Enhancing textual counterfactual explanation intelligibility through Counterfactual Feature Importance

Milan Bhan, Jean-Noël Vittaut, Nicolas Chesneau, Marie-Jeanne Lesot

Enhancing textual counterfactual explanation intelligibility through Counterfactual Feature Importance

Milan Bhan[1,2] **Nicolas Chesneau**[1] **Jean-Noël Vittaut**[2] **Marie-Jeanne Lesot**[2]

[1]**Ekimetrics, Paris, France**
[2]Sorbonne Université, Paris, France
{milan.bhan, nicolas.chesneau}@ekimetrics.com
{jean-noel.vittaut, marie-jeanne.lesot}@lip6.fr

Abstract

Textual counterfactual examples explain a prediction by modifying the tokens of an initial instance in order to flip the outcome of a classifier. Even under sparsity constraint, counterfactual generation can lead to numerous changes from the initial text, making the explanation hard to understand. We propose Counterfactual Feature Importance, a method to make non-sparse counterfactual explanations more intelligible. Counterfactual Feature Importance assesses token change importance between an instance to explain and its counterfactual example. We develop two ways of computing Counterfactual Feature Importance, respectively based on classifier gradient computation and counterfactual generator loss evolution during counterfactual search. Then we design a global version of Counterfactual Feature Importance, providing rich information about semantic fields globally impacting classifier predictions. Counterfactual Feature Importance enables to focus on impacting parts of counterfactual explanations, making counterfactual explanations involving numerous changes more understandable.

1 Introduction

The recent development of the Transformer architecture (Vaswani et al., 2017) has led to great advances in Natural Language Processing (NLP). The inherent complexity of these widespread black box models comes along with the difficulty to understand their predictions. The field of eXplainable Artificial Intelligence (XAI) aims to develop methods to interpret and explain such model behaviour (Molnar et al., 2021). A first main category of XAI methods is called local feature importance. It consists in computing the impact of each input feature in the decision made by the considered machine learning system. A second family explains contrastively by identifying slight perturbations in the initial instance leading to another outcome. Such modified instances are called counterfactual examples.

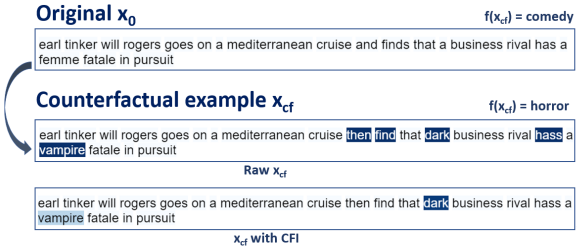

Figure 1: Example of an initial instance classified as comedy and its counterfactual example classified as horror synopsis. The raw x_{cf} is the counterfactual example whose modified tokens are highlighted. Counterfactual feature importance enables to highlight important changes when counterfactual examples are not sparse. Here, CFI highlights **a** $\rightarrow$ **dark** and **femme** $\rightarrow$ **vampire** whereas others changes are not outlined anymore.

When counterfactual examples involve numerous modifications in the initial instance, despite considering sparsity constraints, the identification of important token changes becomes difficult. In this paper we introduce the notion of Counterfactual Feature Importance (CFI), to quantify the impact of each feature modification from an initial instance to its counterfactual example. For example, given a movie genre classifier and a specific counterfactual example provided with a given method, CFI highlights important modifications (see Figure 1) to explain the label flipping.

The main contributions of this paper are summarized as follows:

1. The concept of Counterfactual Feature Importance is presented.

2. Two instantiations of CFI are proposed, depending on the available information about

the classifier and the used counterfactual generator.

3. Global counterfactual feature importance (g-CFI), summarizing the information contained in local CFI, is introduced.

This paper first recalls some basic principles of XAI in NLP with a focus on counterfactual generation in Section 2. We then formalize the CFI method at a local and global scale and propose two ways of computing CFI in Section 3. We finally illustrate the relevance of CFI experimentally on counterfactual examples previously obtained to explain two different classifiers.

2 XAI Background

In this section, we recall some basic principles of XAI methods and existing counterfactual generation methods in NLP.

2.1 Local feature importance

Let $f : \mathcal{X} \rightarrow \mathcal{Y}$ be a NLP classifier mapping an input space $\mathcal{X}$ of token sequences to an output space $\mathcal{Y}$ of classes. Let $x_0 = [t_1, ..., t_d] \in \mathcal{X}$ be a sequence of tokens of interest of maximum size d with $f(x_0) = y_0$. Each token belongs to a dictionary $\mathcal{D}$. A local feature importance operator $g : \mathcal{X} \rightarrow \mathbb{R}^d$ explains the prediction through a vector $[z_1, ..., z_d]$ where z_i is the contribution of the i^{th} token.

Two very common local feature importance methods are LIME (Ribeiro et al., 2016) and SHAP (Lundberg and Lee, 2017). LIME relies on a local approximation of f by an explainable linear model whereas SHAP computes feature contribution through an approximation of Shapley values. Integrated Gradients (Sundararajan et al., 2017) constitute another method specific to deep learning models that approximates the gradient integral of the classifier outputs over the straight line between the instance to explain and a user-selected baseline x^*. The definition of the baseline is essential since it strongly impacts the resulting explanation. Integrated Gradients can only be computed on ML systems allowing gradient computation, such as deep learning models.

2.2 Counterfactual explanation

Counterfactual explanations emphasize what should be different in an input instance to change the outcome of a classifier. Their interest in XAI has been established from a social science perspective (Miller, 2019) in particular. The counterfactual example generation can be formalized as a constrained optimization problem. For a given classifier f and an instance of interest x_0, a counterfactual example x^{cf} must be close to x_0 to highlight minimal changes leading to label flipping. Formally it is defined as:

$$x^{\text{cf}} = \underset{z \in \mathcal{X}}{\operatorname{argmin}}\, c(x_0, z) \;\text{ s.t. }\; f(z) \neq f(x_0) \quad (1)$$

with $c : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ a cost function integrating several constraints to ensure various desirable properties briefly discussed below. The lower the cost function, the better the counterfactual explanation. The simplest case is when c is a distance function.

Many desirable properties for counterfactual explanations have been proposed (Guidotti, 2022; Mazzine and Martens, 2021) to ensure their informative nature. *Sparsity* measures the number of elements changed between the instance of interest and the generated counterfactual example. It is defined as the l_0 norm of $x^{\text{cf}} - x$. *Plausibility* encompasses a set of characteristics to ensure that the counterfactual explanation is not out-of-distribution (Laugel et al., 2019), while being feasible (Poyiadzi et al., 2020) and actionable.

2.3 Textual counterfactual

This section focuses on the case of textual counterfactual generators, presenting two categories in turn.

2.3.1 Text editing heuristics.

A first family of methods addresses the problem introduced in Eq. 1 by slightly modifying the input text to be explained with heuristics.

Model specific methods depend structurally on the models they seek to explain. CLOSS (Fern and Pope, 2021) focuses on the embedding space of the classifier to explain. After generating counterfactual candidates through optimization in the classifier latent space, the most valuable ones are selected according to an estimation of Shapley values. MiCE (Ross et al., 2021) sequentially masks parts of the initial text and performs span infilling using a T5 (Raffel et al., 2019) fine-tuned on the corpus of interest. MiCE targets tokens with high predictive power using gradient attribution metrics. TIGTEC (Bhan et al., 2023) proposes a *model-agnostic* and -*specific* version by targeting

important tokens with local feature importance method such as SHAP or attention coefficient from Transformer-like models. It sequentially replaces tokens by decreasing order of importance using a BERT mask language model. At each step, replacement is made to ensure proximity to the initial instance and to target label flipping.

Generating counterfactual examples shares similarities with generating *adversarial attacks*, aiming to incorrectly flip the prediction by minimally editing the initial text. Numerous heuristics have been proposed differing in constraints, text transformation methods and search algorithms (Morris et al., 2020). Contrary to counterfactual explanations, adversarial attacks seek to fool intentionally a model. Therefore, the resulting text is not generated with an explanatory purpose.

2.3.2 Text generation with large language models

A second category of methods generates counterfactual examples in NLP with large pre-trained *generative language models*. A first approach (Madaan et al., 2022) applies a Plug and Play language model (Dathathri et al., 2020) methodology to generate text under the control of the classifier to explain. It consists in learning latent space perturbations from encoder-decoder models such as BART (Lewis et al., 2020) in order to flip the outcome.

Polyjuice (Wu et al., 2021) proposes to fine-tune a GPT-2 (Radford et al., 2019) model on a set of predefined tasks. It results in a generative language model capable of performing negation, quantification, insertion of tokens or sentiment flipping based on prompt engineering. Polyjuice needs to be trained in a supervised way on ground truth counterfactual examples in order to be able to generate the expected text.

3 Counterfactual feature importance

This section introduces the notion of Counterfactual Feature Importance (CFI). We present two instantiations of CFI, based either on gradient computation from the classifier or inherent information from the counterfactual generator initially used. We propose a *model-specific* approach based on Integrated Gradients called IG-CFI and a *method-specific* one called TIGTEC-CFI directly resulting from the TIGTEC loss evolution during counterfactual search. Finally we define the notion of global Counterfactual Feature Importance to compute pairwise token importance at a global scale.

3.1 Motivation and definition

As presented in the previous section, sparsity is an expected attribute of counterfactual examples. Sparsity ensures counterfactual explanation intelligibility by highlighting few changes leading to label flipping. However, this constraint is not equally addressed among the different existing counterfactual generators. MiCE succeeds more than other methods to find counterfactual examples to explain a sentiment analysis classifier, while generating less sparse explanations than CLOSS and TIGTEC (see (Bhan et al., 2023)). Non-sparse counterfactual examples are difficult to understand, since the label flipping can be explained by the numerous substitutions in the initial instance. Therefore, we propose a method to quantify the importance of each change in order to better understand the explanation provided by a counterfactual example. We call such a method Counterfactual Feature Importance.

We follow here the notations introduced in Section 2. Given a classifier $f : \mathcal{X} \to \mathcal{Y}$, an instance of interest $x_0 \in \mathcal{X}$ and a counterfactual example x_{cf} generated with a counterfactual generator $\mathcal{M}$, h is a counterfactual feature importance operator, $h : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^d$. This operator explains a prediction by computing the importance of each token change between x_{cf} and x_0 with a vector of importance. The i^{th} component of this vector is the contribution of the i^{th} token substitution to the label flipping. Then, CFI can be seen as a pairwise feature importance between an instance and its counterfactual explanation. Therefore, unchanged tokens between the initial instance and its counterfactual example must have a null CFI. In the following, we formalize the two different CFI instanciations by assuming that each initial token from x_0 is replaced by only one token to reach x_{cf}. However, these two approaches could also be applied with token substitutions of various length, following the same logic.

CFI goes one step further, as compared to classical XAI methods, in the explanation of a classifier prediction by applying local feature importance attribution to counterfactual explanation.

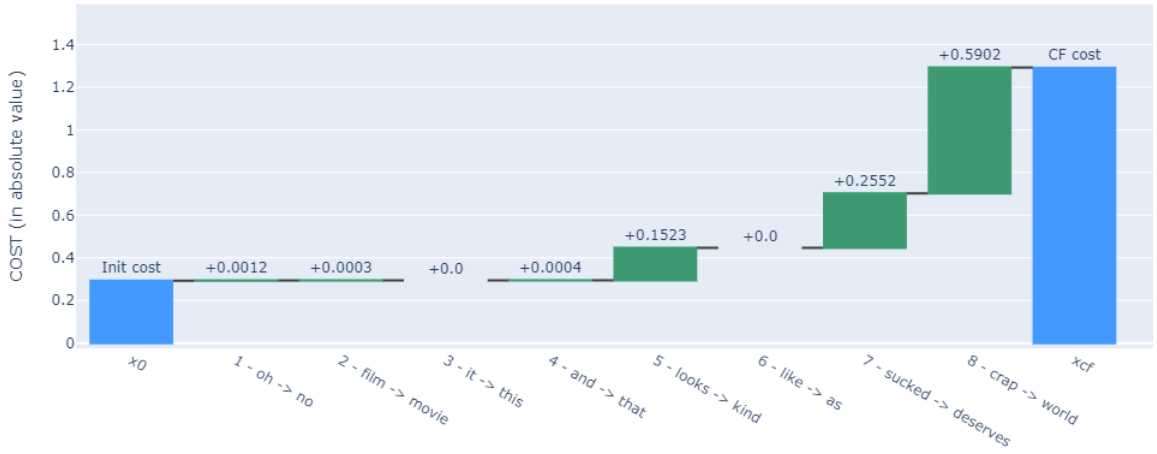

Figure 2: Example of TIGTEC-CFI with nine consecutive changes from negative sentiment x_0 to a positive sentiment x_{cf}. CFI is defined as the cost difference induced by each token substitution during TIGTEC counterfactual search. The **crap** $\rightarrow$ **world** change has the highest TIGTEC-CFI considering the resulting impact on the cost function.

3.2 IG-CFI

We define here a first instanciation of CFI based on Integrated Gradients computation.

As introduced in Section 2, Integrated Gradients are defined as a local feature importance explanation. Their purpose is to explain the difference between an instance of interest and a chosen baseline x^* by assigning classifier output gradient integrals to instance modifications.

We propose to define Integrated Gradients Counterfactual Feature Importance (IG-CFI) as the Integrated Gradients obtained when setting as baseline $x^* = x_0$. Therefore, IG-CFI consists in computing the integral of gradients of the classifier's output over the straight-line between x_0 and x_{cf}. Formally, for the counterfactual explanation x_{cf} of x_0, the IG-CFI related to the i^{th} token change is defined as:

$$\texttt{IG-CFI}_i(x_0, x_{cf}) = (e_i^{x_{cf}} - e_i^{x_0}) \times \int_{\alpha=0}^{1} \nabla_i f\Big(e_i^{x_0} + \alpha \times (e_i^{x_{cf}} - e_i^{x_0})\Big) d\alpha \quad (2)$$

where e_i^x is the embedding of the i^{th} token of a sequence x obtained from the classifier f and ∇_i denotes the gradient along the i^{th} dimension.

This way, Integrated Gradients are computed with respect to embeddings from the classifier latent space to ensure derivability. This instantiation of CFI ensures that the unchanged tokens have a null counterfactual feature importance.

IG-CFI is a *model-specific* approach since it needs to have access to the parameters of the classifier to compute gradients. On the other hand, IG-CFI is *method-agnostic* since it is applicable to counterfactual examples obtained via any counterfactual generation method. However, computing gradients over straight-line in latent space can be text unrepresentative due to the inherent discreteness of text. The CFI instanciation introduced in the next section proposes to address such text unrepresentativeness.

3.3 TIGTEC-CFI

In this section, we introduce a CFI instantiation depending on the counterfactual generator used to compute the considered counterfactual examples. Some textual counterfactual generators search for counterfactual by sequentially modifying the input text until a stop condition is reached (see Section 2). We present how the text sequence breaking down the change from x_0 to x_{cf} from such *text editing heuristics* can provide CFI. We illustrate this approach by considering such a counterfactual generator, namely TIGTEC (Bhan et al., 2023). It can be applied with any other sequential text editing

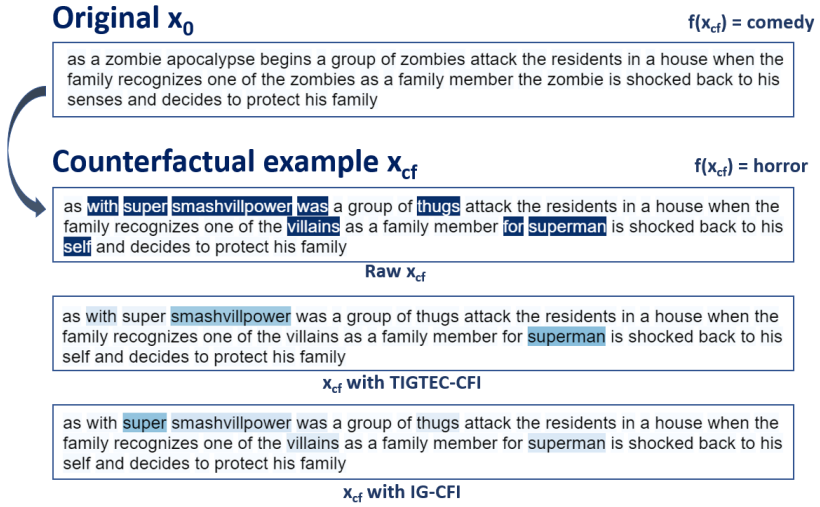

Figure 3: TIGTEC-CFI and IG-CFI example on a counterfactual example classified as an horror synopsis. Tokens highlighted in blue in the raw counterfactual example are those that have replaced initial ones. Below, the more pronounced the shade of blue, the higher the CFI.

heuristics, such as MiCE (Ross et al., 2021).

Let $x_0 \xrightarrow{\varphi_1} x_1 \xrightarrow{\varphi_2} \ldots \xrightarrow{\varphi_{p-1}} x_{p-1} \xrightarrow{\varphi_p} x_{cf}$ be the text sequence breaking down the change from x_0 to x_{cf} during the counterfactual search, φ_i the index of the modified token at the i^{th} step and $c : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ the cost function to minimize to generate counterfactual examples as introduced in Equation 1.

For TIGTEC, c is defined as an aggregation of the target class probability score and semantic distance to the initial instance. We propose to define TIGTEC Counterfactual Feature Importance (TIGTEC-CFI) as the cost difference induced by sequential token modifications during counterfactual search. Formally, TIGTEC-CFI is defined as:

$$\texttt{TIGTEC-CFI}_i(x_0, x_{cf}) = \begin{cases} \Delta c(x_k, x_0) & \text{if } \varphi_k = i \\ 0 & \text{otherwise.} \end{cases} \quad (3)$$

with $\Delta c(x_k, x_0) = c(x_{k-1}, x_0) - c(x_k, x_0)$ the cost difference between x_k and x_{k-1}.

Figure 2 shows an example of TIGTEC-CFI with the following movie review predicted as negative and its token changes highlighted in box and in bold to switch the label to a positive sentiment:

"i think i will make a movie next weekend ***oh → no*** *wait im workingoh im sure i can fit* ***it → this*** *in it looks like whoever made this film fit it in i hope the makers of this* ***crap → world*** *have day jobs because this* ***film → movie*** ***sucked → deserves*** *it* ***looks → kind*** ***like → as*** *someones home movie* ***and → that*** *i dont think more than 100 was spent making it total crap who lets this stuff be released".*

In this example TIGTEC-CFI is almost null at the first step when performing the **oh → no** substitution because it only induces a small variation of cost, whereas the **crap → world** replacement decreases sharply the cost, making the counterfactual candidate acceptable. Two other token substitutions are evaluated as important: **looks → kind** and **sucked → deserves**. This way, TIGTEC-CFI emphasize three token substitutions, which is lower than the nine token changes initially suggested.

This approach is inexpensive compared to IG-CFI since it can be directly measured during the counterfactual search. TIGTEC-CFI also differs from IG-CFI as it computes CFI with texts without referring to any latent space, which avoids text non-representativeness and limits the risk of considering out-of-distribution instances.

3.4 Global-CFI

IG-CFI and TIGTEC-CFI both are local explainers, as they explain the prediction for a given instance of interest, x_0. This section turns to the generation of explanations at a global scale. We present g-CFI that provides information about a given token couple (t_1, t_2), computing the importance of substituting t_1 with t_2 for the considered classifier.

We propose to build global Counterfactual

Feature Importance (g-CFI) by summing up local CFI on the whole dataset. Following the previously introduced notations, we denote the instance of interest x_0 and its counterfactual example x_{cf} obtained with a counterfactual generator $\mathcal{M}$, and the counterfactual feature importance operator h that measures the importance of a token change noted $h(x_0, x_{cf}) = h(x_0, \mathcal{M}(x_0))$ in the label flipping.

Considering a specific pair of tokens (t_1, t_2), its related g-CFI can be formalized as follows:

$$\texttt{g-CFI}(t_1, t_2) = \sum_{x \in \mathcal{T}} \sum_{i \leq d} l(x, i, t_1, t_2)$$

where $\mathcal{T}$ is the text corpus of interest in which g-CFI is computed, and $l(x, i, t_1, t_2)$ is defined as:

$$l(x, i, t_1, t_2) = h_i\big(x, \mathcal{M}(x)\big) \mathbb{1}_{(x_i, \mathcal{M}(x)_i) = (t_1, t_2)}$$

This way, g-CFI is defined as a global pairwise token importance that evaluates which token pairs are the most important at a global scale to switch label.

Such a definition of g-CFI tends to emphasize frequent token changes. Global CFI could also be aggregated by computing the average local CFI. However, we assume that rare token changes are globally less informative about a classifier than frequent ones. Besides, we believe that g-CFI is more informative than a simple token pair frequency calculus. By weighting frequency by CFI, recurrent token pairs with low/middle average CFI appear less important at a global scale.

4 Experimental results

This section presents the experimental results obtained on two data sets with two binary classifiers. IG-CFI and TIGTEC-CFI are computed and compared based on two related sets of counterfactual examples generated with TIGTEC. Then we compute g-CFI in order to assess which token pairs impact globally the most these classifiers.

4.1 Experimental Setup

We apply TIGTEC on two DistilBERT (Sanh et al., 2020) binary classifiers. The first classifier performs sentiment analysis on the IMDB database (Maas et al., 2011) containing movie reviews. The second is trained on movie genre classification on a dataset of horror and comedy synopses from Kaggle[1]. More information about the datasets, the performance of the classifiers and the TIGTEC hyperparameters used are provided in Appendix A. Respectively 982 and 419 counterfactual examples are generated from IMDB and the genre synopses datasets from TIGTEC in which we compute IG-CFI and TIGTEC-CFI.

Token change	IG-CFI ranking	TIGTEC-CFI ranking
a → with	8	3
zombie → super	1	4
apocalypse → smashvillpower	2	2
begins → was	7	7
zombies → thugs	5	5
zombies → villains	4	6
the → for	6	8
zombie → superman	3	1
senses → self	9	9

Table 1: IG-CFI and TIGTEC-CFI ranking comparison on an example involving nine changes. The two CFI methods only agree on four token modifications (highlighted in bold).

Correlation	IMDB	Genre
Spearman	0.26*	0.42*
Kendall	0.21*	0.53*

Table 2: IG-CFI and TIGTEC-CFI ranking correlations on counterfactual examples from sentiment analysis (IMDB column) and movie genre classification (Genre column). Values with * are statistically significantly different from 0 at a risk level of 1%.

4.2 IG-CFI vs. TIGTEC-CFI

IG-CFI and TIGTEC-CFI can lead to different explanations. Figure 3 gives an example of an instance classified as comedy and its related counterfactual example classified as an horror synopsis. In this case, TIGTEC-CFI and IG-CFI attribute different token change importance by emphasizing in blue different tokens. The **zombie → super** substitution is assessed as more important by IG-CFI than TIGTEC-CFI to explain label flipping. However, TIGEC-CFI considers **zombie → superman** as more important than IG-CFI.

We compare important tokens based on IG-CFI and TIGTEC-CFI approaches. We apply each method on the two sets of counterfactual texts previously introduced and compare them. IG-CFI and TIGTEC-CFI are compared through their

[1] https://www.kaggle.com/competitions/movie-genre-classification/overview

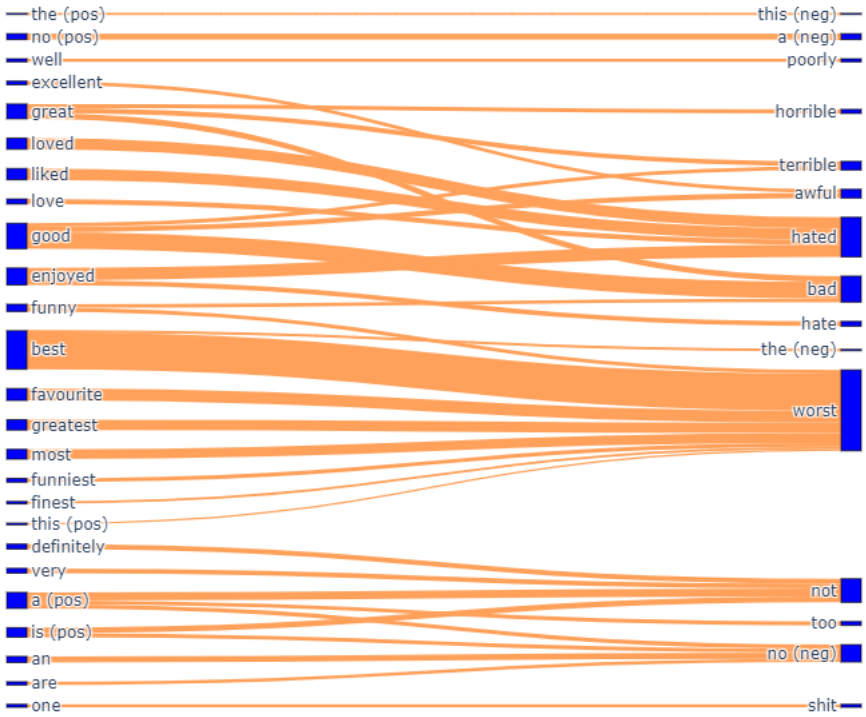

Figure 4: Top 35 important token pairs relatively to TIGTEC-CFI and sentiment analysis. The left side corresponds to tokens associated with positive IMDB reviews, while the right column is related to negative sentiments. The stronger the link between two tokens, the higher the importance of the pair.

resulting token ranking by order of CFI. Non-modified tokens are filtered out to focus only on token subject to CFI computation. Counterfactual examples obtained with only one token substitution are not considered either since their resulting CFI rankings necessarily perfectly match. Following the example presented Figure 3, Table 1 illustrates the two resulting rankings obtained with IG-CFI and TIGTEC-CFI.

Ranking comparison is done with Spearman and Kendall rank correlations (see Table 2). Movie genre IG-CFI and TIGTEC-CFI rankings are globally more similar than the ones obtained from IMDB. Finally, IG-CFI and TIGTEC-CFI seem complementary although moderately correlated.

4.3 Global-CFI results

The results obtained with IG-CFI and TIGTEC-CFI are aggregated at a global scale. Each token CFI is aggregated with respect to its related label. For example, **the** $\rightarrow$ **this** is considered differently if the predicted label of the initial instance is positive or negative. However, token CFI is aggregated in a symmetric way, which makes **love** $\rightarrow$ **hate** equivalent to **hate** $\rightarrow$ **love**. From this perspective, g-CFI enables to build label-specific semantic fields and their interactions. In the following, considering two different tokens t_1 and t_2, we denote (t_1,t_2) equivalently to (t_2,t_1).

Figure 4 shows the 35 most important token pairs relatively to the global TIGTEC-CFI on sentiment analysis. Global TIGTEC-CFI is provided for movie genre classification in Appendix A. The most important token pairs on sentiment analysis are (*best*, *worst*), (*good*, *bad*) or (*liked*, *hated*). In this 35 token pairs, mainly two types of token modification stand out: a sentiment-oriented token is replaced by its antonym, and an indefinite article is replaced by a negation adverb. Considering more token pairs could bring up unexpected tokens

and target biases or classifier errors. Token pairs could also be aggregated at a highest level of abstraction by lemmatizing tokens, merging for example "loved" and "love" CFI.

5 Discussion

In this paper we have introduced the concept of counterfactual feature importance in textual framework. The purpose of CFI is to assess the impact of every feature modification, from an initial instance to its counterfactual example. When counterfactual examples are not sparse, CFI highlights important modifications, making the explanation more competitive. CFI can be aggregated at a global scale, giving valuable insights about the most important token pairs to switch label.

While CFI-based explanations may appear intuitive, it is important to verify this through a human-in-the-loop evaluation before making any definitive conclusions. CFI can be built in different ways: two approaches have been developed, respectively based on Integrated Gradients and the TIGTEC counterfactual generator. We believe that CFI can also be computed in other ways. Other local feature importance methods such as LIME and SHAP can be used to compute CFI by computing decomposition difference of x_{cf} from that of x_0. Loss break down from other sequential counterfactual generators such as MiCE can be used as well.

Besides, g-CFI can help in comparing textual counterfactual generators. Since these generators differ how they target important tokens and generate new text, g-CFI could bring to light differences in the resulting semantic fields. Such analyses could lead to a better understanding of textual counterfactual methods and foster their enhancement.

Finally, the diversity of CFI approaches raises the need of their comparison beyond the similarity analysis performed above. Moreover, the qualitative assessment of the explanations provided by CFI requires human intervention. Human-grounded experiments would enable to compare the quality of CFI explanations to classical counterfactual examples.

6 Conclusion

Textual counterfactual generators sometimes fail to provide sparse explanations. The high number of changed tokens between the initial instance and its counterfactual example make the explanation difficult to understand. We have proposed Counterfactual Feature Importance (CFI) to assess which token changes are the most impactful. CFI enables to focus on important tokens, which is especially useful in the case of non-sparse explanations. Such explanations can be aggregated at a global scale in order to assess the most important token pairs leading to label flipping. In this papier We have only focused on counterfactual explanations. However, CFI can also be applied to adversarial attacks in order to evaluate the token changes that have the most impact on label flipping to fool a model.

CFI is one step further in the understanding of NLP classifiers. We believe that the concept of CFI is also applicable to image and tabular data, as long as counterfactual explanations are previously generated. Therefore, CFI can benefit to any classifier by making counterfactual explanations easier to understand. The generalizability of CFI makes this concept particularly promising.

Ethics Statement

Like any XAI methods, CFI explanations must be taken with caution. Such methods only provide insights about what is important according to a specific classifier. These explanations do not necessarily reflect what one would consider as important. We plan to share our code to make it accessible to everyone. We will do this once the anonymity period is finished.

References

Milan Bhan, Jean-Noel Vittaut, Nicolas Chesneau, and Marie-Jeanne Lesot. 2023. Tigtec : Token importance guided text counterfactuals.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2020. Plug and play language models: A simple approach to controlled text generation. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

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.

Riccardo Guidotti. 2022. Counterfactual explanations and how to find them: literature review and

benchmarking. *Data Mining and Knowledge Discovery*.

Thibault Laugel, Marie-Jeanne Lesot, Christophe Marsala, X. Renard, and Marcin Detyniecki. 2019. The dangers of post-hoc interpretability: Unjustified counterfactual explanations. In *International Joint Conference on Artificial Intelligence*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Scott M Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

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, Srikanta Bedathur, and Diptikalyan Saha. 2022. Plug and Play Counterfactual Text Generation for Model Robustness. ArXiv:2206.10429 [cs].

Raphael Mazzine and David Martens. 2021. A Framework and Benchmarking Study for Counterfactual Generating Methods on Tabular Data. *arXiv:2107.04680 [cs]*. ArXiv: 2107.04680.

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

Christoph Molnar, Gunnar König, Julia Herbinger, Timo Freiesleben, Susanne Dandl, Christian A. Scholbeck, Giuseppe Casalicchio, Moritz Grosse-Wentrup, and Bernd Bischl. 2021. General Pitfalls of Model-Agnostic Interpretation Methods for Machine Learning Models. ArXiv:2007.04131 [cs, stat].

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 *Conference on Empirical Methods in Natural Language Processing*.

Rafael Poyiadzi, Kacper Sokol, Raul Santos-Rodriguez, Tijl De Bie, and Peter Flach. 2020. FACE: Feasible and Actionable Counterfactual Explanations. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pages 344–350. ArXiv:1909.09369 [cs, stat].

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. page 24.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1).

Marco Tulio 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*, pages 1135–1144.

Alexis Ross, Ana Marasović, and Matthew Peters. 2021. Explaining NLP models via minimal contrastive editing (MiCE). In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3840–3852, Online. Association for Computational Linguistics.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. ArXiv:1910.01108 [cs].

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 3319–3328. JMLR.org.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Tongshuang Wu, Marco Tulio Ribeiro, Jeffrey Heer, and Daniel Weld. 2021. Polyjuice: Generating counterfactuals for explaining, evaluating, and improving models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6707–6723, Online. Association for Computational Linguistics.

Descriptive statistics	IMDB	Movie genre
Avg. tokens	57.4	69.71
DistilBERT acc. %	90.1	88.3

Table 3: Data sets descriptive statistics and classifiers performance

A Appendices

A.1 Dataset and classifiers

We apply CFI on two sets of counterfactual examples from two different binary classifiers. The first classifier has been trained to perform sentiment analysis on the IMDB database. The second classifier has been trained on a dataset coming from a Kaggle competition to classify movie genres.

Each DistilBERT is initialized as a DistilBERT base uncased from Hugging Face on PyTorch. The text preparation and tokenization step is performed via Hugging Face's DistilBERT tokenizer. The forward path is defined as getting the embedding of the classification token to perform the classification task. A dense layer is added to perform the classification and fine-tune the models. Each classifier has therefore 66 million parameters and is trained with 3 epochs, with a batch size of 12. The loss for the training is a CrossEntropyLoss, and the optimization is done using Adam with initial learning rate of $5e-5$ and a default epsilon value to $1e-8$. The performances of the classifiers are presented in Table 3.

A.2 TIGTEC hyperparameters

We follow here the notations from the original paper.

- $g = attention$
- $\mathcal{M} = \mathcal{M}_{ft}$ where $\mathcal{M}_{ft}$ is a BERT mask language model fine-tuned on the corpus in which the classifier f has been trained.
- $\alpha = 0.3$
- $\texttt{topk} = 50$
- $\texttt{beam\_width} = 4$
- $\texttt{mask\_div} = 4$
- $\texttt{strategy} = \texttt{evolutive}$
- $\texttt{margin} = 0.15$
- $s = \texttt{sentence\_transformer}$

Figure 5: Top 35 important symmetric token pairs relatively to TIGTEC-CGI and movie genre classification. The left side corresponds to tokens associated with comedy synopses, while the right column corresponds to horror ones. The stronger the link between two tokens, the higher the importance of the pair.