully opaque methods. On one side of the spectrum, *transparent* methods perturb the target text by adding, removing, or changing words and syntactic groups (Martens and Provost, 2014) in the original target text. On the opposite side, more recent *opaque* methods embed the target text in a latent space on which perturbations are carried out subsequently (Robeer et al., 2021). The embeddings are a compressed representation of the classifier's training data, which filters noise and retains the essential information for classification.

While one may think that latent-based CEMs outperform transparent methods, our empirical evidence suggests that, for some downstream NLP tasks such as spam and fake news detection, or sentiment analysis, learning a compressed representation can be an overkill. To illustrate this, and as a proof of concept, we developed two transparent CEMs that outperform opaque methods, mainly because, opaque methods often produce non-intuitive counterfactual explanations, i.e., counter-example texts that do not resemble at all the target. That does not only contradict the main point of counterfactual explanations but also raises the question of how much transparency we are actually gaining by explaining a black box with another black box.

Before elaborating on our experimental setup and findings in Section 3, we first survey the different CEMs in the literature in Section 2.

2 Related Works

CEMs convey the minimum changes in the input that would change a classifier's outcome. Social sciences (Miller, 2019) have shown that human explanations are contrastive and Wachter et al. (2018) have illustrated the utility of counterfactual instances in computational law. When it comes to NLP tasks, a good counterfactual explanation should be sparse (Pearl, 2009), i.e., look like the target instance, and be fluent (Morris et al., 2020), i.e., read like something someone would say.

CEMs have gained popularity in the last few years. As illustrated by the surveys, first by Bodria et al. (2021) and later by Guidotti (2022), around 50 additional CEMs appeared in a one-year time span. Despite this surge of interest in counterfactual explanations, their study for NLP applications remains underdeveloped (Ross et al., 2021). In the following, we elaborate on the existing CEMs for textual data along a spectrum that spans from transparent to opaque approaches.

Transparent Approaches. Given an ML classifier and a target text, transparent CEMs compute counterfactual explanations in a binary space. Each dimension represents the presence (1) or absence (0) of a word from a given vocabulary. Hence, to perturb a text, these methods toggle on and off 0's and 1's, until the classifier yields a different answer. This was first proposed by Martens and Provost (2014) who introduced SEDC, a method that removes the words that impact the classifier's prediction the most. Similarly, feature-attribution explanation methods such as LIME (Ribeiro et al., 2016) and SHAP (Lundberg and Lee, 2017) mask words randomly from the target text. More recently, Ross et al. (2021) developed MICE, a method that employs a Text-To-Text Transfer Transformer to fill masked sentences. Yang et al. (2020) presented PCIG, which generates plausible counterfactuals through edits of single words with lexicons manually selected from the economic domain.

Opaque Methods. We define opaque approaches as those perturbing the input text in a latent space in $\mathbb{R}^n$. Methods such as Decision Boundary (Hase and Bansal, 2020), XSPELLS (S. Punla et al., 2022) or CFGAN (Robeer et al., 2021) operate in three phases. First, they embed the target instance onto a latent space. Second, while the classifier's decision boundary is not traversed, these methods perturb the latent representation of the target phrase. Finally, a decoding stage generates sentences from the latent representation of the perturbed documents. Methods such as Polyjuice (Wu et al., 2021), GYC (Madaan et al., 2021) and Tailor (Ross et al., 2022) also perturb text documents in a latent space, but can be instructed to change particular linguistic aspects of the target text, such as locality or grammar tense. Such methods are not particularly designed to compute counterfactual explanations, but are rather conceived for multiple applications such as data augmentation.

Unlike pure word-based perturbation methods, latent representations are good at preserving *semantic closeness* for small perturbations. That said, these methods are not free of pitfalls. Besides the obvious loss in transparency and simplicity, existing latent-based approaches do not seem optimized for sparse counterfactual explanations, as our evaluation next suggests.

3 Comparative Study

In this section, we conduct a comparison between six representative CEMs of the spectrum depicted in Figure 1. These include the fully transparent method SEDC, the fully opaque approaches XSPELLS, CFGAN, and Polyjuice, and two methods of middle complexity. We excluded PCIG and MICE from the analysis because, PCIG relies on specific rules from economic domain, whereas MICE, albeit using a binary space for perturbation, still relies on transformers to find semantically pertinent word replacements. We therefore fill the middle ground with two methods that implement the same strategy but that rely on less heavy machinery, namely Growing Net and Growing Language.

3.1 Proposed Methods

Similarly to the three existing transparent methods, Growing Net and Growing Language search for counterfactuals by iteratively increasing the number of words modified in the target text. While PCIG integrates external knowledge to replace words, SEDC modifies the target by masking words, which often leads to non-sensical phrases. In contrast, our proposed methods replace words with other words that are syntactically and semantically meaningful. For instance in the phrase "This is an interesting book", the candidate substitutes for the word "interesting" should be adjectives that are pertinent qualifiers for books. Hence, for each word in the target

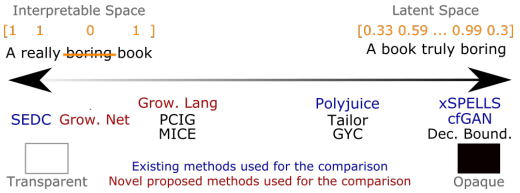

Figure 1: Spectrum for CEMs that goes from the most transparent methods on the left (e.g., SEDC) to the most opaque methods such as cfGAN, and XSPELLS, passing by our methods in red Growing Net and Growing Language. Transparent methods perturb documents in a binary space; opaque methods do it in a latent space.

sentence, our methods generate a set of potential replacement words with the same part-of-speech tagging of the original word. Once the syntactic group of the replacements is defined, we rely on external knowledge to restrict further our list of candidate words. Growing Net exploits the tree structure of WordNet (Fellbaum, 1998), a lexical database of semantic relations, to consider synonyms, antonyms, and hyponyms for each word in the original text. Conversely, Growing Language relies on the Spacy language model to provide candidate words whose similarity with the original word – as defined by Spacy – is smaller than a given threshold. We highlight that in our categorization (Figure 1) Growing Language is deemed less transparent than Growing Net because the former relies on a latent representation of words to filter candidates – even if the underlying exploration method is still conducted in the space of words. More details are given in Appendix A.

3.2 Results

We evaluated six CEMs from Figure 1 – representing each zone of the spectrum – in five rounds of experiments organized in two categories. First, we evaluate the quality of the generated counterfactual explanations based on the criteria of (i) minimality, (ii) fluency, and (iii) outlierness. Second, we evaluate the methods themselves based on (iv) flip change, and (v) runtime. Our evaluation is based on three popular NLP classification tasks: spam detection, polarity review, and fake news detection. For each task, we trained a neural network (more specifically a multi-layered perceptron) and a random forest classifier. For each dataset, black-box classifier, and CEM, we computed counterfactual explanations for 100 target texts. These are the inputs of our evaluation. Implementation details and the experimental datasets are provided in Appendix B. Code and datasets will be released on GitHub upon acceptance.

Counterfactual Quality. By definition (Guidotti, 2022), a good textual counterfactual explanation is an alternative phrase classified differently by the black box such that: (i) it incurs minimal changes, i.e., it looks like the target text; (ii) it is not an outlier, i.e., it looks like other phrases in the classifier's training/testing set; and (iii) it is linguistically plausible, i.e., it reads like something someone would actually write or say. Minimality can be quantified through the distance between the counterfactual and the target sentence. Similarly, outlierness can be operationalized as the distance of the counterfactual to the "manifold" defined by a set of instances (Delaunay et al., 2022). Following Ross et al. (2021, 2022), we measured linguistic plausibility through the perplexity of the counterfactual w.r.t. a pre-trained language model.

Figure 2 shows the results for the minimality of the computed counterfactuals. We measure minimality by embedding both the counterfactual and the original text with a sentence model derived from GPT-3 (Neelakantan et al., 2022). We then measure the cosine distance of the embeddings of the original and perturbed texts. We first observe that middle-ground methods, in particular Growing Net, perform quite well compared to latent approaches. We note that XSPELLS incurs the biggest changes in the original phrase. While XSPELLS makes small perturbations to the original instance, those perturbations happen in a latent space learned through a variational autoencoder (VAE). Nothing guarantees, however, that such small changes in the latent space translate into visually small changes when the resulting phrase is brought back to the original space.

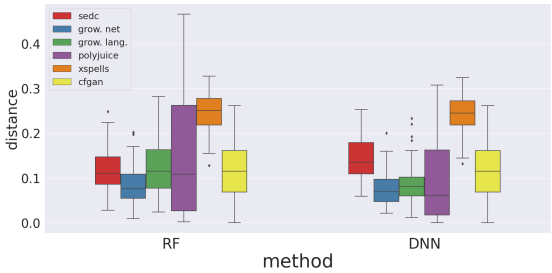

Figure 2: Minimality as the distance between the closest counterfactual and the target document.

Figure 3 depicts the outlierness for our studied CEMs. This is computed as the distance between the generated counterfactual and the closest test instance of our experimental datasets – hence the lower the better. We observe that XSPELLS

Method	SEDC		Grow. Net		Grow. Lang.		cfGAN		XSPELLS		Polyjuice	
Black box	DNN	RF	DNN	RF	DNN	RF	DNN	RF	DNN	RF	DNN	RF
spam	0.57	0.55	0.36	0.35	**0.65**	0.49	0.57	0.57	0.34	**0.61**	0.14	0.21
fake	0.90	0.98	0.79	0.59	0.86	0.86	**1**	**1**	**1**	**1**	0.38	0.28
polarity	0.95	0.92	**1**	0.85	0.93	0.93	0.65	0.65	**1**	**1**	0.46	0.46

Table 1: Average label flip per dataset and black box of the six counterfactual methods.

achieves the best performance for this criterion, again due to its reliance on VAEs, which are designed to compute a compressed representation of a dataset. At the other extreme, SEDC achieves the worst results since removing words randomly from the target can easily lead to outliers. Finally, we highlight that both Growing Net and Growing Language compare to CFGAN and Polyjuice in terms of outlierness, in spite of not relying extensively on heavy NN-based machinery.

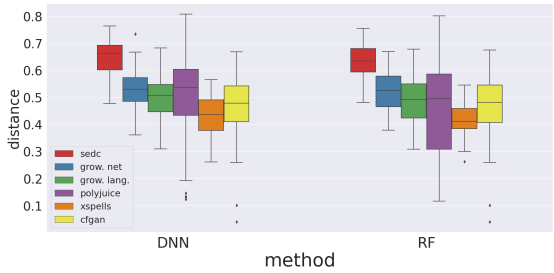

Figure 3: Outlierness as the distance between the counterfactuals and the closest instance in the test set.

Figure 4 represents the plausibility of the counterfactuals through the perplexity of a language model. This score is obtained by computing the mean squared error (MSE) loss of a pre-trained LM predicting the counterfactual's next token (hence the lower the better). We normalise the perplexity by the maximum perplexity observed in the whole set of counterfactuals. We thus observe that SEDC then Polyjuice generate texts with the lowest plausibility. The results for SEDC are not surprising, since this method masks words, resulting into nonsensical sentences. Conversely, both Growing Net and Language obtain a perplexity loss similar to XSPELLS and CFGAN.

Method Quality. We now compare our CEMs in terms of (iv) flip change, i.e., how often they manage to flip the class of the target, and (v) runtime.

Table 1 presents the label flip results. We note that except from the spam detection using neural networks, XSPELLS achieves the highest label flip. We also remark that the transparent methods perform better than CFGAN on the polarity review and

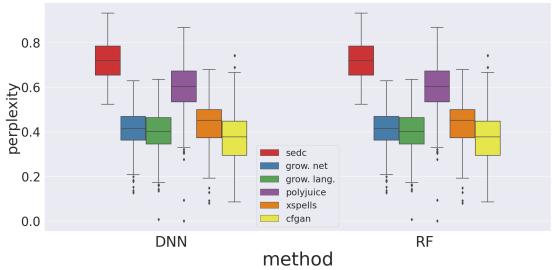

Figure 4: Perplexity as the MSE loss of a pre-trained language model on the generated counterfactuals.

spam detection tasks. In particular, Growing Net performs well for polarity analysis, which shows that replacement with antonyms is effective to find counterfactuals. We point out that both Growing Net and Growing Language can be parameterized to run a more exhaustive search – e.g., by lowering the similarity threshold or by adding further terms from the tree structure. This would increase label flip at the expense of longer runtimes. Finally, our runtime experiments show that methods such as Growing Net are fast enough to be used for the real-time generation of counterfactuals. This stands in sharp contrast to XSPELLS which is two orders of magnitude slower – due to its decoding stage. The results are provided in Appendixes C.

4 Conclusion

Our evaluation shows that when it comes to generating counterfactual explanations for downstream NLP tasks, complex methods based on NNs and latent spaces are not necessarily the most sensible alternatives. Our results show that simpler approaches based on a systematic and judicious replacement of words in the target sentence can still provide satisfactory results in all quality dimensions. We leave to the reader the interpretation of our results, e.g., as an argument for either developing self-explainable approaches, or better post-hoc explanation layers. That said, for scenarios when post-hoc explainability is the only alternative, we believe that explaining a black box using another black box may be perceived as paradoxical if it does not come with further benefits.

5 Limitations

We remind the reader that the evaluation was conducted on three well-studied downstream applications, namely polarity analysis, fake news detection, and spam detection. Our results might therefore not generalize to other NLP tasks in specialized domains or different languages. While this work puts transparent approaches in the spotlight, our results suggest that plausible counterfactual examples need external domain-adapted knowledge either in the form of language models or knowledge graphs. These may not always be available though. Finally, our evaluation was based on popular criteria and metrics for counterfactual explanations. Specialized applications may still take into account additional criteria such as diversity or actionability.

References

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

Julien Delaunay, Luis Galárraga, and Christine Largouët. 2022. When should we use linear explanations? In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management, Atlanta, GA, USA, October 17-21, 2022*, pages 355–364. ACM.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. Bradford Books.

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

Peter Hase and Mohit Bansal. 2020. Evaluating explainable AI: which algorithmic explanations help users predict model behavior? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 5540–5552. Association for Computational Linguistics.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural lanugage understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

Alon Jacovi. 2023. Trends in explainable AI (XAI) literature. *CoRR*, abs/2301.05433.

Scott M. Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 4765–4774.

Nishtha Madaan, Inkit Padhi, Naveen Panwar, and Diptikalyan Saha. 2021. Generate your counterfactuals: Towards controlled counterfactual generation for text. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 13516–13524. AAAI Press.

David Martens and Foster J. Provost. 2014. Explaining data-driven document classifications. *MIS Q.*, 38(1):73–99.

Tim Miller. 2019. Explanation in artificial intelligence: Insights from the social sciences. *Artif. Intell.*, 267:1–38.

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. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, EMNLP 2020 - Demos, Online, November 16-20, 2020*, pages 119–126. Association for Computational Linguistics.

Arvind Neelakantan, Tao Xu, Raul Puri, Alec Radford, Jesse Michael Han, Jerry Tworek, Qiming Yuan, Nikolas Tezak, Jong Wook Kim, Chris Hallacy, Johannes Heidecke, Pranav Shyam, Boris Power, Tyna Eloundou Nekoul, Girish Sastry, Gretchen Krueger, David Schnurr, Felipe Petroski Such, Kenny Hsu, Madeleine Thompson, Tabarak Khan, Toki Sherbakov, Joanne Jang, Peter Welinder, and Lilian Weng. 2022. Text and code embeddings by contrastive pre-training. *CoRR*, abs/2201.10005.

Judea Pearl. 2009. Causal inference in statistics: An overview. *Statistics Surveys*, 3(none):96 – 146.

Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1135–1144. ACM.

Marcel Robeer, Floris Bex, and Ad Feelders. 2021. Generating realistic natural language counterfactuals. In *Findings of the Association for Computational Linguistics: EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 16-20 November, 2021*, pages 3611–3625. Association for Computational Linguistics.

Alexis Ross, Ana Marasovic, and Matthew E. Peters. 2021. Explaining NLP models via minimal contrastive editing (mice). In *Findings of the Association for Computational Linguistics: ACL/IJCNLP*

2021, Online Event, August 1-6, 2021, volume ACL/IJCNLP 2021 of *Findings of ACL*, pages 3840–3852. Association for Computational Linguistics.

Alexis Ross, Tongshuang Wu, Hao Peng, Matthew E. Peters, and Matt Gardner. 2022. Tailor: Generating and perturbing text with semantic controls. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 3194–3213. Association for Computational Linguistics.

Candida S. Punla, https://orcid.org/ 0000-0002-1094-0018, cspunla@bpsu.edu.ph, Rosemarie C. Farro, https://orcid.org/0000-0002-3571-2716, rcfarro@bpsu.edu.ph, and Bataan Peninsula State University Dinalupihan, Bataan, Philippines. 2022. Are we there yet?: An analysis of the competencies of BEED graduates of BPSU-DC. *International Multidisciplinary Research Journal*, 4(3):50–59.

Yujun Shen, Jinjin Gu, Xiaoou Tang, and Bolei Zhou. 2020. Interpreting the latent space of gans for semantic face editing. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 9240–9249. Computer Vision Foundation / IEEE.

Sandra Wachter, Brent Mittelstadt, and Chris Russell. 2018. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harvard Journal of Law and Technology*, 31(2):841–87.

Xiao-Yong Wei and Chong-Wah Ngo. 2007. Ontology-enriched semantic space for video search. In *Proceedings of the 15th International Conference on Multimedia 2007, Augsburg, Germany, September 24-29, 2007*, pages 981–990. ACM.

Tongshuang Wu, Marco Túlio Ribeiro, Jeffrey Heer, and Daniel S. 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, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 6707–6723. Association for Computational Linguistics.

Linyi Yang, Eoin M. Kenny, Tin Lok James Ng, Yi Yang, Barry Smyth, and Ruihai Dong. 2020. Generating plausible counterfactual explanations for deep transformers in financial text classification. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 6150–6160. International Committee on Computational Linguistics.

A Methods

We present further in detail the two methods we introduced, namely Growing Language and Growing Net. These methods generate counterfactuals to explain the prediction made by any ML model for a given text document. Both of these methods replace words from the target text with similar words. Section A.3 presents the details of the implementation of the opponents.

A.1 Growing Language

Growing Language is a method employing a language model to generate a set of similar words for each word from the target text. Growing Language is composed of three phases:

1. Generates a set of similar words for each word from the target document based on a language model.

2. Replaces randomly words from the target by word from its corresponding set of similar words generated in the first step.

3. Tests whether counterfactuals are present, if yes, returns the closest one, otherwise goes to the first phase and reduces the similarity threshold of the words selected.

During the first phase, Growing Language employs the language model Spacy (Honnibal and Montani, 2017) to generate a set of similar words. The similarity is computed by the language model and we fixed the initial similarity threshold at 0.8 since initializing with a high similarity threshold induces a longer runtime. We also restrict the choice of similar words to words with the same part-of-speech tagger –*i.e:* verb, noun, determinant, etc... However, it is worth mentioning that any language model able to generate words and compute the distance between words could be used. Given an example of a complex model classifying the sentiment of a review and a target document x: *"This is a good article"* classified as positive, Growing Language produces sets for each word of the target. For instance, Growing Language generates five sets $\{s_1, ..., s_5\}$ containing similar words with their associated similarity to each word {This, is, a, good, article} as follows:

This:	{These (0.90); There(0.85)}
is:	{Exist (0.9); Happen (0.85); Occur (0.8)}
a:	{}
good:	{nice (0.95); poor (0.85); bad (0.80); evil (0.80)}
article:	{paper (0.95); movie (0.85); work (0.80)}

In the second phase, Growing Language randomly substitutes words from the target document with words from their corresponding set and generates artificial documents such as:

Artificial Document	Classification
there is a good article	O
this is a **nice** article	O
this is a **poor** article	X
this is **an evil** article	X
this is a **bad word**	X

where replaced words are highlighted in bold. Finally, Growing Language employs the prediction of the black-box model to identify whether some artificial documents are classified differently or not. If there is more than one enemy, Growing Language returns the one with the least number of modified words to the target document. Otherwise, Growing Language goes back to the first step and extends each set of similar words by reducing the threshold by two times the step which is initialized at 0.01. Hence, sets are bigger but composed of less similar words.

A.2 Growing Net

Our second method: Growing Net, employs the tree structure of WordNet (Fellbaum, 1998) to generate each set of close words. Hence, similarly to Growing language, Growing Net first generates sets of similar words for each term in the target document. The words added to the set associated with each target term share the same part-of-speech tagging. Moreover, they are close to the target word in the tree structure of WordNet. WordNet allows Growing Net to generate for each word, a set composed of its closest synonyms, antonyms, hyponyms, and hypernyms. Hyponyms and hypernyms are words or phrases that are respectively more specific and general than x_i. Going back to our running example of the target review *"This is a good article"*, classified as positive. Growing Net generates five sets of close words as follows:

This:	{ }
is:	{be; follow; live; exist}
a:	$\{the; A\}$
good:	{safe; bad; respectable; serious; effective}
article:	{paper; news article; magazine article}

Growing Net then randomly replaces words in the target document with words from its corresponding set of similar terms. Similarly to Growing Language and SEDC, Growing Net iteratively increases the distance between artificial instance and x. Hence, at each round, Growing Net increases the number of words modified. For instance, in the first round, Growing Net replaces randomly one word in the target sentence with words from its corresponding set. Thus, at round k, Growing Net replaces k words. The number k increases until Growing Net finds the closest counterfactual. Given once again our example of the commentary, Growing Net generates in the first round:

Artificial Text	Classification
this **be** a good article	O
this is a **safe** article	O
this is a good **news article**	O
this is **A** good article	O
this is a **poor** article	O

Then, in the next round, Growing Net substitutes two words as follows:

Artificial Document	Classification
this **be** a **safe** article	O
this is a **safe magazine article**	O
this **follow** a good **news article**	O
this is **A poor** article	X
this **live** a **serious** article	X

Since in the second round Growing Net obtains instances classified differently by the complex model, it returns the counterfactual with the smallest wup distance (Wei and Ngo, 2007) as an explanation.

In contrast to Growing Language which computes similarity in a latent space, Growing Net employs the tree structure of WordNet (Fellbaum, 1998) which is more interpretable. Hence, we represent in Figure 1 Growing Net closer to the transparent CEMs.

A.3 Opponents

Polyjuice: We employed the code for Polyjuice from the repository https://github.com/tongshuangwu/polyjuice with the default hyperparameters to perturb texts from each test set until we found 100 instances classified differently by the model.

SEDC: The code used to mask words will be given in our github repository upon acceptance as a version of the methods generating transparent explanations.

XSPELLS: We employed the V2 version of XSPELLS found in this repository https://github.com/lstate/X-SPELLS-V2 with the default hyperparameters.

CFGAN: We employed the code found in the official page of the paper release https://aclanthology.org/2021.findings-emnlp.306/. We run CFGAN with the default hyperparameters.

B Experimental Information

We present in this section information about the datasets, the classifiers, and the metrics employed in the experiments as well as in Appendix C.

Name	Nb Words			Instances
	Total	Average	STD	
Spam	15587	18.5	10.6	8559
Polarity	11646	20.8	9.3	10660
Fake †	19419	11.8	3.2	4025

Table 2: Information about the experimental datasets. † indicates generated datasets. The three columns under "Nb Words" represent respectively (a) the total number of distinct words in the whole dataset, (b) the average number of words per sentence, and (c) the standard deviation. The last column indicates the number of text documents per dataset.

B.1 Datasets

We employed three datasets, (a) spam detection from messages, (b) sentiment analysis, and (c) the detection of fake news in newspaper article titles. All datasets define two target classes and comprised between 4000 and 10660 textual documents. The average number of words in each document is comprised between 11.8 and 20.8 as reported in Table 2. We generated the fake news detection based on true titles from a newspapers dataset [1] and fake titles from a fake news dataset [2]. This dataset will be available on GitHub upon acceptance. Concerning the polarity [3] and spam[4] detection dataset, we employed the dataset from Kaggle. We split the dataset for training and testing using the sklearn function [5]: $train_test_split(x,\ y,\ test_size\ =\ 0.4,\ random_state\ =\ 1)$.

B.2 Black-box Classifiers

We evaluate our counterfactual methods on two classifiers of different architectures implemented in scikit-learn and already employed in (S. Punla et al., 2022). These black boxes are (i) a Random Forest (RF) with 500 tree estimators, and (ii) a simple neural network (DNN) with the same amount of neurons as there are words in the dataset. In addition to the class prediction associated with a textual document, the classifiers provide the probabilities prediction. The classifiers were trained on 60% of the dataset and their accuracy as well as the target instance to explain were tested on the remaining 40%. The average accuracy of the two classifiers for each dataset ranges from 67% to 100%. The detailed results are presented in Table 3.

Dataset	Model	
	Neural Network	Random Forest
Spam	100%	100%
Polarity	72%	67%
Fake	84%	84%

Table 3: Average accuracy of the two classifiers for each dataset.

[1] https://www.kaggle.com/datasets/rmisra/news-category-dataset

[2] https://www.kaggle.com/competitions/fake-news/overview

[3] https://www.kaggle.com/datasets/nltkdata/sentence-polarity

[4] https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset

[5] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

We used a count vectorizer [6] to convert the texts from the dataset into a matrix of tokens as input for the model.

C Runtime

dataset	method	DNN	RF
spam	SEDC	21.45 (12.57)	15.8 (9.36)
	Grow. Net	0.97 (1.0)	0.74 (0.8)
	Grow. Lang.	60.21 (15.55)	56.9 (13.84)
	Polyjuice	31.53	62.23
	cfGAN	1.4 (0.02)	1.43 (0.03)
	XSPELLS	219.45 (16.78)	197.63 (16.26)
fake	SEDC	30.7 (14.34)	13.02 (6.15)
	Grow. Net	1.54 (1.4)	1.03 (0.76)
	Grow. Lang.	54.81 (28.11)	54,66 (12,16)
	Polyjuice	38.23	70.04
	cfGAN	1.03 (0.18)	1.0 (0.13)
	XSPELLS	84.11 (6.47)	85.63 (7.0)
polarity	SEDC	12.91 (9.77)	12.39 (9.41)
	Grow. Net	0.69 (0.91)	0.63 (0.83)
	Grow. Lang.	74.7 (33.3)	73.56 (32.44)
	Polyjuice	81.07	82.49
	cfGAN	1.27 (0.23)	1.29 (0.25)
	XSPELLS	135.69 (19.32)	115.94 (10.9)

Table 4: Average runtime in seconds for an instance per dataset and black box of the five counterfactual methods. Note that the time for cfGAN does not take into account the time to train it on a given dataset. It takes around 6755, 4300, and 5770 seconds for respectively spam, fake, and polarity dataset.

In this section, we show the runtime of each of the six CEMs. Table 4 represents the average and standard deviation runtime of each CEM to generate a counterfactual. Results are presented per dataset and classifier. We note that CounterfactualGAN and Growing Net are the fastest to generate a counterfactual but CounterfactualGAN requires training the Variational AutoEncoder on every specific dataset. This training time varies from 4300 seconds on the fake news title detection to 6755 seconds on the spam detection dataset. Moreover, we observe that xSPELLS, as well as Growing Language, are the slowest methods to generate counterfactuals. Growing Language requires around 60 seconds to generate a counterfactual while the average runtime of xSPELLS can vary up to three times depending on the dataset. Concretely, since the two opaque methods need a training step to adapt the inner mechanisms to the dataset, we argue that our two novel methods and SEDC are faster and less effort-consuming to develop.

[6] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html