A General Search-Based Framework for Generating Textual Counterfactual Explanations

Daniel Gilo, Shaul Markovitch

Department of Computer Science, Technion - Israel Institute of Technology
{danielgilo,shaulm}@cs.technion.ac.il

Abstract

One of the prominent methods for explaining the decision of a machine-learning classifier is by a counterfactual example. Most current algorithms for generating such examples in the textual domain are based on generative language models. Generative models, however, are trained to minimize a specific loss function in order to fulfill certain requirements for the generated texts. Any change in the requirements may necessitate costly retraining, thus potentially limiting their applicability. In this paper, we present a general search-based framework for generating counterfactual explanations in the textual domain. Our framework is model-agnostic, domain-agnostic, anytime, and does not require retraining in order to adapt to changes in the user requirements. We model the task as a search problem in a space where the initial state is the classified text, and the goal state is a text in a given target class. Our framework includes domain-independent modification operators, but can also exploit domain-specific knowledge through specialized operators. The search algorithm attempts to find a text from the target class with minimal user-specified distance from the original classified object.

Introduction

An important feature of automatic decision making systems (DMS) is the ability to provide human-understandable explanations for their decisions. Such explanations can be helpful for ensuring bias-free operation, for debugging the DMS, and for avoiding legal issues (Madaan et al. 2021; Doshi-Velez et al. 2017).

Many DMS are based on models generated by machine-learning algorithms. Some of the algorithms are based on *interpretable* models, such as decision trees. Others, however, are based on *opaque* models, such as deep neural networks or randoms forests, that, due to their complexity, are practically being used as a *black box*, making explanation generation a challenging task.

From research done in the social sciences (Miller 2019), it is apparent that human-generated explanations are, often, contrastive. People do not explain why an event occurred per se, but rather explain why it had occurred compared to some other event which did not occur. Such explanations are called *counterfactual explanations*.

Wachter, Mittelstadt, and Russell (2017) presented the problem of generating counterfactual explanations for black-box classifications as an optimization task, where a counterfactual is an object that is classified differently from the original object by the black-box classifier, but is similar to it according to a chosen distance metric. Various algorithms have been proposed to address different forms of this optimization problem (e.g. Karimi et al. 2021; Mothilal, Sharma, and Tan 2020; Galhotra, Pradhan, and Salimi 2021). The vast majority of them generate the counterfactuals by perturbing the input example in the feature space, yielding a feature vector. Such an approach is not applicable, however, in the textual domain, where objects are usually represented as vectors in some latent embedding spaces which are not comprehensible by humans.

To overcome these difficulties, several works have introduced an alternative approach that is based on generative models, yielding textual counterfactuals that are understandable by humans (e.g. Yang et al. 2021; Wu et al. 2021; Madaan et al. 2021). This approach, however, has a fundamental limitation. Generative models are trained to minimize a specific loss function that is tailored to fulfill certain requirements for the generated texts. The challenge arises when these requirements need to be altered or modified across different use cases. For instance, there might be a shift from the need for visually similar counterfactuals to semantically similar ones, or from a requirement for fast response times to situations where more time is available. To accommodate such changes, generative methods often necessitate costly retraining processes to minimize a different loss function whenever the settings are modified.

In this paper, we present TCE-SEARCH[1], a general search-based framework for generating counterfactual explanation for text classification. Our new method can adapt to a range of user-specified constraints and preferences with no retraining required. These may include the distance function to be minimized, the required confidence threshold for counterfactual classification, and time constraints on the method's operation.

Our method is based on modeling the counterfactual generation task as a search problem. The initial state is the classified text whose classification needs to be explained, and

[1]Textual Counterfactual Explanation using Search

Original Text	Counterfactual
this beautifully animated epic is never dull.	this beautifully animated epic is never **entertaining**.
best punjabi food i've had in the north american continent	**worst** punjabi food i've had in the north american continent

Table 1: Examples of TCE-SEARCH counterfactuals for a sentiment classification task.

the goal state is a counterfactual example. We use a fine-tuned masked language model as well as a novel word replacement operator to replace parts of the text and explore the object space. Note that the fine-tuning process is performed only once, and changing the user requirements does not necessitate to fine-tune again. We also allow the use of text modification operators that are specific for particular problem types. The user requirements from the generated text are encapsulated into a given distance function. Our algorithm searches for text from the user-defined target class with minimal distance from the original classified object. To accelerate the search, we introduce a heuristic function that is based on the confidence of the classification model. Table 1 presents two simple examples for counterfactuals generated by TCE-SEARCH for sentiment classification tasks. Additional examples can be found in the appendix[2].

There are five important features that characterize our approach:

1. It is *model-agnostic*. While many existing approaches assume a specific learning model, such as neural networks, our algorithm is independent of the type of model used.
2. It is *domain-independent* and can work for any text classification task using general modification operators, but also allows to exploit domain-specific knowledge, through specialized operators, for improved performance.
3. It can accommodate dynamic requirements on the generated text, such as the similarity criterion to the original text, without retraining.
4. It can accommodate dynamic requirements on the generation process without retraining. Specifically, it is an *anytime* algorithm (Zilberstein 1996), meaning it can operate under various time constraints and provide improved solutions as more time is allocated.
5. It can be applied to texts of any length.

We demonstrate the effectiveness of our algorithm on various types of text classification problems with different user requirements, and show that it can generate counterfactual examples that are valid, close to the original example, and plausible. We empirically show the advantage of our algorithm over state-of-the-art alternatives. Additionally, we report results of a human survey that validates the plausibility of our generated counterfactuals.

[2]An extended version of our paper, including appendices, is available at https://arxiv.org/abs/2211.00369.

Problem Definition

Let S be the set of all texts in English. Let $D \subseteq S$ be a sub-domain (e.g. movie reviews). Let C be a finite set of labels. Let $\sigma : D \times C \rightarrow [0, 1]$ be a model that estimates the probability of an object in D to belong to a class in C, and $\phi(x) = \text{argmax}_{y \in C}\, \sigma(x, y)$ be the associated classifier.

Let $x \in D$ be an object whose classification, $y = \phi(x)$, we need to explain. Let $\hat{y} \in C \setminus \{y\}$ be a *target label*[3]. Let $\tau \in (0, 1]$ be a classification confidence threshold. We define $\hat{x} \in D$ to be a *counterfactual example* to x with respect to $\hat{y}$ and τ if $\phi(\hat{x}) = \hat{y} \wedge \sigma(\hat{x}, \hat{y}) > \tau$. We denote the set of all such examples as $CF_{\tau,\hat{y}}(x)$. This definition follows two common requirements (Verma, Dickerson, and Hines 2020) from counterfactual examples: $\hat{x} \in D$ implies plausibility[4], and $\phi(\hat{x}) = \hat{y}$ implies validity.

Given a counterfactual example $\hat{x}$, we adapt Wachter, Mittelstadt, and Russell's definition (Wachter, Mittelstadt, and Russell 2017) to the textual domain, and define a *counterfactual explanation* as a text of the form: "If *x* had been changed to $\hat{x}$, the classification would have changed from y to $\hat{y}$". As a *counterfactual explanation* is immediate from a *counterfactual example*, we use the two terms interchangeably.

Humans find counterfactual examples that are minimally changed, compared to the original example, easier to utilize as explanations (Thagard 1989; Miller 2019; Alvarez-Melis et al. 2019; Wachter, Mittelstadt, and Russell 2017). Therefore, given a user-defined distance function between two texts $d : S \times S \rightarrow \mathbb{R}$, in addition to σ, x, $\hat{y}$ and τ, we can formulate our goal as an optimization problem of finding $\text{argmin}_{\hat{x} \in CF_{\tau,\hat{y}}(x)}\, d(x, \hat{x})$.

Search-based Counterfactual Generation

In this section, we present our new framework for the counterfactual generation task. We first formulate it as a search problem:

1. The set of states is D. The initial state is x, an object whose classification we need to explain.
2. The goal predicate for $t \in D$ is $goal(t) \iff t \in CF_{\tau,\hat{y}}(x)$.
3. The set of operators is defined in the next subsections.
4. The cost function g is defined over states. Given a user-specified distance function d, and a state $t \in D$, $g(t) = d(x, t)$.

Modification Operators

To perturb the example in the object space, we utilize two types of modification operators: *domain-agnostic* and *domain-specific* that take a text, x_{in}, as input and produce a modified text - a candidate. The first domain-agnostic operator is **mask filling**. In pre-processing time, a masked

[3]In the case of binary classification we can omit the specification of $\hat{y}$.

[4]Also appears in the literature as "fluency" or "likelihood" (Ross, Marasovic, and Peters 2021; Ross et al. 2022)

language model (MLM)[5] is fine-tuned for each class, using the examples of this class from the training set. During search, for each word w in x_{in}, we swap it with a mask token, and apply the MLM, fine-tuned on the target class, to get $R(w)$, a set of replacements with their associated score. The operator returns the highest-score suggestions: $\hat{R}(w) = \{(w', \xi') \in R(w) | \xi' \geq \alpha \cdot \max_{(\bar{w},\bar{\xi}) \in R(w)} \bar{\xi}\}$ where ξ' is the MLM score of the suggested replacement[6]. Similarly, in order to add words to x_{in}, we insert a mask token between every two words in the text, and ask the MLM for suggested filling. In addition, we utilize a word-removal operator, that modifies x_{in} by simply omitting words from it.

To produce a wider variety of goal-directed modifications, we introduce a second, novel domain-agnostic operator: **differentiating words banks**. This operator utilizes words that were identified, during pre-processing, as differentiating a specific class from others, and replaces words in x_{in} with words that are specifically associated with the target class.

To preserve grammatical correctness, we replace words with differentiating words of the same part of speech (POS). For a given POS, we create a bank of words for each class, that differentiate that class from the others. Given a word w that belongs to that POS (and appears in a text from the training set) and a class c, we conduct a multinomial test to determine if w is significantly over-represented in texts from c ($p < 0.05$). The bank of each class consists of the k over-represented words with the lowest p value[7].

Depending on the problem domain, some POS are more likely to separate the classes. For example, "adjective" may be a differentiating POS for sentiment analysis, while "noun" is such for topic classification. To reduce the branching factor of the search procedure, we therefore first identify which parts of speech are the relevant ones, and construct differentiating words banks for those POS only. For every POS and every class, we collect the list of words of that POS that appear in texts in the training set that belong to the class (e.g. all the verbs that appear in a positive example). We then use a semantic embedder[8] to transform every word in each classes' lists to a vector space, and we use MANOVA[9] to determine if the clusters' means are significantly different. POS where MANOVA test returns a significantly small p value ($p < 0.05$) are the differentiating POS. For a given word w in x_{in}, we check if its POS is a differentiating one, in which case we swap w with each of the words in the bank associated with that POS in the target class.

In some domains, we may be able to utilize prior knowledge and define *domain-specific* operators that direct the search process towards a goal state. Namely, some of the experiments described in this paper deal with sentiment analysis problems. The domain-specific operators we introduce for this task are based on WordNet antonyms (Miller 1995). For a given word in x_{in}, we swap it with its antonyms.

[5]We have used DistilBERT (Sanh et al. 2019).

[6]$0 \leq \alpha \leq 1$ is a parameter set to 0.5 in our experiments.

[7]We used k = 10.

[8]We used GloVe (Pennington, Socher, and Manning 2014) .

[9]Some of MANOVA's assumptions may be violated.

Algorithm 1: TCE-SEARCH

Input: $x, d, \hat{y}, \tau, TS$ $/* TS$ is the training set$*/$
$cf \leftarrow \text{argmin}_{\hat{x} \in TS \cap CF_{\tau,\hat{y}}(x)} d(x, \hat{x})$
$w_h \leftarrow 1.0$
while not interrupted **do**
  $new\_cf \leftarrow \text{WA}^*(w_h, d, x, \hat{y}, \tau)$
  if $d(x, new\_cf) < d(x, cf)$ **then**
    $cf \leftarrow new\_cf$
  end if
  $w_h \leftarrow \frac{w_h}{2}$
end while
return cf

Plausibility Enforcement

There is a possibility that our operators will take us out of D, violating the *plausibility* requirement. To ensure plausibility, we use a fine-tuned language model (LM) [10] to filter out the candidates (produced by the modification operators) that are estimated to be out of D. Since the original example x is in D, we demand that every node in the search procedure is nearly as likely to appear in D as x, i.e. nearly as *plausible* as x. To this end, following (Ross, Marasovic, and Peters 2021; Ross et al. 2022), we compute the language-modeling loss for x and for each of the candidates, and return only those with a ratio of losses (candidate loss / original loss) that is at most γ, where γ is a positive parameter [11]. Lower γ values introduce a more strict plausibility threshold.

The Heuristic Function

Our heuristic function h estimates the distance of $t \in D$ to $CF_{\tau,\hat{y}}(x)$ by the normalized distance between the estimated target class probability for t, and the required classification confidence threshold. Formally, $h(t) = \max\{0, \frac{\tau - \sigma(t,\hat{y})}{\tau}\}$.

The Search Framework

We base our framework on a variation of the weighted-A^* algorithm (Pohl 1970), where the node selected to be expanded is the one with the lowest $f = (1 - w_h) \cdot g + w_h \cdot h$ score where $w_h \in [0, 1]$. Contrary to the original weighted-A*, we define the cost function over states rather than graph edges, and the heuristic function is not admissible. Our search framework works by iteratively calling our variation of weighted-A* (WA*) with decreasing values of w_h. The algorithm maintains the best solution found so far and returns it when interrupted (by the user or by the clock). This makes our search an *anytime* process.

Algorithm 1 presents TCE-SEARCH. To ensure that a valid counterfactual is generated, the algorithm starts with a very fast baseline that returns a counterfactual from the training set with the lowest d (*default counterfactual*). If the training set is very large, we can return the closest counterfactual found in a sample of it. It then calls WA^* with $w_h = 1$, and continues to call it, iteratively, while reducing

[10]We used a fine-tuned GPT-2 (Radford et al. 2019) .

[11]We used $\gamma = 1.5$ in our experiments.

w_h in every iteration, reducing the effect of h. Thus, each iteration is expected to generate a better solution than the previous at the expense of increased search time. To save time, we cache the results of the calls to the MLM and LM and use it in the following iterations of the anytime algorithm.

Handling Longer Texts

As the branching factor of the search space is proportional to the length of the input text, we introduce a focused version of the algorithm to deal with longer texts. Let $t = \langle s_1, \dots s_n \rangle$ be a longer text where s_i are sentences. The modified algorithm starts, as before, with $w_h = 1$. Instead of working on the entire text at once, the algorithm iteratively selects a sentence s_i and uses the basic algorithm to find a counterfactual $\hat{s_i}$ for it. If $\langle s_1, \dots s_{i-1}, \hat{s_i} \dots s_n \rangle \in CF_{\tau,\hat{y}}(t)$, or it runs out of sentences, the algorithm proceeds to the next top-level iteration with a reduced w_h as previously described.

The sentences are selected according to their estimated importance evaluated by: $\theta(s_i) = \sigma(t \setminus \{s_i\}, \hat{y}) - \sigma(t, \hat{y})$, where $t \setminus \{s_i\}$ is t with the sentence s_i removed. Intuitively, θ is an importance score for each sentence, measured by the amount of confidence gained in the target prediction once the sentence has been omitted from the input text.

Empirical Evaluation

We evaluated the performance of our framework on a variety of classification tasks.

Experimental Methodology

We report results for 8 datasets: (1) **Yelp**. Yelp business reviews. (2) **Amazon** (Ni, Li, and McAuley 2019). Video game reviews on Amazon. (3) **SST** (Socher et al. 2013). Stanford sentiment treebank. (4) **Science**. Comments from Reddit on scientific subjects. (5) **Genre**. Movie plot descriptions labeled by genre. (6) **AGNews** (Zhang, Zhao, and LeCun 2015). AG's news articles labeled by topic. (7) **Airline**. Tweets about flight companies. (8) **Spam** (Almeida, Hidalgo, and Yamakami 2011). SMS labeled either as spam or as legitimate. We applied a standard pre-processing procedure, the details of which can be found in the appendix.

For each dataset, we randomly sampled 200 examples to serve as the explanation test set. We randomly split the rest of the dataset, so that 80% serves as the black-box training set and 20% remains for the black-box test set.

For each training set, we fine-tuned a LM to be used for plausibility enforcement, and a MLM for each class for the mask filling operator[12]. Importantly, these models only require fine-tuning **once per training set**. Afterwards, we can utilize them in different scenarios, employing various distance functions that represent changing user requirements, without further training.

We consider three distance criteria between the original text x and the generated counterfactual $\hat{x}$:

1. Normalized word-level Levenshtein distance (Levenshtein et al. 1966) $\frac{d_{lev}(x,\hat{x})}{|x|}$, where $|x|$ is the number of words instances in x.
2. Normalized cosine distance between x and $\hat{x}$'s encodings[13]: $\frac{d_{cosine}(enc(x),enc(\hat{x}))}{2}$.
3. Normalized syntactic tree distance (Zhang and Shasha 1989) $\frac{d_{tree}(x,\hat{x})}{|x|}$. When dealing with texts longer than a single sentence, we sum the distances of the sentences.

As previously outlined, TCE-SEARCH is an anytime algorithm, which terminates its run when the allocated computational budget is exhausted. A common method for measuring the resources consumed by a search algorithm is to count the number of nodes that the algorithm expanded during its execution. However, in our experiments we found that metric to be problematic, as the time it takes to expand a node varies significantly between nodes of different text length, as well as between cached (previously seen) nodes and uncached ones. Since uncached calls to the LM and MLM dominate our running time ($> 95\%$), we decided to use the number of such expensive calls (EC) as our metric for measuring the time resources consumed by the algorithm.

In all the following experiments, unless otherwise stated, the following configuration was used as the standard setting:

- A random forest was used as the black box classifier.
- The full version of our framework was utilized, incorporating all introduced operators to explore the search space.
- TCE-SEARCH was given a budget of 2000 expensive calls (EC).
- Levenshtein distance was used as the distance function, the results were averaged across the explanation test set.
- The classification confidence threshold τ is 0.5.
- No text length limitation was imposed on the datasets.

Comparison to Existing Works

We evaluate the performance of TCE-SEARCH compared to 3 baselines with published source code: **MiCE** (Ross, Marasovic, and Peters 2021), **Polyjuice** (Wu et al. 2021) and **Tailor** (Ross et al. 2022). Further description of these works is in the Related Work section.

User requirements for generated counterfactuals can be formulated as a distance function from the original text. We compare the performance of the methods over 3 distance metrics, representing different possible user requirements.

All three baseline methods are based on generative models and are not guaranteed to generate a valid counterfactual (i.e. a text classified as the target class). Additionally, the models were trained to minimize a distance function that do not necessarily correspond to the one used to measure performance. To address these issues, we use augmented versions of the 3 baseline models. Each generative model is used to generate a set of 10 candidate texts. The set is then filtered to include only valid counterfactuals, and the one

[12]We used the same fine-tuning procedure for both the LM and the MLM: 3 epochs with initial LR of 5e-05 and weight decay of 0.0 for AdamW. We used a batch size of 2.

[13]For encoding, we used the HuggingFace model available at: https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

	Yelp_{256w}			Amazon_{256w}			Airline_{256w}			SST_{256w}		
Method	Lev	Cosine	Syn	Lev	Cosine	Syn	Lev	Cosine	Syn	Lev	Cosine	Syn
TCE-SEARCH	0.56	**0.066**	**0.121**	0.448	**0.087**	**0.144**	**0.257**	**0.047**	**0.145**	**0.207**	**0.043**	**0.222**
MiCE	**0.266***	0.084*	0.351*	**0.32***	0.106*	0.386*	0.269	0.1*	0.289*	0.256*	0.11*	0.336*

Table 2: Average normalized Levenshtein (Lev), Cosine and Syntactic tree (Syn) distances, for sentiment analysis datasets with 256 words (256w) text length restriction. RoBERTa-LARGE model was used as the black box classifier. Best (lowest) results are bold. We use an asterisk (*) to indicate results that are statistically significantly different (paired t-test, $p = 0.05$) from TCE-SEARCH.

	Yelp_{1s}			Amazon_{1s}			Airline_{1s}		
Method	Lev	Cosine	Syn	Lev	Cosine	Syn	Lev	Cosine	Syn
TCE-SEARCH	**0.148**	**0.027**	**0.076**	**0.229**	**0.05**	**0.208**	**0.174**	**0.025**	**0.157**
Polyjuice	0.692*	0.16*	0.542*	0.591*	0.163*	0.596*	0.723*	0.168*	0.657*
Tailor	0.8*	0.19*	0.65*	0.715*	0.199*	0.752*	0.763*	0.162*	0.671*
	SST_{1s}			Science_{1s}			AGnews_{1s}		
	Lev	Cosine	Syn	Lev	Cosine	Syn	Lev	Cosine	Syn
TCE-SEARCH	**0.085**	**0.009**	**0.082**	**0.266**	**0.048**	**0.379**	**0.14**	**0.027**	**0.175**
Polyjuice	0.562*	0.159*	0.653*	0.678*	0.199*	0.818*	0.799*	0.313*	0.891*
Tailor	0.658*	0.174*	0.784*	0.716*	0.218*	0.903*	0.818*	0.311*	0.948*

Table 3: Average normalized distances for datasets with 1 sentence (1s) text length restriction. Best (lowest) results are bold. We use an asterisk (*) to indicate results that are statistically significantly different (paired t-test, $p = 0.05$) from TCE-SEARCH.

with the lowest distance (according to the distance function used for evaluation) is selected. Importantly, for our TCE-SEARCH algorithm, this procedure is not necessary, as it is guaranteed to generate a valid counterfactual and the distance function is used as the cost function for the search procedure, so it is directly optimized during the search.

To ensure fair comparison, all methods produce the *default counterfactual* if they are unable to generate a closer valid counterfactual.

Since MiCE is limited to differentiable models and requires access to the model's gradient, when comparing to MiCE, we used the exact model from the MiCE paper, RoBERTa-LARGE trained on the IMDB sentiment analysis dataset (Maas et al. 2011)[14], as the black box classifier. Since the model is trained on a sentiment analysis dataset, we use only the sentiment analysis datasets for comparison with MiCE. Furthermore, since MiCE is based on the T5 model (Raffel et al. 2020) with a maximum input sequence length of 512 tokens, we used modified versions of the datasets for comparison with MiCE, omitting texts longer than 256 words, as a word may be composed of multiple tokens. Similarly, for comparison with Polyjuice and Tailor, which are designed to modify individual sentences, we created modified versions of the datasets containing only single sentences. We did not include the Genre and Spam datasets in this comparison becaues of the small number of single-sentence texts in them.

The comparisons results are available at Tables 2 and 3. The results show that TCE-SEARCH is able to produce counterfactuals that are generally closer to the original example than the alternatives, across different distance criteria, datasets, and text length limitations. This experiment displays TCE-SEARCH's ability to adapt to different user requirements, encapsulated in the distance function that was optimized in the search procedure, without retraining. It is worth noting, however, that the baselines used fewer EC compared to TCE-SEARCH[15].

Ablation Study

In order to evaluate the effectiveness of the operators previously introduced, we compare 3 variations of our proposed framework: (1) **TCE-SEARCH-full** that uses all operators, (2) **TCE-SEARCH-no-DWB**, that does not use the differentiating words banks operator, and (3) **TCE-SEARCH-no-antonyms**, that uses only the domain-agnostic operators, without using the WordNet antonym operator.

Table 4 shows the performance of the algorithm variations on all eight datasets. The DWB operator had a statistically significant impact on performance, while the domain-specific antonyms operator's effect was not significant (paired t-test, $p = 0.05$).

Anytime Performance

As previously mentioned, TCE-SEARCH is an anytime algorithm: It returns a higher quality solution with increase

[14]For full details on the model's architecture and training, please refer to MiCE paper.

[15]To the best of our understanding, both Tailor and Polyjuice require a single forward pass (EC) to generate a modified text. As for MiCE, the authors state that the algorithm performs three "edit rounds", each costs approximately 360 EC, resulting in a total of around 1080 EC.

Variation	SST	Yelp	Amazon	Genre	Spam	Airline	AGnews	Science
TCE-SEARCH-full	0.088	**0.426**	**0.398**	**0.269**	0.261	**0.153**	**0.234**	**0.305**
TCE-SEARCH-no-antonyms	**0.086**	0.437	0.411	0.29	**0.254**	0.154	0.237	0.311
TCE-SEARCH-no-DWB	0.135*	0.510*	0.47*	0.464*	0.362*	0.2*	0.343*	0.442*

Table 4: Average normalized Levenshtein distance, for various TCE-SEARCH variations. Best (lowest) results are bold. We use an asterisk (*) to indicate results that are statistically significantly different ($p = 0.05$) from the full version.

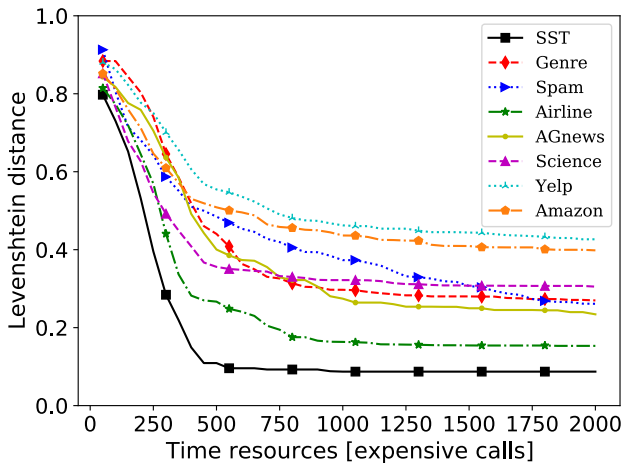

Figure 1: Anytime performance: Levenshtein distance as a function of allocated expensive calls (EC).

in allocated time. This is important for our use case, where the explanation is likely needed during an interactive session (Nielsen 1994).

The anytime performance of TCE-SEARCH can be observed in Figure 1. The results demonstrate that for all 8 datasets, as more resources are allocated, the average normalized distance decreases until a stable point is reached. The plots indicate that TCE-SEARCH can generate valid counterfactuals quickly when resources are limited but can also improve the quality of counterfactuals as more resources become available.

Robustness to Different Classifiers

To demonstrate the model-agnostic nature of TCE-SEARCH we have tested its performance using 3 different classifiers that used 3 different embedding methods[16]: (1) A random forest classifier with CountVectorizer, used in most previous experiments[17]. (2) A logistic regression classifier with HashingVectorizer. (3) A SVM classifiers with TfidfVectorizer.

Table 5 illustrates that TCE-SEARCH is relatively consistent in its performance across different embedding techniques and classification algorithms, at least within the scope of this experiment.

[16]The scikit-learn library (Pedregosa et al. 2011) was used for models and embeddings.

[17]When comparing with MiCE we used RoBERTa-LARGE.

The Plausibiliy of Generated Counterfactuals

As previously mentioned, in order to ensure plausibility, we monitor the LM loss ratio of candidate states during the search process, and allow only candidates with a loss ratio below the γ threshold (1.5 in our experiments).

Similarly to Ross, Marasovic, and Peters (2021) and Ross et al. (2022), we first evaluate the plausability of the generated counterfactuals using their measured average LM loss ratio. The results are presented in Table 6. As can be seen, the actual average ratios are significantly lower than γ and are close to a ratio of 1.0, which would mean equal LM loss (indicating equal plausibility) for the counterfactual and the original example.

We further evaluated the plausabilty using human raters. We randomly sampled 50 examples of the SST explanation test set, and asked non-native English speakers with a high level of proficiency in the language to rank the plausibility of each of the 50 original examples, and the 50 counterfactuals generated by TCE-SEARCH for these texts. We used 3 annotators for each example. The annotators were asked: "Please rank each text on a scale of 1 to 5, where 1 indicates that the text is not a movie review, and 5 indicates that it is a movie review".

We averaged the 3 labels for each text. The average score for the original texts was 3.66, and for the generated counterfactuals 3.33. A paired t-test ($p = 0.05$) has shown that the difference is not statistically significant. The average standard deviation between the 3 labels of the original texts is 0.749, and for the counterfactuals generated by TCE-SEARCH is 0.643.

Multi-Class Setting

The experiments in this paper have focused on binary classification problems for simplicity. However, TCE-SEARCH is also well-suited multi-class classification tasks. To demonstrate this, we modified the Genre and AGnews datasets to have three classes, as outlined in the appendix, and tested TCE-SEARCH's performance on them. We generated 2 counterfactual examples for each sample, one for every possible target label. The average distance for the Genre explanation test set was 0.57, and 0.257 for the AGnews.

Related Work

Research from the social sciences suggests that often, when humans provide explanations as to why an event had occured, they do not explain the cause to the event per se, but rather explain the cause compared to some other event that did not occur (Miller 2019; Alvarez-Melis et al. 2019;

Classifier	SST	Yelp	Amazon	Genre	Spam	Airline	AGnews	Science
RF	0.088	0.426	0.398	0.269	0.261	0.153	0.234	0.305
LR	0.102	0.525	0.386	0.431	0.293	0.214	0.296	0.413
SVM	0.111	0.617	0.48	0.448	0.335	0.215	0.32	0.428

Table 5: Average normalized Levenshtein distance, for TCE-SEARCH with various black-box classifiers.

Plausibility Ratio							
Yelp	SST	Amazon	Genre	Science	AGN	Airline	Spam
1.05	1.03	1.02	1.04	1.07	1.2	1.09	1.05

Table 6: Language modeling loss ratio between TCE-SEARCH generated counterfactuals and original examples over explanation test sets.

Hilton 1990). The common term to this kind of explanations is *counterfactual explanations*.

There are various cases where counterfactual explanations are better than feature-importance based explanations (Fernandez, Provost, and Han 2020; Ross, Marasovic, and Peters 2021). In recent years, therefore, researchers have developed algorithms for generation of counterfactual explanations. One of the first works to propose counterfactual explanations to decisions made by black-box classifiers was by Wachter, Mittelstadt, and Russell (2017). Several works have proposed to perturbe the original example in the feature space (Karimi et al. 2021; Mothilal, Sharma, and Tan 2020; Galhotra, Pradhan, and Salimi 2021). While these methods are of great use in some domains, they may be inapplicable in other important domains, such as computer vision and NLP, where points in the feature space are incomprehensible to humans (Zarecki and Markovitch 2020).

Recent research has investigated the concept of generating counterfactuals in the textual domain. These methods often utilize transformer-based generative language models (Vaswani et al. 2017), which have a limitation on the maximum length of input sequences. Some prevalent approaches are designed to modify single sentences. Madaan et al. (2021) generate counterfactuals for sentences in order to produce test cases and debias the model. Polyjuice (Wu et al. 2021) is a general-purpose counterfactual creator, which generates modified sentences with small syntactic and semantic distance to the original example. Tailor (Ross et al. 2022) modifies sentences in a semantically-controlled fashion, using control codes derived from semantic representation. Robeer, Bex, and Feelders (2021) generates perceptibly distinguishable (large semantic distance compared to original example) counterfactuals, in a model agnostic manner.

Other approaches require a white box access to the model (Yang et al. 2020; Fern and Pope 2021). The frameworks suggested by Jacovi et al. (2021) and Yang et al. (2021) are limited to neural models.

MiCE (Ross, Marasovic, and Peters 2021) and CAT (Chemmengath et al. 2021) are similar to TCE-SEARCH in utilizing a MLM to generate plausible counterfactuals. Unlike TCE-SEARCH , however, these methods are not completely model agnostic, as they use the black box's gradient information to select which parts of the text to mask. This limits their use to differentiable models.

Lampridis, Guidotti, and Ruggieri (2020) builds a decision tree in the latent neighborhood of the original instance, and then extracts exemplars and counter-exemplars, close to the original example according to Euclidean distance. Dixit et al. (2022) generates counterfactuals for the purpose of data augmentation, by constructing detailed prompts to GPT-3 (Brown et al. 2020).

Another limitation of methods that are based on generative language models is that they are optimized for a specific loss function, which may not be suitable for other tasks. Therefore, in order to use these models for other tasks, retraining is often required, which can be costly. In contrast, TCE-SEARCH has the ability to adapt to different user needs and constraints without the need for retraining.

We are aware of no other work that generates counterfactual explanations for text classification in an anytime, model-agnostic and domain-agnostic manner, with the ability to handle texts of any length while being able to adapt to varying user requirements without retraining.

Martens and Provost (2014), Fernandez, Provost, and Han (2020) and Ramon et al. (2019) generate explanations by selecting important words for classification. While they use the term "counterfactual explanations", their definition of such explanations is different from the common usage that refers to a complete object of the opposite class.

Note that while *adversarial learning* algorithms also perturb examples, their goals and constraints differ (Verma, Dickerson, and Hines 2020). The aim in adversarial learning is to fool the classifier by generating slightly altered examples with the same true label but a different predicted label. The problem of counterfactual explanation generation is not subject to this constraint. On the other hand, adversarial learning algorithms are not restricted to generating plausible and human understandable objects, hence they cannot be used for explanation.

Conclusion

In this paper, we presented TCE-SEARCH, a general model-agnostic search-based framework for generating counterfactual explanations for text classification. Our approach adapts to user-specified constraints and preferences, such as a distance function and computational budget, *without retraining*. TCE-SEARCH's counterfactuals are plausible, valid and close to the original examples, outperforming current state-of-the-art approaches.

Future work should investigate the *utility* of these explanations to humans through further user studies.

References

Almeida, T. A.; Hidalgo, J. M. G.; and Yamakami, A. 2011. Contributions to the study of SMS spam filtering: new collection and results. In *Proceedings of the 2011 ACM Symposium on Document Engineering, Mountain View*, 259–262. ACM.

Alvarez-Melis, D.; III, H. D.; Vaughan, J. W.; and Wallach, H. M. 2019. Weight of Evidence as a Basis for Human-Oriented Explanations. *CoRR*, abs/1910.13503.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901.

Chemmengath, S.; Azad, A. P.; Luss, R.; and Dhurandhar, A. 2021. Let the CAT out of the bag: Contrastive Attributed explanations for Text. *CoRR*, abs/2109.07983.

Dixit, T.; Paranjape, B.; Hajishirzi, H.; and Zettlemoyer, L. 2022. CORE: A Retrieve-then-Edit Framework for Counterfactual Data Generation. *arXiv preprint arXiv:2210.04873*.

Doshi-Velez, F.; Kortz, M.; Budish, R.; Bavitz, C.; Gershman, S.; O'Brien, D.; Schieber, S.; Waldo, J.; Weinberger, D.; and Wood, A. 2017. Accountability of AI Under the Law: The Role of Explanation. *CoRR*, abs/1711.01134.

Fern, X. Z.; and Pope, Q. 2021. Text Counterfactuals via Latent Optimization and Shapley-Guided Search. In *Proceedings of the 2021 EMNLP*, 5578–5593.

Fernandez, C.; Provost, F. J.; and Han, X. 2020. Explaining Data-Driven Decisions made by AI Systems: The Counterfactual Approach. *CoRR*, abs/2001.07417.

Galhotra, S.; Pradhan, R.; and Salimi, B. 2021. Explaining Black-Box Algorithms Using Probabilistic Contrastive Counterfactuals. In *International Conference on Management of Data,2021*, 577–590.

Hilton, D. J. 1990. Conversational processes and causal explanation. *Psychological Bulletin*, 107(1): 65.

Jacovi, A.; Swayamdipta, S.; Ravfogel, S.; Elazar, Y.; Choi, Y.; and Goldberg, Y. 2021. Contrastive Explanations for Model Interpretability. In *Proceedings of the 2021 EMNLP*, 1597–1611.

Karimi, A.-H.; Barthe, G.; Balle, B.; and Valera, I. 2021. Model-agnostic counterfactual explanations for consequential decisions. In *Oroceedings of AISTATSS 2021*, 895–905.

Lampridis, O.; Guidotti, R.; and Ruggieri, S. 2020. Explaining Sentiment Classification with Synthetic Exemplars and Counter-Exemplars. In Appice, A.; Tsoumakas, G.; Manolopoulos, Y.; and Matwin, S., eds., *Discovery Science - 23rd International Conference, DS 2020, Thessaloniki, Greece, October 19-21, 2020, Proceedings*, volume 12323 of *Lecture Notes in Computer Science*, 357–373. Springer.

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

Maas, A. L.; Daly, R. E.; Pham, P. T.; Huang, D.; Ng, A. Y.; and Potts, C. 2011. Learning Word Vectors for Sentiment Analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 142–150. Portland, Oregon, USA: Association for Computational Linguistics.

Madaan, N.; Padhi, I.; Panwar, N.; and Saha, D. 2021. Generate Your Counterfactuals: Towards Controlled Counterfactual Generation for Text. In *Thirty-Fifth AAAI 2021*, 13516–13524.

Martens, D.; and Provost, F. J. 2014. Explaining Data-Driven Document Classifications. *MIS Q.*, 38(1): 73–99.

Miller, G. A. 1995. WordNet: a lexical database for English. *Communications of the ACM*, 38(11): 39–41.

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

Mothilal, R. K.; Sharma, A.; and Tan, C. 2020. Explaining machine learning classifiers through diverse counterfactual explanations. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, 607–617.

Ni, J.; Li, J.; and McAuley, J. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 (EMNLP-IJCNLP)*, 188–197.

Nielsen, J. 1994. *Usability engineering*. San Francisco, Calif.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; and Duchesnay, E. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12: 2825–2830.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global Vectors for Word Representation. In *Proceedings of EMNLP 2014*, 1532–1543.

Pohl, I. 1970. Heuristic search viewed as path finding in a graph. *Artificial intelligence*, 1(3-4): 193–204.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; and Sutskever, I. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8): 9.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; Liu, P. J.; et al. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140): 1–67.

Ramon, Y.; Martens, D.; Provost, F. J.; and Evgeniou, T. 2019. Counterfactual Explanation Algorithms for Behavioral and Textual Data. *CoRR*, abs/1912.01819.

Robeer, M.; Bex, F.; and Feelders, A. 2021. Generating Realistic Natural Language Counterfactuals. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, 3611–3625.

Ross, A.; Marasovic, A.; and Peters, M. E. 2021. Explaining NLP Models via Minimal Contrastive Editing (MiCE). In *Findings of ACL/IJCNLP 2021*, 3840–3852.

Ross, A.; Wu, T.; Peng, H.; Peters, M. E.; and Gardner, M. 2022. Tailor: Generating and Perturbing Text with Semantic Controls. In *Proceedings of the 60th Annual Meeting of the*

Association for Computational Linguistics. Online: Association for Computational Linguistics.

Sanh, V.; Debut, L.; Chaumond, J.; and Wolf, T. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108.

Socher, R.; Perelygin, A.; Wu, J.; Chuang, J.; Manning, C. D.; Ng, A. Y.; and Potts, C. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 EMNLP*, 1631–1642.

Thagard, P. 1989. Explanatory coherence. *Behavioral and brain sciences*, 12(3): 435–502.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Verma, S.; Dickerson, J. P.; and Hines, K. 2020. Counterfactual Explanations for Machine Learning: A Review. *CoRR*, abs/2010.10596.

Wachter, S.; Mittelstadt, B.; and Russell, C. 2017. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harv. JL & Tech.*, 31: 841.

Wu, T.; Ribeiro, M. T.; Heer, J.; and Weld, D. S. 2021. Polyjuice: Generating Counterfactuals for Explaining, Evaluating, and Improving Models. In *Proceedings of ACL/IJCNLP 2021*, 6707–6723.

Yang, F.; Liu, N.; Du, M.; and Hu, X. 2021. Generative Counterfactuals for Neural Networks via Attribute-Informed Perturbation. *SIGKDD Explor.*, 23(1): 59–68.

Yang, L.; Kenny, E. M.; Ng, T. L. J.; Yang, Y.; Smyth, B.; and Dong, R. 2020. Generating Plausible Counterfactual Explanations for Deep Transformers in Financial Text Classification. In *Proceedings of ICL 2020*, 6150–6160.

Zarecki, J.; and Markovitch, S. 2020. Textual Membership Queries. In *Proceedings of IJCAI 2020*, 2662–2668.

Zhang, K.; and Shasha, D. 1989. Simple fast algorithms for the editing distance between trees and related problems. *SIAM journal on computing*, 18(6): 1245–1262.

Zhang, X.; Zhao, J. J.; and LeCun, Y. 2015. Character-level Convolutional Networks for Text Classification. In *NeurIPS 2015*, 649–657.

Zilberstein, S. 1996. Using Anytime Algorithms in Intelligent Systems. *AI Magazine*, 17(3): 73.