TIGTEC : Token Importance Guided TExt Counterfactuals

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

TIGTEC : Token Importance Guided TExt Counterfactuals

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

[1] Ekimetrics, Paris, France
[2] Sorbonne Université, CNRS, LIP6, F-75005 Paris, France

Abstract. Counterfactual examples explain a prediction by highlighting changes in an instance that flip the outcome of a classifier. This paper proposes TIGTEC, an efficient and modular method for generating sparse, plausible and diverse counterfactual explanations for textual data. TIGTEC is a text editing heuristic that targets and modifies words with high contribution using local feature importance. A new attention-based local feature importance is proposed. Counterfactual candidates are generated and assessed with a cost function integrating a semantic distance, while the solution space is efficiently explored in a beam search fashion. The conducted experiments show the relevance of TIGTEC in terms of success rate, sparsity, diversity and plausibility. This method can be used in both model-specific or model-agnostic way, which makes it very convenient for generating counterfactual explanations.

Keywords: XAI · NLP · Counterfactual examples · Local Feature Importance · Attention

1 Introduction

The high level of performance in the field of natural language processing (NLP) achieved by Transformer models [30] comes along with complex architectures. The domain of eXplainable Artificial Intelligence (XAI) aims at understanding and interpreting the predictions made by such complex systems [18]. Among the main categories of XAI methods to explain the prediction of a given instance, local feature importance [3] quantifies the impact of each feature on the considered outcome. Another family of XAI methods consists in explaining with counterfactual examples (see [9] for a recent survey), defined as instances close to the instance of interest but associated with another prediction.

This paper proposes a new method to generate counterfactual explanations in the case of textual data, called Token Importance Guided TExt Counterfactuals (TIGTEC). For example, given a classifier that predicts film synopsis genre and an instance of interest predicted to be a comedy, TIGTEC outputs several slightly modified instances predicted to be horror synopses (see Figure 1).

The main contributions of TIGTEC are as follows: (*i*) textual counterfactual examples are generated by masking and replacing important words using local feature importance information, (*ii*) a new model-specific local feature importance method based on attention mechanisms [2] from Transformers is proposed, (*iii*) a new cost function integrating

Fig. 1. Example of *sparse*, *plausible* and *diverse* counterfactual examples generated by TIGTEC for a film genre classifier that discriminates between horror and comedy synopses. Here, the counterfactual generation goes from comedy to horror.

textual semantic distance to preserve the initial content is introduced, (*iv*) the solution space is explored with a new tree search policy based on beam search that leads to diversity in the generated explanations. In this manner, TIGTEC bridges the gap between local feature importance, mask language models, sentence embedding and counterfactual explanations. TIGTEC can be applied to any NLP classifier in a model-specific or model-agnostic fashion, depending on the local feature importance method employed.

This paper is organized as follows: we first introduce some basic principles of XAI and the related work in Section 2. The architecture of TIGTEC is presented in Section 3. Section 4 describes the performed experimental study and compare TIGTEC to a competitor. Finally Section 5 concludes this paper by discussing the results and future work.

2 Background and Related Work

We recall here some basic principles of XAI methods and existing counterfactual generation methods in NLP.

2.1 XAI Background

Local Feature Importance. Let $f : \mathcal{X} \to \mathcal{Y}$ be a NLP classifier mapping an input space $\mathcal{X}$ to an output space $\mathcal{Y}$. Let $x_0 = [t_1, ..., t_{|x_0|}] \in \mathcal{X}$ be a sequence of interest with $f(x_0) = y_0$. A local feature importance (or *token importance* in NLP) operator $g : \mathcal{X} \to \mathbb{R}^{|x_0|}$ explains the prediction through a vector $[z_1, ..., z_{|x_0|}]$ where z_i is the contribution of the $i-$th token. Two common local feature importance methods in NLP are Local Interpretable Model-agnostic Explanations (LIME) [26] and SHapley Additive eXplanations (SHAP) [13]. These methods have the advantage of being model-agnostic since they can be used without any information about the model to explain.

Counterfactual Explanation Counterfactual explanations aim to 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 [17]. 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 and is basically defined as:

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

with $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ a given distance operator measuring proximity. The counterfactual explanation is then the difference between the generated counterfactual example and the initial data point, $x^{\text{cf}} - x_0$.

Many additional desirable properties for counterfactual explanations have been proposed [9,16] to ensure their informative nature that we summarize in three categories. *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 [11] while being feasible [22]. Since several instances of explanation can be more informative than a single one [28,20], *diversity* measures the extent to which the counterfactual examples differ from each other.

2.2 Related Work

This section presents two categories of methods for generating textual counterfactual examples.

Text Editing Heuristics. A first family of methods aims at addressing 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 [8] focuses on the embedding space of the classifier to explain. After generating counterfactual candidates through optimization in the latent space, the most valuable ones are selected according to an estimation of Shapley values. MiCE [27] iteratively masks parts of the initial text and performs span infilling using a T5 [24] fine-tuned on the corpus of interest. This method targets tokens with high predictive power using model-specific gradient attribution metrics. While the label flipping success rate of CLOSS and MiCE are high and the counterfactual texts are *plausible*, the notions of *semantic distance* and *diversity* are not addressed. We show in Section 3 how the TIGTEC approach that we propose tackles these constraints.

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 [19]. Contrary to counterfactual explanations, adversarial attacks seek to fool intentionally a model without explanatory purpose. Therefore, *plausibility* and *sparsity* are not addressed.

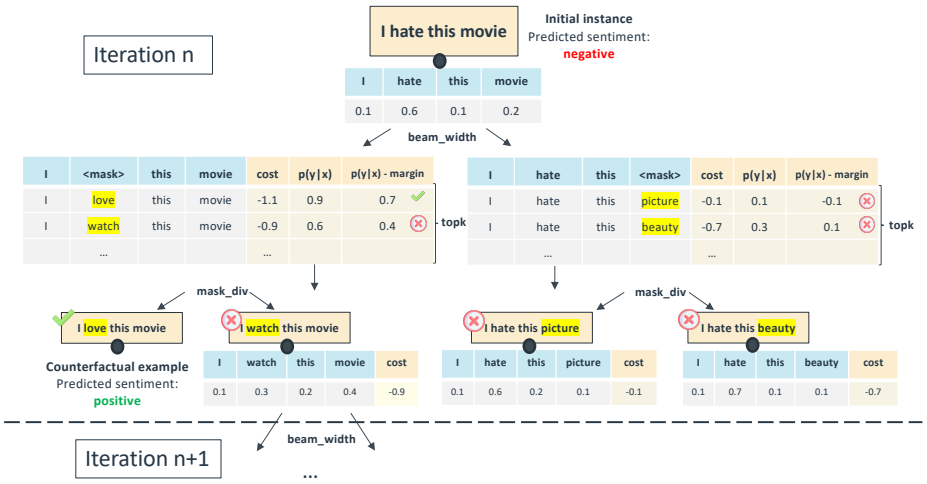

Fig. 2. Illustration of the tree search policy with `beam_width` = 2, `mask_div` = 2, `strategy` = evolutive, `margin` = 0.2. At each step, the `beam_width` highest important tokens are masked and replaced. The substitution token is selected considering the cost function depending on the semantic similarity method s and the balancing parameter α. Among the `topk` candidates, only `mask_div` one are considered in the tree search. A candidate is accepted if the prediction of the classifier changes and moves `margin` away from the prediction threshold. Here, "I love this movie" is accepted. Since only one counterfactual candidate was found out of two, the next iteration starts from the nodes with the lowest cost value, here "I watch this movie".

Text Generation with Large Language Models. A second category of methods aims at generating counterfactual examples in NLP with large pre-trained *generative language models*. A first approach [15] applies a Plug and Play language model [6] 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 [12] in order to flip the outcome. Polyjuice [31] proposes to fine-tune a GPT-2 [23] 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. Therefore, the use of Polyjuice to generate counterfactual examples is not generalizable since counterfactual labels do not exist for all classification problems.

3 Proposed approach: TIGTEC

This section describes the architecture of Token Importance Guided TExt Counterfactuals (TIGTEC) by detailing its four components. The main idea is to iteratively change tokens of the initial text by decreasing order of importance instance to find a compromise between proximity to the initial instance and label flipping. This way, TIGTEC belongs to the *text editing heuristics* category of counterfactual example generators in NLP.

3.1 TIGTEC Overview

TIGTEC is a 4-step iterative method illustrated in Figure 2. Algorithm 1 describes the generation and evaluation steps, Algorithm 2 summarizes the whole process. The code is available online on a public repository[3]. TIGTEC takes as input a classifier f and a text of interest $x_0 = [t_1, ..., t_{|x_0|}]$.

Targeting. To modify the initial text to explain, tokens with highest impact on prediction are targeted given their local importance. TIGTEC implements two methods of local token importance and a random importance generator as a baseline.

Generating. High importance tokens are masked and replaced, with a fine-tuned or pretrained mask model. Various counterfactual candidates are then generated.

Evaluating. The generated candidates are evaluated by a cost function that balances the probability score of the target class and the semantic distance to the initial instance. Candidates minimizing the cost function are considered valid if they meet acceptability criteria.

Tree search policy. The lowest cost candidates are kept in memory and a new iteration begins from the most promising one. The solution space is explored in a beam search fashion until a stopping condition is reached.

As outlined in Figure 2, the counterfactual search heuristic is a tree search algorithm, in which each node corresponds to a counterfactual candidate, and each edge is a token replacement. Therefore, the root of the tree corresponds to the instance to explain, and the deeper a node is in the tree, the more it is modified.

3.2 Targeting

The first step consists in identifying the most promising tokens to be replaced in the initial instance to modify the outcome of the classifier f. We use token importance metrics to focus on impacting tokens and efficiently guide the search for counterfactual examples. In particular, we integrate the possibility of computing both model-agnostic (e.g. SHAP [13]) and model-specific token importance metrics. We propose a new model-specific token-importance method based on the attention coefficients when the classifier f is a Transformer. Token importance is computed by focusing on the attention of the last encoder layer related to the classification token representing the context of the entire sequence. The efficiency gain of this token importance method is shown in Section 4. If the information provided by SHAP is rich, its computation time is high, whereas attention coefficients are available at no cost under a *model-specific* paradigm.

TIGTEC is also defined by its `strategy` which can take two values. The static `strategy` consists in fixing the token importance coefficients for the whole search, whereas the evolutive `strategy` recomputes token importance at each iteration. Since SHAP has a high computational cost, it is not recommended to combine it with the evolutive `strategy`.

In order to consider several counterfactual candidates at each iteration, several tokens can be targeted in parallel. The `beam_width` parameter allows to control the number of tokens of highest importance to target at each step to perform a beam search during the space exploration.

[3] https://github.com/milanbhan/tigtec

Algorithm 1 Mask Language Inference (MLI)

Require: $x = [t_1, ..., t_n]$ an input sequence
Require: $f : \mathcal{X} \rightarrow \mathcal{Y}$ = {1,2,...,k} a classifier
Require: i the input token to be masked
Require: $\mathcal{M}$ a BERT-like mask language model
Require: s, α, `topk`, `mask_div`
Ensure: $\hat{x}$ = $[\hat{x}_{(1)}, ..., \hat{x}_{(\text{mask\_div})}]$
1: $t_i \leftarrow$ [MASK]
2: $x_{\text{mask}} \leftarrow [t_1, ..., [\text{MASK}], ..., t_n]$
3: $[\hat{t}_1, ..., \hat{t}_{\text{topk}}] = \mathcal{M}(x_{\text{mask}})$ the `topk` most likely tokens
4: **for** j in {1,...,`topk`} **do**
5: $\quad \hat{x}_j = x[t_i \leftarrow \hat{t}_j]$
6: $\quad$ Compute $\texttt{cost}(\hat{x}_j)$ see Eq. 4
7: **end for**
8: Retrieve in $\hat{x}$ the `mask_div` sequences with lowest cost
9: **return** $\hat{x}$

3.3 Generating

The second step of TIGTEC generates counterfactual candidates and corresponds to the first part of the mask language inference (MLI) formally described in Algorithm 1, from line 1 to 5. Once high importance tokens have been targeted in the previous step, they are masked and replaced with a BERT-type [7] mask language model denoted $\mathcal{M}$. Mask language models enable to replace tokens considering the context while keeping grammatical correctness and semantic relevance. This step ensures the plausibility of the generated text. Such models take a masked sequence $[t_1, ..., [\text{MASK}], ..., t_n]$ as an input and output a probability score distribution of all the tokens contained in the BERT-type vocabulary. The mask model can be either pretrained or fine-tuned on the text corpus on which the classifier f has been trained.

Since replacing a token with another with low plausibility can lead to out-of-distribution texts, inaccurate prediction and grammatical errors, the number of substitutes proposed by $\mathcal{M}$ is limited to `topk`. The higher `topk`, the more we consider tokens with low contextual plausibility.

3.4 Evaluating

Once the `topk` candidates are generated, we build a cost function to evaluate them. This evaluation step corresponds to line 6 in Algorithm 1. The cost function has to integrate the need to flip the outcome of the classifier f and the distance to the original instance as formalized in Eq. 1. In order to ensure semantic relevance, we define a distance based on text embedding and cosine similarity measures. Finally, conditions for the acceptability of counterfactual candidates are introduced to ensure the reliability of the explanations.

Distance. The widely used Levenshtein distance and BLEU score [21] do not integrate the notion of semantics. An alternative is to compare sentence embeddings in order to

Algorithm 2 TIGTEC: Token Importance Guided Counterfactual Text Generation

Require: $f : \mathcal{X} \to \mathcal{Y}$ a k-class classifier
Require: $x_0 = [t_1, ..., t_n]$ an input sequence of n tokens to be explained
Require: y_{target} : target counterfactual class
Require: p : number of counterfactual examples to generate
Require: g, s, $\mathcal{M}$, α, `topk`, `beam_width`, `mask_div`, `strategy`, `margin`, *early_stop*
Ensure: $x^{\text{cf}} = [x_1^{\text{cf}}, ..., x_p^{\text{cf}}]$

`waiting_list` = $[(x_0, \texttt{cost}(x_0))]$ the priority queue of counterfactual candidates sorted by increasing `cost` (see Eq. 4)
$i \leftarrow 0$ the number of evaluated texts
$x^{\text{cf}} \leftarrow []$
Compute token importance $[z_1, ..., z_n] = g(x_0)$
while $len(x^{\text{cf}}) < p$ and $i <$ *early_stop* **do**
  `parent_node` $\leftarrow$ `waiting_list`.pop() the candidate with the lowest cost (see Eq. 4)
  $[t_{(1)}, ..., t_{(n)}] \leftarrow$ `sort`(`parent_node`) by decreasing importance order with respect to `strategy` and g
  for t in $[t_{(1)}, ..., t_{(\text{beam\_width})}]$ **do**
    $i \leftarrow i + 1$
    $[x_1, ..., x_{\text{mask\_div}}]$ = MLI(`parent_node` , f, t, $\mathcal{M}$, `topk`, `mask_div`, s α) (see Algorithm 1)
    for x in $[x_1, ..., x_{\text{mask\_div}}]$ **do**
      if $p(y_{\text{target}}|x) \geq \frac{1}{k} +$ `margin` **then**
        x^{cf}.append(x)
      else
        `waiting_list`.push$((x, \texttt{cost}(x)))$ keep in the waiting list rejected candidates with their `cost`
      end if
    end for
  end for
end while
return x^{cf}

measure the similarity of representations in a latent space. Sentence embeddings have been introduced to numerically represent textual data as real-value vectors, including Sentence Transformers [25]. Such networks have been trained on large corpus of text covering various topics. This encoders are compatible with a model-agnostic approach, as they do not require any prior information about the classifier f.

Another text embedding approach can be used when the classifier f is a BERT-like model and when the prediction is made through the classification token. It consists in using the embedding of the classification token directly from f. This embedding is however strongly related to the task of the classifier f. Therefore, if the model has been trained for sentiment analysis, two texts with the same associated sentiment will be considered similar, regardless of the topics covered.

We derive the textual distance from the normalized scalar product of the two embeddings: $d : \mathcal{X} \times \mathcal{X} \to [0, 1]$ with:

$$d_s(x, x') = \frac{1}{2}(1 - s(x, x')) \tag{2}$$

$$s(x, x') = \frac{\langle e_x, e'_x \rangle}{||e_x||.||e'_x||} \tag{3}$$

where e_x is the embedding representation of input sequence x.

Cost. The cost function aims to represent the counterfactual optimization problem introduced in Eq. 1. We propose to integrate the probability score of the target class to define the cost as:

$$\texttt{cost}(x^{\text{cf}}, x_0) = -\left(p(y_{\text{target}}|x^{\text{cf}}) - \alpha d_s(x^{\text{cf}}, x_0)\right) \tag{4}$$

where y_{target} is the target class and $p(y_{\text{target}}|x^{\text{cf}})$ represents the probability score of belonging to the class y_{target} given x^{cf} from the classifier *f*. The probability score is the information that guides the heuristic towards the target class. The α coefficient enables for a balanced approach to the need to reach the target class while remaining close to the initial point. The generated `topk` candidates are evaluated with the cost function defined above.

Acceptability Criteria. A counterfactual candidate x^{cf} is accepted if two conditions are met:

$$f(x^{\text{cf}}) = y_{\text{target}} \tag{5}$$

$$p(y_{\text{target}}|x^{\text{cf}}) \geq \frac{1}{k} + \texttt{margin} \tag{6}$$

where k is the number of classes of the output space and $\texttt{margin} \in [0, \frac{k-1}{k}]$ the regularization hyperparameter ensuring the certainty of the prediction of the model *f*. We assume then that all the counterfactual examples must reach the same target class. The closer `margin` is to its upper bound, the more polarized the classifier prediction must be in order to satisfy the acceptability criterion, and the stronger the constraint.

3.5 Tree Search Policy

TIGTEC generates a set of diverse counterfactual examples. We address the diversity constraint by considering the `mask_div` candidates with the lowest cost function among the generated `topk` from Algorithm 1 and keep them in memory in a priority queue (see line 15 in Algorithm 2). Therefore, we evaluate more possibilities and aim to foster diversity in the counterfactual examples found by TIGTEC. Once these candidates are stored in memory, the iterative exploration step (Algorithm 2 from line 6 to 11) starts again, until a stopping condition is reached. The stopping condition can either be to reach the target number of counterfactual examples or to reach the maximum number of nodes in the tree (see line 5 in Algorithm 2). The higher the maximum number of nodes, the longer TIGTEC can search for counterfactual examples.

The candidate with the lowest cost is then selected from the priority queue (see line 6 in Algorithm 2) in order to apply again the targeting, generation and evaluation sequence. We call predecessor this previous candidate. Since we evaluate several possibilities in parallel through beam search, Algorithm 1 is this time applied to the `beam_width` tokens

with the highest token importance within the predecessor. From this perspective, the exploration approach enables to start from a candidate that seemed less advantageous at a specific stage, but leads to better results by going deeper into the tree. A tree search example is illustrated in Figure 2.

4 Experimental analysis

This section presents the conducted experimental study and introduces five metrics to quantitatively assess the counterfactual examples generated by two different versions of TIGTEC and three comparable state-of-the-art competitors.

4.1 Evaluation Criteria

Considering the various objectives to be achieved, we propose a 5-metric evaluation. Given an instance associated with p counterfactual examples, the evaluation metrics are aggregated on average over the generated examples, except for diversity. The same operation is performed on all the instances to be explained, and the average metrics are finally computed.

Success Rate. Since TIGTEC does not guarantee to find counterfactual examples in all cases, the success rate (**%S**) is calculated.

Sparsity. For some methods we compare to, the lengths of the generated counterfactual examples may differ from the initial instance. Therefore, sparsity (**%T**) is measured assessed with word-based Levenshtein distance normalized by the length of the sequence.

Proximity. We evaluate *ex-post* the semantic proximity between x_0 and x_{cf} with cosine similarity (**s**) between Sentence Transformer embedding. This choice is justified by the wish to remain in a general framework that does not depend on the classifier f and the task for which it has been trained. The library used to import the Sentence Transformer is `sentence_transformers` and the model backbone is `paraphrase-MiniLM-L6-v2`.

Plausibility. One approach to evaluate text plausibility is the perplexity score [10]. This score can be computed based on the exponential average loss of a foundation model like GPT-2. We calculate the ratio (Δ**PPL**) between the perplexity of the initial text and its counterfactual examples to compare the quality of the generated text with the original one. The library used to import the pretrained GPT2 is `transformers` and the backbone is `GPT2LMHeadModel`.

Diversity. Based on the distance measure d, we define diversity (**div**) as in [20] where $div_d = det(K)$ with $K_{i,j} = \frac{1}{\lambda + d(x_i^{\text{cf}}, x_j^{\text{cf}})}$ and $\lambda \in \mathbb{R}$ a regularization weight set to 1.

4.2 TIGTEC Agnostic and Specific Variants

Two different versions of TIGTEC are assessed. The first one is model-specific with access to the corpus of interest. Attention coefficients guide the counterfactual example search and a fine-tuned mask language model is used to mask and replace important tokens. We call this version TIGTEC-specific.

The second version is model-agnostic without access to the corpus of interest. SHAP is used to compute token importance and the mask language model is only pre-trained. We call this second version TIGTEC-agnostic. Since SHAP computational cost is high compared to attention, we use the static `strategy` for the *agnostic* version of TIGTEC, whereas the evolutive `strategy` is used for the *specific* one.

4.3 Datasets and Competitors

We apply TIGTEC-agnostic and -specific on two DistilBERT [29] binary classifiers. We limit our analysis to DistilBERT, since it achieves almost the same level of performance as BERT, while being significantly lighter. TIGTEC could, however, be applied to larger models, the methodology remaining the same. The first classifier performs sentiment analysis on the IMDB dataset [14] containing movie reviews. The second classifier is trained on movie genre classification on a dataset of horror and comedy synopses from Kaggle[4]. More information about the datasets and the performance of the classifiers are provided online[5].

The two versions of TIGTEC are compared to Polyjuice [31], MiCE [27] and CLOSS [8]. The objective of each version of TIGTEC is to generate three counterfactual examples associated with an initial instance. We apply Polyjuice by generating three counterfactual examples for each instance to explain. As Polyjuice was trained to flip sentiment on IMDB with negation prompt, Polyjuice's counterfactual examples are generated in the same way. MiCE and CLOSS do not address diversity, they only generate one counterfactual example per initial text. We assess TIGTEC and Polyjuice performance by selecting the instance that is semantically closest to the initial point among the 3 generated to compare them to MiCE and CLOSS. We distinguish the results obtained with one and three counterfactual examples by the notation TIGTEC$_{1d}$ and TIGTEC$_{3d}$ and respectively Polyjuice$_{1d}$ and Polyjuice$_{3d}$.

Each method is evaluated on the same 1000 texts from IMDB. The hyperparameters of TIGTEC are fixed at their optimal level as described in the next section. TIGTEC-specific is also applied on the movie synopsis dataset from Kaggle on 474 texts. Since movie genre classification is a more complex task, we relax the hyperparameters by lowering the margin to 0.05 and alpha to 0.15.

4.4 Hyperparameter Setting

We optimize the nine hyperparameters presented in Section 3 with respect to success rate, similarity, diversity and sparsity. The optimization is performed on IMDB with the Optuna [1] library. The solution space is as follows:

- $g \in \{random, attention\}$, the input token importance method.
- $\mathcal{M} \in \{\mathcal{M}_{ft}, \mathcal{M}_{pt}\}$ where $\mathcal{M}_{ft}$ is a mask language model fine-tuned on the corpus in which the classifier f has been trained. $\mathcal{M}_{pt}$ is a pretrained mask language model without fine tuning phase.

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

[5] See the documentation on the publicly available repository : https://github.com/milanbhan/tigtec

Dataset	Method	Success rate ↑%S	Similarity ↑%s	Sparsity ↓%T	Plausibility ↓ ΔPPL	Diversity ↑div
IMDB	Polyjuice$_{1d}$	60.8	55.6	72.2	**1.09**	-
	Polyjuice$_{3d}$	29.6	53.5	74.4	2.16	0.088
	MiCE	**99.6**	81.1	18.0	1.35	-
	CLOSS	97.3	95.4	**2.3**	1.47	-
	TIGTEC-specific$_{1d}$	98.2	**96.8**	4.2	**1.25**	-
	TIGTEC-specific$_{3d}$	98.2	**95.8**	4.4	1.34	0.019
	TIGTEC-agnostic$_{1d}$	92.7	96.1	4.5	**1.24**	-
	TIGTEC-agnostic$_{3d}$	92.7	94.6	4.7	1.34	0.075
Movie genre	TIGTEC-specific$_{1d}$	88.4	91.7	8.8	1.42	-
	TIGTEC-specific$_{3d}$	88.4	89.8	9.0	1.38	0.120

Table 1. TIGTEC evaluation on 2 datasets and comparison with competitors on IMDB.

- $\alpha \in [0, 1]$ the parameter balancing target probability and distance with the initial point in the cost function
- `topk` $\in \{10, 11, ..., 100\}$ the number of candidates considered during mask inference
- `beam_width` $\in \{2, 3, ..., 6\}$ the number of paths explored in parallel at each iteration
- `mask_div` $\in \{1, 2, 3, ..., 4\}$ the number of candidates kept in memory during a tree search iteration
- `strategy` $\in \{$*static*, *evolutive*$\}$ where *static* is the strategy consisting in computing token importance only at the beginning of the counterfactual search. The *evolutive* strategy consists in computing token importance at each iteration.
- `margin` $\in \{0.05, 0.3\}$ the probability score spread defining the acceptability threshold of a counterfactual candidate
- $s \in \{$*sentence_transformer*, *CLS_embedding}* the method used to compute the semantic distance.

We perform the optimization over 100 iterations, with the objective to generate 3 counterfactual examples on 20 initial texts. An ablation study thoroughly analyzes the sensibility to TIGTEC to its hyperparameters. For the other hyperparameters, `beam_width` = 4, `mask_div` = 4, `topk` = 50, `margin` = 0.15 and $\alpha = 0.3$ and Sentence Transformer embedding are reasonable. The maximum number of nodes is set to 1000, which can lead to long searches for counterfactual examples before TIGTEC stops.

4.5 Results

Global Results. Overall, TIGTEC-specific gives very good results on IMDB, succeeding in more than 98% of the time in generating counterfactual examples (Table 1). The counterfactual examples are sparse, plausible and highly similar to their original instance. TIGTEC-agnostic succeeds less than the specific version, with a success rate at circa 93%. Similarity, sparsity and plausibility are at the same level as the specific version, while the counterfactual examples are more diverse. The significant gap in success rates between the agnostic and the specific versions of TIGTEC can be explained by the cumulative effect of the evolutive `strategy` and the fine-tuned mask model compared to the static

Hyperparameter		Success rate% mean ± std	Similarity% mean ± std	Sparsity% mean ± std
Token importance	random (ref.)	92.0 ± 14.0	91.4 ± 3.5	9.4 ± 3.0
	attention	**96.2*** ± 7.0	**95.0***** ± 1.7	**4.2***** ± 1.1
	SHAP	**95.6*** ± 7.2	**95.0***** ± 1.5	**4.4***** ± 1.4
Exploration strategy	static (ref.)	93.6 ± 11.4	94.2 ± 2.9	5.9 ± 2.9
	evolutive	95.4 ± 8.5	93.7 ± 2.9	5.8 ± 3.1
Mask model	pretrained (ref.)	94.6 ±10.5	93.3 ± 3.5	6.0 ± 3.5
	fine-tuned	94.8 ± 9.2	**94.4**** ± 2.1	5.6 ± 2.6

Table 2. Ablation study of token importance, exploration `strategy` and mask model. With p as the p-value of the one-tailed t-test, $^*p < 10\%$, $^{**}p < 5\%$, $^{***}p < 1\%$. Ref stands for the reference modality.

`strategy` and the pretrained mask model. We detail these effects separately in the following ablation study. While the movie genre classification task is more complex (see online[6] for classifier accuracy), TIGTEC manages to generate plausible counterfactual examples close to the initial instance, with more diversity compared to the sentiment analysis task.

Comparative Results. TIGTEC-specific succeeds more often than CLOSS and Polyjuice, while remaining on average closer to the initial instance and being more plausible. The success rate of Polyjuice is low, and the counterfactual examples differ from the original instances in terms of proximity and sparsity. This result is due to the absence of label switching constraint and the independence of the text generation process to the classifier.

MiCE succeeds more often to flip labels than any other counterfactual generator. While the text generated by MiCE is plausible, the counterfactual examples differ strongly from the original instances in terms of semantic proximity and sparsity. TIGTEC-specific succeeds in the same proportion as MiCE and produces much more sparse, similar and plausible counterfactual examples. The low similarity of the counterfactual examples generated by MiCE can be explained by the underlying T5 model used to generate text. Such encoder-decoder models perform mask span infilling by generating text whose meaning and length can sharply change from the masked text.

TIGTEC-agnostic generates more similar, sparse and plausible counterfactual texts than MiCE and Polyjuice. However, if the success rate of TIGTEC-agnostic is high, it is lower than MiCE and CLOSS. Whether in its agnostic or specific version, and with or without the diversity constraint, TIGTEC performs well on all evaluation metrics. Finally, TIGTEC appears to be the best trade-off in terms of success rate, proximity, sparsity, plausibility and diversity.

Ablation Study. This analysis comes from the data resulting from the hyperparameter optimization. We assess the sensitivity of TIGTEC to its hyperparameters through success rate, similarity and sparsity. Each comparison is made with a one-tailed t-test to

[6] https://github.com/milanbhan/tigtec

Hyperparameter		Success rate % mean ± std	Similarity % mean ± std	Sparsity % mean ± std
beam_width	2 (ref.)	94.4 ±11.3	92.9 ± 3.6	6.6 ± 3.4
	3	96.6 ± 7.4	93.8 ± 2.7	5.8 ± 3.3
	4	96.0 ±7.8	**94.5****± 1.9	**4.5*** ± 1.4
	5	90.2 ± 12.5	**95.1**** ± 1.7	**5.6**** ± 2.7
	6	95.7 ± 6.5	**95**** ± 1.6	**5.1**** ± 2.6
mask_div	1 (ref.)	97.0 ± 7.6	93.2 ± 3.1	6.7 ± 3.3
	2	94.7 ± 10.7	**94.3*** ± 2.0	**4.9*** ± 2.0
	3	90.6 ± 12.2	**94.6*** ± 1.9	**5.5**** ± 3.0
	4	93.3 ± 9.0	**94.4*** ± 3.9	**5.3*** ± 3.3

Table 3. Ablation study of `beam_width` and `mask_div`. With p as the p-value of the one-tailed t-test, $*p < 10\%$, $**p < 5\%$, $***p < 1\%$. Ref stands for the reference modality.

determine whether the mean of a first sample is lower than the mean of a second one. We first evaluate the impact of hyperparameters specific to the targeting and generating steps of TIGTEC in Table 2. We compare the attention-based token importance and SHAP to a random baseline. The evolutive exploration `strategy` is compared to the static one and the contribution of the fine-tuned mask model is assessed with respect to the pretrained one. Attention-based token importance and SHAP give better results both in terms of success rate, similarity and sparsity with statistical significance. The fine-tuned mask model induces higher similarity with statistical significance. While the evolutive `strategy` yields higher success rates on average, the results are not statistically significant.

Besides, we focus on the hyperparameters specific to the exploration and tree search step. The results for the `beam_width` and `mask_div` hyperparameters are presented in Table 3. Each beam width is compared to the reference case where `beam_width`$= 2$. Mask diversity is also analyzed with respect to the reference case where `mask_div`$= 1$. The higher `beam_width` and `mask_div`, the higher the similarity and sparsity. This results are statistically significant.

5 Discussion

We have introduced TIGTEC, an efficient textual counterfactual explainer, generating sparse, plausible, content-preserving and diverse counterfactual examples in an *agnostic* or *specific* fashion. Other NLP counterfactual generators strongly depend on the classifier to explain or the text corpus on which it has been trained. As matter of fact, CLOSS [8] generates counterfactual candidates by optimizing in the latent space from the classifier. MiCE [27] uses gradient-based information from the classifier to target important tokens, while modifying the initial instance with a language model fine-tuned on the corpus of interest. Polyjuice [31] needs to learn to generate counterfactual examples in a supervised way, which requires ground-truth counterfactual data. The adaptability of TIGTEC to any type of NLP classifier and the fact that it works in an *agnostic* way make it particularly flexible.

The proposed framework is versatile and can work with any token importance method. Since the high computational cost of SHAP can be limiting for large-scale applications, other methods such as gradient-based attributions can be used. Besides, the token importance sensitivity analysis highlighted that attention drives TIGTEC as well as SHAP in the search process in terms of success rate, similarity and sparsity. This study therefore favors the interpretabiltiy of self-attention as other recent work [5] [4]. If the experimental study has been performed on binary classifiers only, TIGTEC can also be extended to multi-class classifiers by specifiny the target class.

Finally, the use of TIGTEC is not limited to BERT-like classifiers. Our proposed framework can be adapted to any type of classifier as long as a token importance method is given as input. For other NLP classifiers such as recurrent neural networks, SHAP or gradient-based methods could be used to target impactful tokens. TIGTEC can also help in explaining machine learning models such as boosted trees with LIME as token importance method.

6 Conclusion and Future Work

This paper presents TIGTEC, a method for generating sparse, plausible and diverse counterfactual explanations. The architecture of TIGTEC is modular and can be adapted to any type of NLP model and to classification tasks of various difficulties. TIGTEC can cover both model-agnostic and model-specific cases, depending on the token importance method used to guide the search for counterfactual examples.

A way of improvement of TIGTEC could be to cover more types of classifiers as mentioned in the previous section. Other gradient-based token importance methods could also be integrated to TIGTEC. Furthermore, diversity is only implicitly addressed through the exploration `strategy`. We believe that diversity could be improved by transcribing it into the cost function during the evaluation step or sharpening the exploration `strategy`.

Finally, automatic evaluation of the counterfactual examples quality has its limits. The metrics introduced above provide good indications of the performance of TIGTEC, but they do not ensure human understanding. From this perspective, human-grounded experiments would be more appropriate to assess the relevance of the generated text and its explanatory quality.

Ethics Statement

Since the training data for mask language models, Sentence Transformers and classifiers can be biased, there is a risk of generating harmful counterfactual examples. One using TIGTEC to explain the predictions of one's classifier must be aware of these biases in order to stand back and analyze the produced results. On the other hand, by generating unexpected counterfactual examples, we believe that TIGTEC can be useful in detecting bias in the classifier it seeks to explain. Finally, as any method based on deep learning, this method consumes energy, potentially emitting greenhouse gases. It must be used with caution.

References

1. Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M.: Optuna: A Next-generation Hyperparameter Optimization Framework. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (2019)
2. Bahdanau, D., Cho, K., Bengio, Y.: Neural Machine Translation by Jointly Learning to Align and Translate. ArXiv **1409** (2014)
3. Barredo Arrieta, A., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., Garcia, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R., Herrera, F.: Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. Information Fusion **58**, 82–115 (2020)
4. Bhan, M., Achache, N., Legrand, V., Blangero, A., Chesneau, N.: Evaluating self-attention interpretability through human-grounded experimental protocol. arXiv (2023)
5. Bibal, A., Cardon, R., Alfter, D., Wilkens, R., Wang, X., Francois, T., Watrin, P.: Is attention explanation? an introduction to the debate. In: Proc. of the Association for Computational Linguistics (ACL) (2022)
6. Dathathri, S., Madotto, A., Lan, J., Hung, J., Frank, E., Molino, P., Yosinski, J., Liu, R.: Plug and play language models: A simple approach to controlled text generation. In: 8th International Conference on Learning Representations, ICLR (2020)
7. Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: Proc. of the Association for Computational Linguistics (ACL) (2019)
8. Fern, X., Pope, Q.: Text Counterfactuals via Latent Optimization and Shapley-Guided Search. In: Proc. of Conference on Empirical Methods in Natural Language Processing (EMNLP) (2021)
9. Guidotti, R.: Counterfactual explanations and how to find them: literature review and benchmarking. Data Mining and Knowledge Discovery (2022)
10. Jelinek, F., Mercer, R.L., Bahl, L.R., Baker, J.K.: Perplexity—a measure of the difficulty of speech recognition tasks. The Journal of the Acoustical Society of America **62** (1977)
11. Laugel, T., Lesot, M.J., Marsala, C., Renard, X., Detyniecki, M.: The dangers of post-hoc interpretability: Unjustified counterfactual explanations. In: Int. Joint Conf. on Artificial Intelligence (IJCAI (2019)
12. Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., Zettlemoyer, L.: BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In: Proc. of the Association for Computational Linguistics (ACL) (2020)
13. Lundberg, S.M., Lee, S.I.: A Unified Approach to Interpreting Model Predictions. In: Advances in Neural Information Processing Systems. NeurIPS (2017)
14. Maas, A.L., Daly, R.E., Pham, P.T., Huang, D., Ng, A.Y., Potts, C.: Learning Word Vectors for Sentiment Analysis. In: Proc. of the Association for Computational Linguistics (ACL (2011)
15. Madaan, N., Bedathur, S., Saha, D.: Plug and Play Counterfactual Text Generation for Model Robustness. arXiv (2022)
16. Mazzine, R., Martens, D.: A Framework and Benchmarking Study for Counterfactual Generating Methods on Tabular Data. CoRR (2021)
17. Miller, T.: Explanation in artificial intelligence: Insights from the social sciences. Artificial intelligence **267**, 1–38 (2019)
18. Molnar, C.: Interpretable Machine Learning. https://christophm.github.io/interpretable-ml-book, 2 edn. (2022)
19. Morris, J.X., Lifland, E., Yoo, J.Y., Grigsby, J., Jin, D., Qi, Y.: Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in NLP. In: Proc. of Conference on Empirical Methods in Natural Language Processing (EMNLP) (2020)

20. Mothilal, R.K., Sharma, A., Tan, C.: Explaining machine learning classifiers through diverse counterfactual explanations. In: Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency (FAT* (2020)
21. Papineni, K., Roukos, S., Ward, T., Zhu, W.J.: Bleu: a Method for Automatic Evaluation of Machine Translation. In: Proc. of Association for Computational Linguistics (ACL) (2002)
22. Poyiadzi, R., Sokol, K., Santos-Rodriguez, R., De Bie, T., Flach, P.: FACE: Feasible and Actionable Counterfactual Explanations. In: Proc. of the AAAI/ACM Conference on AI, Ethics, and Society (AIES) (2020)
23. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.: Language Models are Unsupervised Multitask Learners. OpenAI blog (2019)
24. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J.: Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of Machine Learning Research **21** (2019)
25. Reimers, N., Gurevych, I.: Sentence-bert: Sentence embeddings using siamese bert-networks. In: Proc. of Empirical Methods in Natural Language Processing (EMNLP) (2019)
26. Ribeiro, M.T., Singh, S., Guestrin, C.: " Why should I trust you?" Explaining the predictions of any classifier. In: Proc. of the 22nd ACM SIGKDD Int. Conference on Knowledge Discovery and Data Mining (2016)
27. Ross, A., Marasović, A., Peters, M.: Explaining NLP models via minimal contrastive editing (MiCE). In: Findings of the Association for Computational Linguistics (ACL) (2021)
28. Russell, C.: Efficient search for diverse coherent explanations. In: Proc. of the Conference on Fairness, Accountability, and Transparency. p. 20–28. FAT* (2019)
29. Sanh, V., Debut, L., Chaumond, J., Wolf, T.: DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter (2020)
30. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in Neural Information Processing Systems (NeurIPS) (2017)
31. Wu, T., Ribeiro, M.T., Heer, J., Weld, D.: Polyjuice: Generating counterfactuals for explaining, evaluating, and improving models. In: Proc. of the Association for Computational Linguistics (ACL) and the Joint Conference on Natural Language Processing (JCNLP) (2021)