Explaining a Black Box without another Black Box

Anonymous ACL submission

Abstract

Although Counterfactual Explanation Methods (CEMs) are popular approaches to explain ML classifiers, they are less widespread in NLP. A counterfactual explanation encodes the smallest changes required in a target document to modify the classifier's output. Most CEMs find those explanations by iteratively perturbing the document until it is classified differently by the black box. We identified two main families of approaches for 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 overkill for downstream applications such as fake news detection or sentiment analysis since they add an additional level of opaqueness with no significant performance gain.

1 Introduction

The latest advances in Machine Learning (ML) have revolutionized many downstream NLP tasks such as fake news detection or sentiment analysis. However, the boost in accuracy achieved by modern ML algorithms comes at the expense of transparency and interpretability. 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. Unless the method relies on an inherently white-box model, explaining the outcomes of an ML agent requires the deployment of an explanation layer that opens the black box *a posteriori*. This is known as *post-hoc explainability*.

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. Consider a classifier for sentiment analysis applied to the book review "This is a really interesting book", which is 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. 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 compute counterfactual explanations 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, a more recent line of *opaque* methods embed the target text in a latent space on which perturbations are carried out subsequently (Robeer et al., 2021). This latent space is a compressed representation of the classifier's training data, which filters noise and focuses on the essential information for classification.

While one may think that latent-based CEMs outperform transparent methods, the empirical evidence provided in this paper suggests that, for some downstream NLP tasks such as spam detection, detection of fake news, or sentiment analysis, learning a compressed representation can be an overkill. More precisely, our experimental evaluation shows that 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 compute contrastive explanations for ML black-box algorithms by providing examples that resemble a target instance but that lead to a different answer in the black box. These counterfactual explanations convey the minimum changes in the input that would change the model's outcome. When it comes to NLP tasks, a good counterfactual explanation should be sparse, i.e., look like the target instance, and be plausible, i.e., read like something someone would say. As discussed by Wachter et al. (2018), counterfactual instances are particularly useful for applications in computational law.

Counterfactual explanations 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. In the following, we elaborate on the existing CEMs applied to textual data along a spectrum that spans from transparent to opaque approaches.

Transparent Approaches. Given an ML classifier and a target text (also called a document), transparent CEMs compute counterfactual explanations in a binary space, where each dimension represents the absence or presence of a word from a given vocabulary. Hence, to perturb a document, these methods toggle on and off 0's and 1's, which is tantamount to adding, removing, or replacing words until the classifier yields a different answer. Search for Explanations for Document Classification –SEDC– (Martens and Provost, 2014), and Plausible Counterfactual Instances Generation –PCIG– (Yang et al., 2020) are examples of transparent CEMs. These methods remove the words for which the classifier exhibits the highest *sensitivity*. These are words whose removal from the target document reduces the classifier's probability on the class of the original document – in favor of other classes. To do so both SEDC and PCIG assume that the classifier provides class probabilities for documents. Unlike SEDC, PCIG can also replace words from the target text with highly sensitive words that, when present, push the classifier's prediction toward other classes. Moreover, PCIG resorts a pre-trained masked language model to evaluate whether the perturbed documents are grammatically plausible. This language model is tailored for the financial domain –specifically for mergers and acquisitions. For this reason, we omit PCIG from our evaluation and propose two novel transparent CEMs instead.

Opaque Methods. Approaches such as XSPELLS (S. Punla et al., 2022) or CFGAN (Robeer et al., 2021) generate counterfactual explanations by perturbing the input text in a latent space defined by vectors of real numbers. These methods operate in three phases. First, they embed the target instance onto the latent space, learned through a Variational AutoEncoder (VAE) in the case of XSPELLS, and a pre-trained language model (LM) for CFGAN. Second, while the classifier's decision boundary is not traversed, these methods perturb the latent representation of the target phrase. This is done by adding Gaussian noise in the case of XSPELLS, whereas CFGAN resorts to a Conditional Generative Adversarial Network. Finally, a decoding stage generates sentences from the latent representation of the perturbed documents.

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 different types of CEMs spanning from fully transparent techniques such as SEDC (Martens and Provost, 2014) to fully opaque approaches, namely XSPELLS (S. Punla et al., 2022) and CFGAN (Robeer et al., 2021). We fill the middle ground with two novel methods, inspired on SEDC and PCIG, that perturb the target sentence while taking semantics into account. This is achieved through either a knowledge graph or a language model (LM). Our new methods are called Growing Net and Growing Language and lie halfway between the fully transparent and the fully latent approaches discussed in the previous section. This idea is illustrated in Figure 1.

Figure 1: A complexity scale for CEMs that goes from the most transparent method on the left SEDC to the most opaque methods cfGAN, and XSPELLS, passing by our methods Growing Net and Growing Language. Transparent methods perturb documents in a binary space; opaque methods do it in a latent space.

3.1 Proposed Methods

Similarly to SEDC, Growing Net and Growing Language search for counterfactuals by iteratively increasing the number of words modified in the target text. SEDC modifies the target by removing words, which often leads to non-sensical candidate counterfactual explanations. 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 sentence, our new 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.

3.2 Results

We compared the different CEMs of Figure 1 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 and (ii) outlierness. Second, we evaluate the methods themselves based on (iii) recall, (iv) stability, 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 deployed 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. This serves as input to our evaluation. Details of the implementation and the experimental datasets are provided in the appendix. 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 it is linguistically plausible, i.e., it reads like something someone actually would write or say. Minimality can be quantified through the similarity between the counterfactual and the target sentence. Similarly, outlierness can be operationalized as the similarity of the counterfactual to the "manifold" defined by a set of instances. Conversely, linguistic plausibility is subjective, thus it needs user studies for a proper evaluation.

Figure 2 shows the similarity of the counterfactuals to the target texts for our studied CEMs. Several ways to compute such similarity exist in the literature, e.g., scores based on LMs, L0, and cosine similarity. We show the aggregated results across the different datasets with the Spacy's LM similarity score since the language models capture the semantics better. The results for the L0 and cosine similarities are in the appendix. We first observe that middle-ground methods, in particular Growing Language, perform quite well compared to latent approaches. We note that XSPELLS incurs the biggest changes in the target phrase. While XSPELLS makes small perturbations to the target instance, those perturbations happen in a latent space learned through a 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 3 depicts the outlierness for our studied CEMs. This is computed as the average similarity between the computed counterfactual and the test instances of our experimental datasets – hence

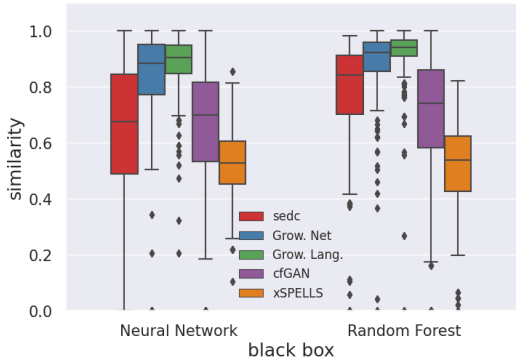

Figure 2: Minimality as the similarity between the closest counterfactual and the target target document.

the higher the better. We observe that XSPELLS 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 in terms of outlierness, in spite of not relying extensively on heavy NN-based machinery.

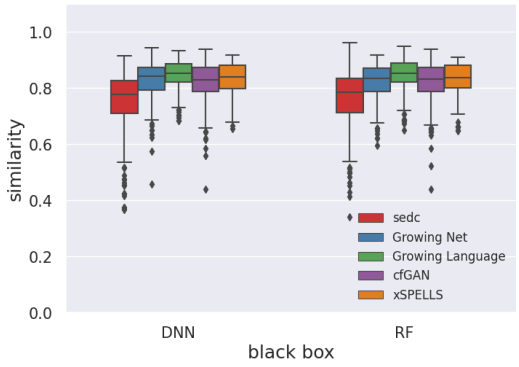

Figure 3: Outlierness as the similarity between the generated counterfactuals and the instances in the test set.

Method Quality. We now compare our CEMs in terms of (iii) recall, i.e., how often they can compute a counterfactual explanation, (iv) stability, i.e., how frequently the answer is stable across (five) different runs on the method on the same inputs, and (v) runtime.

Table 1 presents the recall results for each of the experimental datasets and classifiers. We note that except for spam detection with neural networks, XSPELLS achieves the highest recall. We also remark that the transparent methods perform better than CFGAN on the polarity review and spam detection tasks. In particular, Growing Net performs well for polarity analysis showing that replacement

dataset	method	DNN	RF
spam	SEDC	0.57	0.55
	Grow. Net	0.36	0.35
	Grow. Lang.	**0.65**	0.49
	cfGAN	0.57	0.57
	XSPELLS	0.34	**0.61**
fake	SEDC	0.90	0.98
	Grow. Net	0.79	0.59
	Grow. Lang.	0.86	0.86
	cfGAN	**1**	**1**
	XSPELLS	**1**	**1**
polarity	SEDC	0.95	0.92
	Grow. Net	**1**	0.85
	Grow. Lang.	0.93	0.93
	cfGAN	0.65	0.65
	XSPELLS	**1**	**1**

Table 1: Average recall per dataset and black box of the five counterfactual methods.

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 recall at the expense of longer runtimes. Our results on stability suggest that transparent methods are less sensitive to random seeding than methods based on NN learning. 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 its decoding stage. The results are provided in Appendixes C.1 and C.2.

4 Conclusion

The empirical 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. 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.

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*.

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

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

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.

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.

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.

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 Proposed 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.

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.

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]. These datasets will be available on GitHub upon acceptance.

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 70% of the dataset and their accuracy as well as the target instance to explain were tested on the remaining 30%. The average accuracy of the two classifiers for each dataset ranges from 67% to 100%. The detailed results are presented in Table 3.

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

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

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.

B.3 Metrics

We evaluated in Figure 2 the minimality of a counterfactual by measuring its similarity to the target document. Similarly, Figure 3 shows the outlierness distance from each counterfactual generated to the manifold. Here, the manifold is represented by the similarity to the closest text document from the test set sharing the counterfactual's classification. Hence, these two metrics require measuring the similarity between two texts, which is a hot topic in NLP. Employing a specific metric involves a specific purpose, for instance, the l0 similarity represents the sparsity of the explanation while a language model similarity measures the general meaning similarity. Thus, we compute the similarity between two text documents t_1 and t_2, by employing three different metrics:

- **L0:** Computes the ratio between the number of words in t_1 that are in t_2 by the number of words from t_2 present in t_1: $\frac{|t_1 \cap t_2|^2}{|t_1||t_2|}$
- **Cosine Similarity:** Measures the similarity between two vectors of an inner product space. Here the two vectors correspond to a bag of words representation of t_1 and t_2.
- **Language Model:** We employ SpaCy (Honnibal and Montani, 2017) as the reference language model and its similarity method as a reference for the language model similarity. SpaCy compresses words onto a latent representation of 300 dimensions and averages each word representation in a document to generate the document's latent representation.

Since stop words may influence the similarity computation while not impacting the classification or overall sense, we employ the natural language toolkit library [3] to remove every stop word before computing any similarity. We chose the language model similarity in the paper, but present results with additional similarities metrics in Section C.

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

C Additional Results

In this section, we introduce new findings as well as disaggregated results. We first show the stability and runtime of each of the five CEMs. Finally, we present the minimality and outlierness results per dataset with different metrics.

C.1 Stability

For each of the five CEMs, we iterate five times over the same target document and measure the average similarity measure between each of the five counterfactuals generated. This similarity is computed five times for 100 different instances on two black boxes, per dataset and aggregated. We compute the similarity of two counterfactuals based on the language model metric.

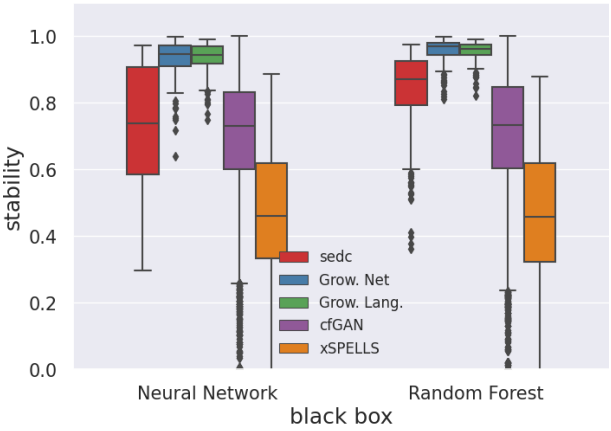

Figure 4: Stability score of the five CEMs for two black boxes. Results are aggregated over three datasets.

Figure 4 shows that both Growing Language and Growing Net are the most robust to uncertainty. These methods obtain almost a stability maximal of one while opaque methods and especially xSPELLS have a wide variability between different runs. We expect these results to be impacted by the training of the latent module such as the VAE or the GAN. Indeed, Growing Language and Growing Net are general methods, therefore the sets of similar words are the same over successive generations. Our results also present better stability with SEDC compared to the two opaque methods, but wider variance on the Neural Network classifier. We note that the high stability of Growing Language, Growing Net, and SEDC is due to the

iterative mechanism of extending the perturbation by increasing the number of words modified.

C.2 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)
	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)
	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)
	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.

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.

C.3 Minimality

We present in Figure 5 to 7, the minimality score of the five CEMs over two classifiers for the three datasets. For each CEM, dataset, and classifier, we compute the minimality with the language model measure for 100 target documents and their corresponding counterfactuals. Figure 5 to 7 show the similarity over respectively fake news detection, spam detection, and polarity review.

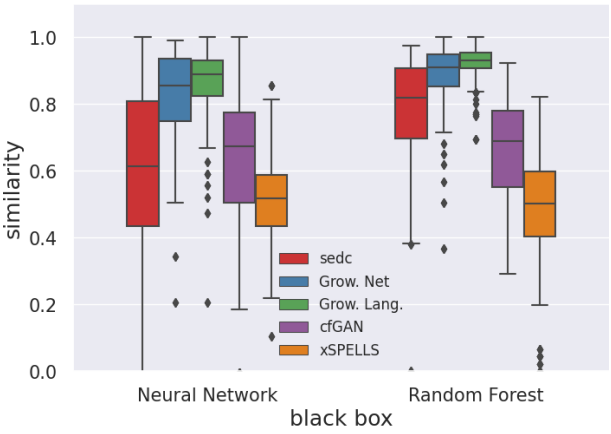

Figure 5: Minimality score of the counterfactuals generated by the five CEMs over two classifiers and the fake news detection dataset.

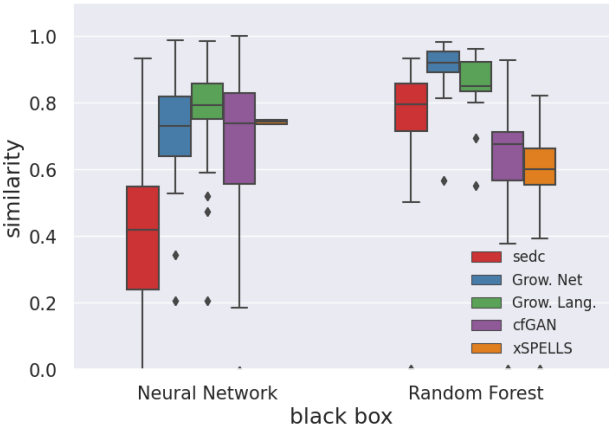

Figure 6: Minimality score of the counterfactuals generated by the five CEMs over two classifiers and the spam detection dataset.

We observe from these Figures that Growing Language and Growing Net generate closer counterfactuals to the target since the similarity is higher for each dataset and classifier. Moreover, we note that results are similar for the fake news detection and the polarity review datasets while the spam detection dataset induces lower similarity. The spam detection dataset is the most difficult for the transparent methods since messages are written for SMS. Hence, it is not so well written – many spelling mistakes are present, and replacing poorly written words with error-free words affects the overall closeness of the counterfactuals.

Figure 8 and 9 show the average minimality em-

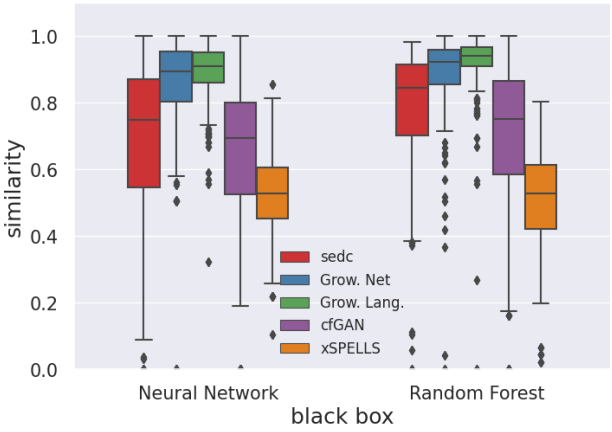

Figure 7: Minimality score of the counterfactuals generated by the five CEMs over two classifiers and the polarity review dataset.

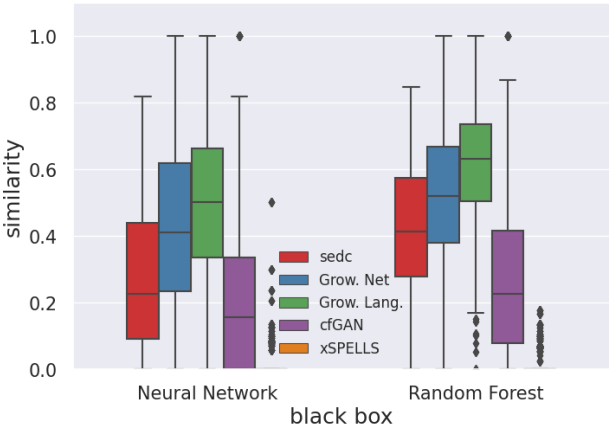

Figure 9: Minimality score of the five CEMs over two classifiers aggregated over three datasets. The similarity is computed based on the cosine metric.

ploying two different similarity metrics: respectively the L0 and the cosine. First, results are consistent with the language model metric since both Growing Net and Growing Language obtain the highest similarity. Second, we remark that the opaque methods may generate counterfactuals comprising zero words in common with the target document.

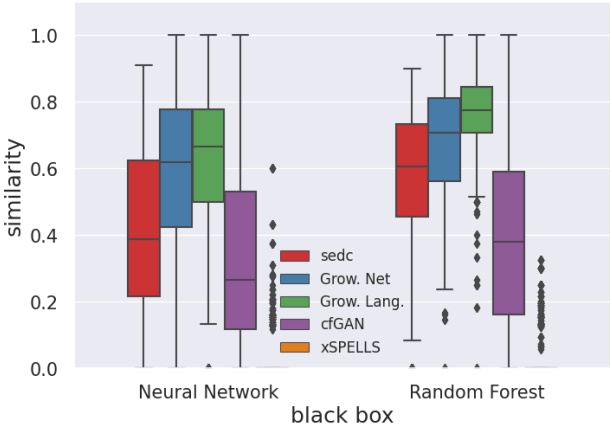

Figure 8: Minimality score of the five CEMs over two classifiers aggregated over three datasets. The similarity is computed based on the L0 metric.

C.4 Outlierness

We present in Figure 10 to 12 the outlierness score of the five CEMs for respectively the fake news detection, the spam detection, and the polarity review datasets. The similarity is measured through the language model metric and results are shown by classifiers. We observe that for every dataset, SEDC generates counterfactuals the farthest from the domain. This makes sense since the generated counterfactuals are only subparts of the target document with hidden parts. Conversely, we remark that Growing Language and xSPELLS are the methods obtaining the highest average outlierness score for every dataset and classifier.

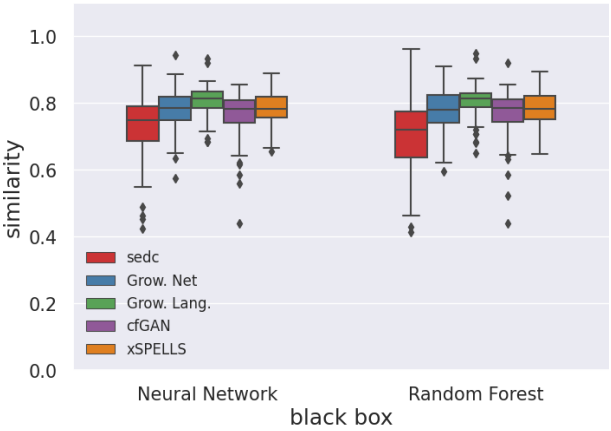

Figure 10: Outlierness score of the five CEMs over two classifiers for the fake news detection dataset. The similarity is computed with the language model metric.

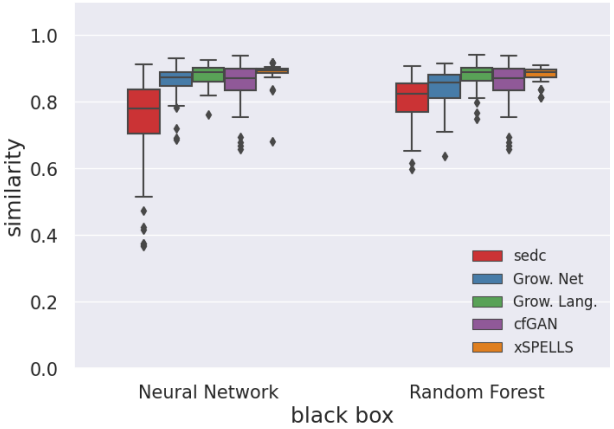

Figure 11: Outlierness score of the five CEMs over two classifiers for the spam detection dataset. The similarity is computed with the language model metric.

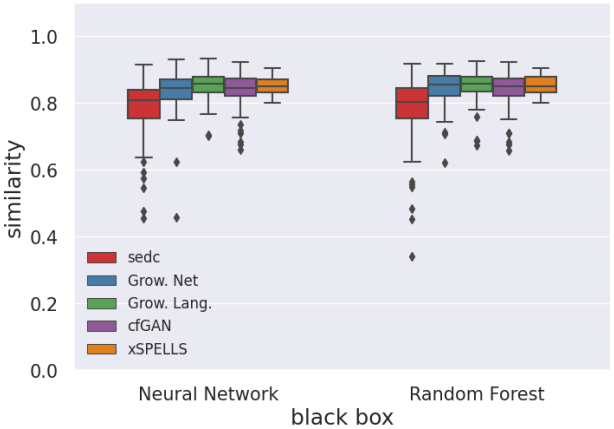

Figure 12: Outlierness score of the five CEMs over two classifiers for the polarity review dataset. The similarity is computed with the language model metric.