

Zero-shot LLM-guided Counterfactual Generation for Text

Amrita Bhattacharjee^{a,*}, Raha Moraffah^a, Joshua Garland^a and Huan Liu^a

^aSchool of Computing and AI, Arizona State University, Arizona, USA

Abstract.

Counterfactual examples are frequently used for model development and evaluation in many natural language processing (NLP) tasks. Although methods for automated counterfactual generation have been explored, such methods depend on models such as pre-trained language models that are then fine-tuned on auxiliary, often task-specific datasets. Collecting and annotating such datasets for counterfactual generation is labor intensive and therefore, infeasible in practice. Therefore, in this work, we focus on a novel problem setting: *zero-shot counterfactual generation*. To this end, we propose a structured way to utilize large language models (LLMs) as general purpose counterfactual example generators. We hypothesize that the instruction-following and textual understanding capabilities of recent LLMs can be effectively leveraged for generating high quality counterfactuals in a zero-shot manner, without requiring any training or fine-tuning. Through comprehensive experiments on various downstream tasks in natural language processing (NLP), we demonstrate the efficacy of LLMs as zero-shot counterfactual generators in evaluating and explaining black-box NLP models.

1 Introduction

Over the last couple of decades, machine learning and natural language processing (NLP) systems have developed massively, especially in terms of the complexity and scale of the models used in different downstream tasks. For example, for most NLP tasks, such as tasks in the GLUE or SuperGLUE benchmark, the state-of-the-art performance is achieved by large, black-box models such as pre-trained language models (PLM) [23]. Effective use and deployment of such models, especially in high stakes areas, require careful evaluation, validation and stress-testing. Furthermore, models should also be explainable or interpretable, i.e., decisions made by such black-box models should ideally be accompanied by how and/or why the model reached that decision [28]. While such endeavors are still challenging in the context of black-box models, in this regard, *counterfactual examples* have been used to perform evaluation, explanation, robustness testing and even improvement of NLP models [48, 26, 5]. For example, the following two sentences - *s1: This movie is brilliant!*, *s2: This movie is boring*, are counterfactual examples for the input sentence *This movie is great*. Such minimally perturbed variations of the input text can be used in a variety of settings to evaluate models, to understand whether a model is able to focus on the task-specific features in the input text in order to classify the input text correctly.

* Corresponding Author. Email: abhatt43@asu.edu.

original	sentiment
Their cheese burger is amazing.	pos
counterfactuals	
Their cheese burger is <u>bland</u> .	neg
Their cheese burger is <u>so original</u> .	pos
Their cheese burger is <u>not</u> amazing.	neg

Figure 1. Examples of an input sentence and its corresponding counterfactual examples with same or opposite label.

While several previous works have investigated the applicability of human expert annotators to design such counterfactual examples [16, 32], this is not scalable in practice, thereby motivating the exploration of automated counterfactual generation methods. Automated counterfactual generation methods such as [48, 26] use pre-trained language models, or mask-filling, or models trained via control codes for the generation. Such models require training or fine-tuning using multiple datasets that are often task-specific in nature. For example, training a conditional generation model in Polyjuice [48] requires sentence-pair dataset for each control code (such as: negation, quantifier, shuffle, lexical, etc.). Similar methods requiring vast amounts of training and/or data are used in other automated counterfactual generation methods. However, having access to such task-specific training datasets may be infeasible in practice, especially for newly emerging data domains and tasks. Therefore, we are interested in investigating: *Is there a way to simplify the counterfactual generation process and perform the generation without any auxiliary data?*

To this end, in this work, we address a new problem setting: *zero-shot counterfactual generation*. We tackle this problem by using the power of recent state-of-the-art instruction-tuned large language models (LLMs). Given that these LLMs are trained on massive amounts of text data, followed by subsequent supervised fine-tuning and alignment steps, there is empirical evidence to suggest that such LLMs can be used as pseudo-oracles or general purpose solvers especially in natural language processing tasks [8]. As an extension, we propose the paradigm of using LLMs as zero-shot counterfactual generators for text. To this end, we propose a pipeline that leverages recent LLMs in order to generate plausible, human-interpretable counterfactual examples in a completely zero-shot manner. Our proposed pipeline requires only the input text along with either the ground truth label or the predicted label from the black-box classifier (depending on the use-case) and uses a simple hard-prompting method to use off-the-shelf LLMs for generating the counterfactuals,

without any fine-tuning or training with additional data. We envision that automating the task of counterfactual generation via a carefully designed pipeline that leverages LLMs can help to reduce costs and make NLP model development, evaluation and explanation more streamlined and efficient. We use our proposed pipeline to generate counterfactual explanations and we empirically evaluate the quality and effectiveness of the generated counterfactuals in (1) explaining and, (2) evaluating NLP models for a variety of downstream tasks. Our results demonstrate that, when used in our pipeline, LLMs can generate high quality and effective zero-shot counterfactuals that preserve high semantic similarity with the original input text.

To the best of our knowledge, this is the first piece of work to tackle the problem of zero-shot counterfactual generation in text, and also the first to utilize LLMs as general purpose counterfactual generators. Overall our contributions in this paper are as follows:

1. We propose a **F**ramework for **I**nstructed **Z**ero-shot Counterfactual Generation with **L**anguage **E** Models, which we refer to as **FIZLE** for brevity¹.
2. We demonstrate **FIZLE** for two important use-cases: *explaining* and *evaluating* black-box text classification models.
3. Through experiments on three benchmark datasets and tasks we investigate the effectiveness of the proposed pipeline compared to recent baselines and discuss implications for future work in this direction.

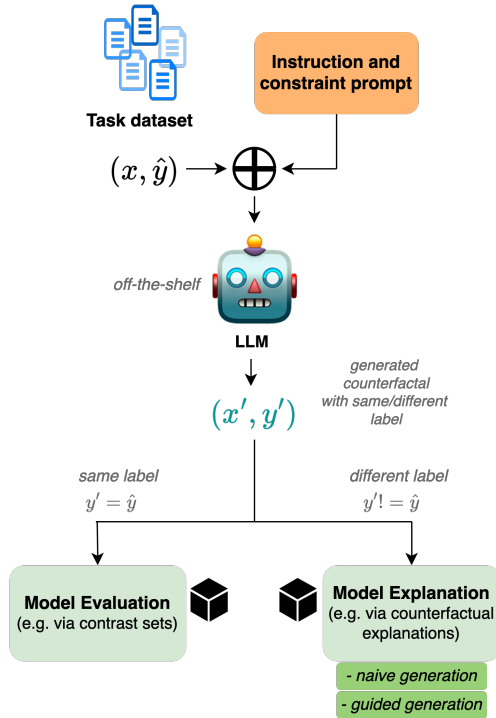


Figure 2. Our proposed **FIZLE** framework for zero-shot LLM-guided general purpose counterfactual generation, along with applications in model evaluation and explanation.

The rest of the paper is organized as follows: Section 2 dives into the background and relevant related works. Section 3 describes our pipeline and methodology in detail. We briefly talk about our general experimental settings in Section 4. We demonstrate our two use-cases, Counterfactual Explanations in Section 5 and Evaluation using

Counterfactuals as Contrast Sets in Section 6. We finally conclude with a discussion on future works in Section 7.

2 Background and Related Works

In this section we describe some preliminary concepts along with related works.

Counterfactual Generation According to most definitions in literature [28], counterfactuals in text are minimally edited versions of an original text that can flip the label of a classifier. Counterfactuals are typically similar to the input instance, and vary from it in a small number of features. Counterfactual examples are widely used to stress-test trained models, provide explanations in the form of counterfactual explanations, and also for model improvement via training with counterfactual examples and counterfactually augmented data. While several efforts have been made in the manual creation of counterfactuals [16, 32], this method does not scale up and is therefore infeasible in practice for most use-cases. Automated methods for counterfactual generation are therefore more prevalent. Such methods often use language models trained on some control codes for conditional text generation in order to generate plausible and diverse counterfactuals [26, 48]. In the context of text classification, which is the scope of this paper, counterfactual examples can be generated from the input text via token-based substitution methods, masked-language modeling, controlled text generation via control codes [27]. Authors in [37] create realistic counterfactuals via language modeling using a Counterfactual GAN architecture. However, all of these methods use either auxiliary models and/or training data, for example, to capture the style characteristics of different control codes. In contrast, in this work we focus on *zero-shot counterfactual generation*.

Large Language Models and Applications in NLP Large Language Models (LLMs) are usually transformer-based models capable of generating human-like text. Recent examples of large language models include the GPT family of models [33, 7, 29], Llama [41], Llama-2 [42], Falcon², etc. A general training recipe for training LLMs include an unsupervised pre-training step, where the model is trained using a huge corpora of text, typically comprising of text from the internet [31, 15], followed by one or more supervised fine-tuning steps, such as instruction-tuning [30, 12]. More recent state-of-the-art LLMs such as ChatGPT-3.5 or GPT-4 are further fine-tuned via reinforcement learning with human feedback (RLHF) [11], in order to ‘align’ such models more with human preferences and values. During the fine-tuning and RLHF stages LLMs learn to follow instructions for specific tasks and respond in a helpful manner. Instruction-tuning essentially fine-tunes the model on massive datasets of (instruction, output) pairs, whereby LLMs *learn* to follow instructions in a prompt in order to perform tasks. The vast amount of training, both via the pre-training and the instruction-tuning stages, enables the LLMs to perform complex tasks [8], perform in-context learning [14], etc. Recent advancements in LLMs have sparked simultaneous exploration and research into the applicability of these LLMs on a variety of different tasks, such as data labeling [19, 3, 40], text classification [39, 4], model explanation [5], etc. We add on to this emerging body of work and perform zero-shot general purpose counterfactual generation using LLMs.

¹ All code and data will be available at <link-to-be-inserted-after-blind-review>.

² <https://huggingface.co/tiiuae/falcon-40b>

3 FIZLE: Zero-shot LLM-guided Counterfactual Generation

In this section, we describe our main framework as shown in Figure 2. Following the causal explanation generation procedure in prior work [5], we use state of the art LLMs in an off-the-shelf manner, without any fine-tuning. We improve upon prior work [5] by expanding and broadening their pipeline into a more general framework that can work for tasks other than causal explanation. Note that in this paper, we formulate and evaluate our pipeline on the broad task of text classification, whereas formulations for other text tasks can be derived in a similar manner. To facilitate this we explain the following components in our framework:

Input Dataset and Other Task-specific Input The first component in our pipeline takes a task dataset as input and pre-processes it into tuples denoted by (x_i, \hat{y}_i) , where $x_i \in X$ denote a text sample in the input dataset X , and $\hat{y}_i \in \{0, 1, \dots, k\}$ denote the ground truth label of the corresponding input, in a k -class classification problem. Depending on the use-case, we also have black-box access to a text classification model $f(\cdot)$ whereby we get $f(x_i) = y_i$, which is the predicted label. In this case, we also build tuples of the form (x_i, y_i) for use in the generation step.

LLM as the Counterfactual Generator We leverage recent state-of-the-art LLMs as the counterfactual generators. Given that these models have been trained on vast amounts of textual data along with extensive instruction tuning, we assume that LLMs can learn to modify and perturb text input to *simulate* how human annotators generate counterfactuals for specific tasks [18]. For this, we use both proprietary models from OpenAI and open-source models from Meta AI, and use carefully crafted instructions and constraint prompts to generate the counterfactuals. Specifically, we use the following proprietary models via the OpenAI API where applicable:

- *text-davinci-003*: This model was a pre-cursor to OpenAI’s GPT-3.5.
- *GPT-3.5³*: Often referred to as ChatGPT. This is the model that has been explored in a variety of text applications. Specifically we use the *gpt-3.5-turbo* version as of January-February 2024.
- *GPT-4⁴*: This is the successor to GPT-3.5 and is known to be more capable. Specifically this model is purported to be able to understand complex instruction better, thereby making it highly suitable to our task of counterfactual generation. We use the *gpt-4* and *gpt-4-turbo* versions in our experiments.

Among the open-source models, we use two sizes of the Llama 2 model wherever applicable. These are:

- *Llama 2 7B*: This is the 7 billion parameter version of the model. We specifically use the ‘chat’ variant via Huggingface⁵.
- *Llama 2 13B*: This is the 13 billion parameter version of the model. Similar to the previous one, we use the ‘chat’ variant from Huggingface⁶.

Due to resource constraints, we were unable to use the largest 70B Llama 2 model.

Instruction and Constraint Prompt To generate the counterfactuals in a zero-shot manner using the chosen LLM, the prompt needs to have informative instructions and constraints to guide the generation. We generate two types of counterfactuals: (1) counterfactuals with different label from original: these are used in the counterfactual explanation experiments (see Section 5), and (2) counterfactuals with same label as original: these are used in the contrast set experiments (see Section 6). For setting (1), we experiment with two variants of the generation process: (i) *naive*: Here the LLM is directly prompted to generate a counterfactual, and (ii) *guided*: Here we use a two-step process - first leveraging the LLM to identify the important input features (i.e., words) that result in the predicted label, and then prompting the same LLM to edit a minimal set of those identified features to generate the counterfactual. We do this since generating a counterfactual that flips the label of the original input is intuitively more challenging and providing additional guidance to the LLM in the form of steps may potentially improve the quality of the generated counterfactuals. For ease of extraction of the generated rationales, we also specify an output constraint that allows easy parsing based on a regular expression string match.

4 Experimental Settings

We use our proposed FIZLE pipeline to generate counterfactuals and demonstrate these in two specific use-cases: (1) Counterfactual explanations for explaining decisions of black-box text classifiers and (2) Evaluating black-box text classification models via contrast sets. To facilitate this, here we go over the datasets used and the general experimental setup for all our generation and evaluation experiments:

4.1 Datasets

In this work, we focus on two broad categories of language tasks: text classification and natural language inference (NLI)⁷. For text classification we use two datasets: IMDB [24] for sentiment classification and AG News⁸ for news topic classification. For NLI, we use the SNLI dataset [25, 6]. This variety of datasets allows us to evaluate the LLM-generated counterfactuals over a variety of label situations from binary to multi-class.

The IMDB dataset⁹ consists of a total of 50k highly polar movie reviews from IMDB (Internet Movie Database). Each data instance consists of a text string comprising the review text, and a label, either ‘negative’ or ‘positive’. The AG News dataset consists of over 120k news articles, belonging to one of four news topics: ‘world’, ‘sports’, ‘business’ and ‘science/technology’. The Stanford Natural Language Inference (SNLI) dataset¹⁰ consists of 570k sentence pairs consisting of a premise and a hypothesis. Each premise-hypothesis pair is labeled with one of ‘entailment’, ‘contradiction’ or ‘neutral’ labels.

4.2 Experimental Setup

All experiments on open-source models were performed on two A100 GPUs with a total of 80G memory. For 13B Llama 2 models, we use 4-bit quantization using the optimal ‘nf4’ datatype [13]. For all LLMs, we use top_p sampling with $p = 1$, temperature $t = 0.4$

³ <https://platform.openai.com/docs/models/gpt-3-5-turbo>

⁴ <https://platform.openai.com/docs/models/gpt-4-turbo-and-gpt-4>

⁵ <https://huggingface.co/meta-llama/Llama-2-7b-chat-hf>

⁶ <https://huggingface.co/meta-llama/Llama-2-13b-chat-hf>

⁷ NLI here is also treated as a text classification task where the labels for each input are simply one of {entailment, neutral, contradiction}.

⁸ https://huggingface.co/datasets/ag_news

⁹ <https://huggingface.co/datasets/imdb>

¹⁰ <https://huggingface.co/datasets/snli>

and a repetition penalty of 1.1. We use PyTorch and fine-tuned models hosted on Huggingface for in both Sections 5 and 6.

5 Counterfactual Explanations via LLM-generated Counterfactuals

Explainability is a major challenge in many NLP applications such as text classification [35, 2, 9]. Although recent models involving pre-trained transformer based language models [44] have achieved or even exceeded human-level performance on several tasks [45, 46], most of these models are black-box by design and hence are not interpretable. Such models do not offer transparency on *why* it predicted a certain label, or even *what features in the input* resulted in the prediction. The ubiquity of these black-box classifiers necessitates the development of explanation frameworks and techniques that provide some degree of understanding into the decision-making function of the model [1]. Counterfactual explanations [28] give an insight into what *could have been* different in the input to change the output label predicted by the classifier. Gold-standard counterfactual generation requires human annotators and is also task-specific [20, 21], therefore making it an extremely expensive and labor-intensive endeavor. Therefore, we use our LLM-generated counterfactuals in the of counterfactual explanations for black-box text classifiers.

5.1 Methodology

To generate counterfactual explanations for a black-box text classifier $f(\cdot)$ that predicts $f(x_i) = y_i$, we use the tuple (x_i, y_i) in the generation step, thereby replacing the ground truth label in Figure 2 by the model-predicted label, since we aim to explain why the model predicted y_i for the input sample x_i . In our experiments, we use a DistilBERT model [38], fine-tuned on the specific task dataset as the black-box model we aim to explain. Note that since our counterfactual generation process is model-agnostic, the same procedure can be applied to any black-box classifier in place of DistilBERT. Inspired by prior work [5], we develop and experiment with two variants of FIZLE: (1) FIZLE_{naive}: which directly generates the counterfactual explanation, and (2) FIZLE_{guided}: which first extracts words that may have caused the predicted label, and then uses those selected words to generate a counterfactual explanation, in a two-step manner. We hypothesize that the two-step generation may result in more effective and better quality counterfactual explanations, due to the additional guidance provided to the LLM, analogous to prior work such as Chain of Thought [47]. We show the prompts used in both the variants in Table 1.

5.2 Evaluation Metrics

To evaluate the goodness of the counterfactual explanations generated by our zero-shot LLM-guided pipeline, we use a variety of evaluation metrics following prior work [48, 26, 5]. Ideally, the generated counterfactual explanations should be able to flip the label of the classifier, thereby showcasing what *could have* changed in the input that would flip the label of the classifier. Furthermore, counterfactual explanations should also be minimally edited samples of the input text, i.e., they should be as close as possible to the input sample both in the token space and the semantic space. To capture and evaluate these criteria, we use the following metrics:

Label Flip Score We use Label Flip Score to measure the effectiveness of the generated counterfactual explanations. For each input text x_i in the test split of the dataset, with correctly predicted label $f(x_i) = y_k$, we evaluate the corresponding LLM-generated counterfactual x_i^{cf} using the same black-box classifier $f(\cdot)$ and obtain a label for the counterfactual. For an effective counterfactual, the obtained label should be different from the original label y_k . Then Label Flip Score (LFS) is computed as:

$$LFS = \frac{1}{n} \sum_{i=1}^n \mathbb{1}[f(x_i) \neq f(x_i^{cf})] \times 100 \quad (1)$$

where n is the number of samples in the test set and $\mathbb{1}$ is the identity function.

Textual Similarity Counterfactual explanations generated by the LLMs should ideally be as ‘similar’ to the original input text as possible. To evaluate this similarity, we use two metrics: similarity of the text embeddings using the Universal Sentence Encoder (USE) [10] in the latent space, and a normalized Levenshtein distance [22] to measure word edits in the token space. The semantic similarity using the embeddings of the original input and the generated counterfactual is computed as the inner product of the original and the counterfactual embeddings, averaged over the test dataset:

$$sim_{semantic} = \frac{1}{n} \sum_{i=1}^n Enc(x_i) \cdot Enc(x_i^{cf}) \quad (2)$$

where $Enc(\cdot)$ refers to the Universal Sentence Encoder, n is the number of samples in the test set.

Levenshtein distance [22] between two strings is defined as the minimum number of single character edits that are required to convert one string to another. To measure the distance between the original input text and the generated counterfactual in the token space we use a normalized Levenshtein distance, further averaged over the test dataset. This is computed as:

$$edit_dist = \frac{1}{n} \sum_{i=1}^n \frac{lev(x_i, x_i^{cf})}{\max(|x_i|, |x_i^{cf}|)} \quad (3)$$

where $|x_i|$ and $|x_i^{cf}|$ refer to the length of x_i and x_i^{cf} respectively, $lev(\cdot, \cdot)$ refers to the Levenshtein distance, and n is the number of samples in the test set.

5.3 Baselines

Similar to other counterfactual generation methods [26, 48], we compare our proposed FIZLE pipeline with three representative baselines from three categories of similar works: (i) BAE [17] is a recent adversarial attack method that uses masked language modeling with BERT to perturb the input text by replacing masked words; (ii) CheckList [36] is a method for behavioral testing of NLP models via test cases generated by template-based methods as well as masked language models like RoBERTa; (iii) Polyjuice [48] is a recent counterfactual generation method that uses an auxiliary language model (such as GPT-2) to generate diverse counterfactuals. Note that unlike these baselines, our FIZLE pipeline does not require any auxiliary model, language model or dataset, thereby enabling a completely zero-shot generation.

Framework Variant	Prompt Structure
$\text{FIZLE}_{\text{guided}}$	<p>Step 1: In the task of <code><task on task-dataset></code>, a trained black-box classifier correctly predicted the label '<code><y_i></code>' for the following text. Explain why the model predicted the '<code><y_i></code>' label by identifying the words in the input that caused the label. List ONLY the words as a comma separated list.\n—\nText: <code><x_i></code></p> <p>Step 2: Generate a counterfactual explanation for the original text by ONLY changing a minimal set of the words you identified, so that the label changes from '<code><y_i></code>' to '<code><y_{cf}></code>'. Use the following definition of 'counterfactual explanation': "A counterfactual explanation reveals what should have been different in an instance to observe a diverse outcome." Enclose the generated text within <code><new></code> tags.</p>
$\text{FIZLE}_{\text{naive}}$	<p>In the task of <code><task on task-dataset></code>, a trained black-box classifier correctly predicted the label '<code><y_i></code>' for the following text. Generate a counterfactual explanation by making minimal changes to the input text, so that the label changes from '<code><y_i></code>' to '<code><y_{cf}></code>'. Use the following definition of 'counterfactual explanation': "A counterfactual explanation reveals what should have been different in an instance to observe a diverse outcome." Enclose the generated text within <code><new></code> tags.\n—\nText: <code><x_i></code>.</p>

Table 1. Prompt structure for the two variants of our pipeline when used for counterfactual explanations. `<task on task-dataset>` denotes the task description such as "natural language inference on the SNLI dataset". `<xi>` denotes the input text. `<yi>` denotes the label output by the black-box classifier for the input text `<xi>`. `<ycf>` denotes the desired label for the counterfactual text (any label other than `<yi>`).

Model	IMDB			AG News			SNLI		
	LFS ↑	Sem. Sim. ↑	Edit Dist. ↓	LFS ↑	Sem. Sim. ↑	Edit Dist. ↓	LFS ↑	Sem. Sim. ↑	Edit Dist. ↓
BAE [17]	79.6	0.99	0.044	25	0.97	0.063	74.4	0.95	0.054
CheckList [36]	2.6	0.99	0.013	1.6	0.92	0.083	3	0.96	0.036
Polyjuice [48]	96.86	0.25	0.884	72.64	0.22	0.749	<u>95.8</u>	0.74	0.367
text-davinci-003 (<i>guided</i>)	95.78	0.87	0.186	54.9	0.84	0.153	39.6	0.83	0.179
text-davinci-003 (<i>naive</i>)	95.39	0.88	0.219	66.4	0.88	0.252	40.2	0.89	0.218
ChatGPT-3.5 (<i>guided</i>)	78.52	0.91	0.126	30.55	0.95	0.084	32.47	0.89	0.102
ChatGPT-3.5 (<i>naive</i>)	59.19	0.88	0.236	49	0.91	0.325	56.39	0.92	0.182
GPT-4 (<i>guided</i>)	97.2	0.89	0.142	82.39	0.65	0.232	73.6	0.88	0.153
GPT-4 (<i>naive</i>)	<u>99.6</u>	0.87	0.226	<u>84.39</u>	0.65	0.278	78	0.88	0.152
Llama 2 7B (<i>guided</i>)	76.64	0.66	0.546	51.11	0.77	0.244	36.82	0.74	0.304
Llama 2 7B (<i>naive</i>)	64.7	0.59	0.68	35.25	0.7	0.492	58.33	0.62	0.577
Llama 2 13B (<i>guided</i>)	51.11	0.7	0.533	51.65	0.77	0.266	50.2	0.67	0.495
Llama 2 13B (<i>naive</i>)	66.67	0.52	0.715	37.63	0.58	0.606	59.95	0.55	0.621

Table 2. Evaluation results of both variants of our FIZLE framework in comparison to baselines: BAE [17], CheckList [36] and Polyjuice [48]. We report the Label Flip Score (LFS), semantic similarity (Sem. Sim) and normalized Levenshtein distance (Edit Dist.). Best LFS scores for each dataset are underlined.

5.4 Results: Effectiveness of Generated Counterfactual Explanations

Following the experimental setup described above, we evaluate the generated counterfactuals to explain black-box classifiers for the three datasets, and compare to baselines. We show these quantitative results in Table 2. For each LLM, we evaluate both variants of our framework: `FIZLEguided` and `FIZLEnaive`. For effective and good quality counterfactual explanations, ideally we would expect high values of LFS and semantic similarity with low values of edit distance. Overall, we see varied performance of the LLMs and the two variants across the different tasks. Similar to other counterfactual generation works [26], we see an obvious trade-off between the Label Flip Score and the semantic similarity. This is intuitive since the more the generated counterfactual deviates from the original input text, higher the chances are for it to be a successful counterfactual for the original input (i.e., it would result in a label flip). Among the three baselines, we see CheckList fails completely in generating counterfactual explanations. We see satisfactory performance by BAE, except for the AG News dataset. For Polyjuice, even though the LFS scores are high, the poor textual similarity scores imply that the counterfactuals generated are not good quality and deviate from the input text significantly.

For most of the LLMs we evaluated, `FIZLEguided` outperforms `FIZLEnaive` on the IMDB and AG News datasets. This may imply that the additional ‘guidance’ provided by identifying the input words before the counterfactual explanation generation step enables the generation of better counterfactuals. Interestingly, we do not see this trend for GPT-4, where the *naive* variant performs better than the *guided* one. This might be due to the fact that GPT-4 is extremely good at understanding complex tasks and instructions [8], and the additional feature extraction step in the *guided* variant does not provide additional useful guidance, and perhaps even confuses the LLM. GPT-4 when used in the *naive* variant of our pipeline, has the best performance for zero-shot counterfactual explanation generation, in terms of LFS. For natural language inference on the SNLI dataset, we see all LLMs struggle to generate good counterfactual explanations. GPT-4 performs well, possibly owing to its instruction and textual understanding capabilities [8], but the best performance is by the Polyjuice baseline. This poor performance of LLMs particularly on the SNLI dataset is further evidence towards LLMs struggling with inference and reasoning. This gap in the capabilities of recent LLMs on reasoning tasks has been observed by several recent efforts as well [34, 43].

Lastly, we see the open-source models Llama 2 7B and 13B struggle to generate zero-shot counterfactual explanations with small number of edits, thus resulting in very high edit distances. The Llama 2 models struggle to keep the generated counterfactuals semantically similar to the original input, implying they either make too many edits to the input text, or output some unrelated, low-quality text that does not conform to the instructions provided in the prompt.

6 Evaluating Models via LLM-generated Counterfactuals

Deep learning models such text classification models are often trained in a supervised manner using labeled training sets, and then evaluated on a hold-out test set. Such train-test splits of data usually arise from the same corpus that has same or similar sources and annotation guidelines. Therefore, in essence, standard evaluation using such hold-out test sets measure merely the in-distribution perfor-

mance of the model, while in reality, the same model may demonstrate sub-par performance on out-of-distribution or in-the-wild test data [16]. To alleviate this issue to some degree, approaches such as evaluating using challenge sets or robustness to label-preserving perturbations, etc. have been explored by the community. One specific method of stress-testing such models is via *contrast sets* [16]. A contrast set $C(x)$ is essentially a sample of points around a data point x , that is close to the local ground truth decision boundary. Samples in $C(x)$ may have same or different ground truth label as x . In practice, $C(x)$ can be a set of samples that are ‘close’ to x , i.e., have minimal edit distance from x , yet be ‘challenging’ for a trained model to classify. In the original contrast sets work, the authors advocate for an evaluation paradigm where dataset authors themselves create and release such contrast sets for model evaluation. However, we note that this is highly infeasible in practice, given the cost of expert creation of such challenging data points. Therefore, automated methods for designing such challenging evaluation sets in the form of contrast sets are highly desirable, albeit at the expense of trading off expert insights. One such automated method for developing contrast sets to evaluate models is that of counterfactual examples, as demonstrated by previous work [48]. Motivated by the effectiveness of counterfactuals as contrast sets in prior work [48], we envision the use of LLM-generated contrast sets as well for the same purpose of model evaluation. Here we describe the methodology for the generation and evaluation of such contrast sets using LLMs in a zero-shot manner.

6.1 Methodology

For generating the contrast sets, we use the same LLMs as used in Section 5, except `text-davinci-003`¹¹, and prompt the LLM to generate counterfactuals in a zero-shot manner using the input text and ground truth label tuple (x_i, \hat{y}_i) . Unlike [48], we do not use human annotators to label the generated counterfactuals. Therefore, differing from [48], we only focus on counterfactuals that have the same label as the original input, and use these as contrast sets. We make this choice since the lack of human annotation and lack of step-by-step guidance (such as in `FIZLEguided`) would make it harder to validate whether the edits performed by the LLM are actually label flipping or not. Instead, we guide the generation process via the instruction in the prompt. We use the following prompt to perform the generation:

‘You are a robustness checker for a machine learning algorithm. In the task of `<taski>`, the following data sample has the ground truth label `<ŷi>`. Make minimal changes to the data sample to create a more challenging data point while keeping the ground truth label the same. Text: `<xi>`’

where, $task_i$ is the description of the task, such as “sentiment classification”, x_i is the input text, and \hat{y}_i is the ground truth label.

6.2 Evaluation Metrics

For evaluating the goodness of the generated counterfactuals as contrast sets, we compare the accuracy of the target model $f(\cdot)$ on both the original test set and the generated contrast set. We also measure the consistency,

¹¹ since OpenAI deprecated this model by the time we performed these contrast set generation experiments.

Counterfactual generator	IMDB					SNLI					AG News				
	Original Test Acc.	C.s. Acc. ↓	Edit Dist. ↓	Sem. Sim. ↑	Cons. %↓	Original Test Acc.	C.s. Acc. ↓	Edit Dist. ↓	Sem. Sim. ↑	Cons. %↓	Original Test Acc.	C.s. Acc. ↓	Edit Dist. ↓	Sem. Sim. ↑	Cons. %↓
Polyjuice [48]	94.3	84.9	-	-	76.1	86.5	72.3	-	-	56.4	-	-	-	-	-
Expert [16]	96.31	84.84	0.136	0.939	81.56	-	-	-	-	-	-	-	-	-	-
GPT-3.5	93.35	88.82	0.162	0.931	85.22	86.25	57.42	0.175	0.908	53.53	94.6	93.42	0.287	0.883	92.07
GPT-4		92.65	0.157	0.942	90.95		73.01	0.277	0.841	68.82		94	0.352	0.855	93
Llama 2 7B		87.32	0.559	0.728	83.74		63.88	0.382	0.782	57.87		92.94	0.438	0.808	91.72
Llama 2 13B		84.76	0.580	0.710	82.2		48.6	0.476	0.738	43.21		93.5	0.427	0.808	92.03

Table 3. Performance of FIZLE-generated counterfactuals as contrast sets. C.s. Acc. refers to accuracy on the generated contrast sets, Original Test Acc. is the test accuracy on the corresponding paired original samples. Sem. Sim. refers to semantic similarity as computed by Eq. 2, Edit. Dist. is the token level distance as computed by Eq. 3, Cons. (%) is the consistency as computed by Eq. 4.

$$consistency = \frac{1}{n} \sum_{i=1}^n \mathbb{1}[f(x_i) = \hat{y}_i f(x_i^{cs}) = \hat{y}_i^{cs}] \times 100 \quad (4)$$

where x_i^{cs} is the LLM-generated contrast set for the original input x_i , \hat{y}_i^{cs} is the ground truth label for the contrast set example and n is the number of test samples. Consistency measures the percentage of times when the model correctly classifies both the original and the contrast set example. Like the previous set of experiments, we want the generated counterfactuals (or contrast sets) to be as close to the original text input as possible, i.e., the edits should ideally be minimal. Therefore, we capture the textual similarity again in the token space via Equation 3 and the latent space via Equation 2.

6.3 Baselines

Since there is not much work on contrast sets, we have a limited set of baselines here. We use the original expert-created contrast sets for the IMDB dataset from the original work [16]. This consists of 488 original test data samples, and 488 contrast samples created by the dataset experts. Furthermore, we use Polyjuice-generated counterfactuals [48] as contrast sets for comparison.

6.4 Results: Effectiveness of Counterfactuals as Contrast Sets

We use the same DistilBERT models as in Section 5 that are fine-tuned for each of the 3 tasks (IMDB, SNLI and AG News). We evaluate each of these 3 models on both the original test set and the counterfactual one (i.e., the contrast sets) and show these results in Table 3. We obtain the performance values for Polyjuice-generated contrast sets from the original paper [48]. For the ‘Expert’ baseline, the IMDB contrast sets are created by human experts in [16]. Unfortunately, there does not exist any expert created contrast set for SNLI and AG News datasets. As evident from the test accuracies on both the original test set and the counterfactual one, we see a consistent decrease in performance on the generated counterfactuals over the original samples. The performance drop for the AG News dataset seems to be the least while interestingly, we see the highest performance drop on contrast sets for the SNLI dataset. Furthermore, we see GPT-3.5 and GPT-4 are able to create contrast sets with high degree of semantic similarity and low edit distance, thus being more desirable over Llama 2 generated contrast sets. Overall, the drop in accuracy and the consistency values seem analogous to similar results in literature (average drop in classification accuracy of around 6.8% according to [48]) that use human-generated contrast sets for evaluation [16, 48].

While this is promising, we do note the ethical concerns surrounding this: LLM-generated contrast sets may induce pre-existing biases

that can propagate further bias and errors through evaluation and subsequent model improvement steps. One way to effectively use LLM-generated contrast sets is by broadly identifying the failure models of the model via probing the model using the LLM-generated contrast sets, and *then* employing human annotators or data creators to hone in on that specific failure mode to either generate more contrast sets or counterfactually augmented training data to fill the identified gap. Such a combined method would greatly reduce costs while still being effective in terms of model evaluation and development.

7 Conclusion & Future Work

In this paper, we tackle the novel task of zero-shot general purpose counterfactual generation. To this end, we propose the paradigm of leveraging large language models as pseudo-oracles in order to generate plausible, high quality, and effective counterfactuals. Such a framework is zero-shot, requiring no training data or auxiliary models or even in-context learning. We validate our hypothesis via demonstrating the use and effectiveness of LLM-generated counterfactuals on two broad NLP model development tasks: (1) counterfactual explanation of black-box text classifiers, and (2) evaluation of black-box text classification models via contrast sets. Our results are promising and we see benefits to using our proposed FIZLE framework across the two use-cases and three downstream tasks. Our findings suggest the effective use of LLM-generated counterfactuals for explaining black-box NLP models, as well as potentially identifying failure models of NLP models via evaluation with contrast sets. We further discuss implications and how hybrid human-and-AI methods may benefit from our exploration.

Future work may investigate modifications to this proposed framework such as generating contrast sets with opposite labels. More effort can also be put into validating the faithfulness of the generated counterfactual explanation and correctness of generated contrast sets. Another gap that can be addressed is how to improve the reasoning capabilities of the LLMs used in the pipeline, to improve explanation performance on the NLI task. Since one of the challenges in our method was to ensure that the generated text is actually a counterfactual, devising ways and human annotations to ensure more conformity of the generated counterfactual explanations to the definition of ‘counterfactual explanation’ is also an area that can be improved. Finally, apart from these two use-cases of counterfactuals, LLM-generated counterfactuals can also be evaluated in tasks such as model training or improvement, uncovering biases in model predictions, incorporating fairness into model predictions, etc.

Acknowledgements

This work is supported by the DARPA SemaFor project (HR001120C0123), Army Research Office (W911NF2110030) and

Army Research Lab (W911NF2020124). The views, opinions and/or findings expressed are those of the authors.

References

- [1] S. Ali, T. Abuhmed, S. El-Sappagh, K. Muhammad, J. M. Alonso-Moral, R. Confalonieri, R. Guidotti, J. Del Ser, N. Díaz-Rodríguez, and F. Herrera. Explainable artificial intelligence (xai): What we know and what is left to attain trustworthy artificial intelligence. *Information Fusion*, 99:101805, 2023.
- [2] P. Atanasova, J. G. Simonsen, C. Lioma, and I. Augenstein. A diagnostic study of explainability techniques for text classification. *arXiv preprint arXiv:2009.13295*, 2020.
- [3] P. Bansal and A. Sharma. Large language models as annotators: Enhancing generalization of nlp models at minimal cost. *arXiv preprint arXiv:2306.15766*, 2023.
- [4] A. Bhattacharjee and H. Liu. Fighting fire with fire: can chatgpt detect ai-generated text? *ACM SIGKDD Explorations Newsletter*, 25(2):14–21, 2024.
- [5] A. Bhattacharjee, R. Moraffah, J. Garland, and H. Liu. Llms as counterfactual explanation modules: Can chatgpt explain black-box text classifiers? *arXiv preprint arXiv:2309.13340*, 2023.
- [6] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*, 2015.
- [7] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.
- [8] S. Bubeck, V. Chandrasekaran, R. Eldan, J. Gehrke, E. Horvitz, E. Kamar, P. Lee, Y. T. Lee, Y. Li, S. Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.
- [9] O.-M. Camburu, T. Rocktäschel, T. Lukasiewicz, and P. Blunsom. e-snli: Natural language inference with natural language explanations. *Advances in Neural Information Processing Systems*, 31, 2018.
- [10] D. Cer, Y. Yang, S.-y. Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, et al. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*, 2018.
- [11] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.
- [12] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, E. Li, X. Wang, M. Dehghani, S. Brahma, et al. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*, 2022.
- [13] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*, 2023.
- [14] Q. Dong, L. Li, D. Dai, C. Zheng, Z. Wu, B. Chang, X. Sun, J. Xu, and Z. Sui. A survey for in-context learning. *arXiv preprint arXiv:2301.00234*, 2022.
- [15] L. Gao, S. Biderman, S. Black, L. Golding, T. Hoppe, C. Foster, J. Phang, H. He, A. Thite, N. Nabeshima, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.
- [16] M. Gardner, Y. Artzi, V. Basmova, J. Berant, B. Bogin, S. Chen, P. Dasigi, D. Dua, Y. Elazar, A. Gottumukkala, et al. Evaluating models’ local decision boundaries via contrast sets. *arXiv preprint arXiv:2004.02709*, 2020.
- [17] S. Garg and G. Ramakrishnan. Bae: Bert-based adversarial examples for text classification. *arXiv preprint arXiv:2004.01970*, 2020.
- [18] F. Gilardi, M. Alizadeh, and M. Kubli. Chatgpt outperforms crowd workers for text-annotation tasks. *Proceedings of the National Academy of Sciences*, 120(30):e2305016120, 2023.
- [19] X. He, Z. Lin, Y. Gong, A. Jin, H. Zhang, C. Lin, J. Jiao, S. M. Yiu, N. Duan, W. Chen, et al. Anollm: Making large language models to be better crowdsourced annotators. *arXiv preprint arXiv:2303.16854*, 2023.
- [20] D. Kaushik, E. Hovy, and Z. C. Lipton. Learning the difference that makes a difference with counterfactually-augmented data. *arXiv preprint arXiv:1909.12434*, 2019.
- [21] D. Khashabi, T. Khot, and A. Sabharwal. More bang for your buck: Natural perturbation for robust question answering. *arXiv preprint arXiv:2004.04849*, 2020.
- [22] V. I. Levenshtein et al. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union, 1966.
- [23] X. Liu, T. Sun, J. He, J. Wu, L. Wu, X. Zhang, H. Jiang, Z. Cao, X. Huang, and X. Qiu. Towards efficient nlp: A standard evaluation and a strong baseline. *arXiv preprint arXiv:2110.07038*, 2021.
- [24] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. 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, June 2011. Association for Computational Linguistics. URL <http://www.aclweb.org/anthology/P11-1015>.
- [25] B. MacCartney and C. D. Manning. Modeling semantic containment and exclusion in natural language inference. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 521–528, 2008.
- [26] N. Madaan, I. Padhi, N. Panwar, and D. Saha. Generate your counterfactuals: Towards controlled counterfactual generation for text. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 13516–13524, 2021.
- [27] N. Madaan, S. Bedathur, and D. Saha. Plug and play counterfactual text generation for model robustness. *arXiv preprint arXiv:2206.10429*, 2022.
- [28] C. Molnar. *Interpretable machine learning*. Lulu. com, 2020.
- [29] OpenAI. Gpt-4 technical report. *arXiv*, pages 2303–08774, 2023.
- [30] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.
- [31] G. Penedo, Q. Malartic, D. Hesslow, R. Cojocaru, A. Cappelli, H. Alobeidli, B. Pannier, E. Almazrouei, and J. Launay. The refined-web dataset for falcon llm: outperforming curated corpora with web data, and web data only. *arXiv preprint arXiv:2306.01116*, 2023.
- [32] L. Qin, A. Bosselut, A. Holtzman, C. Bhagavatula, E. Clark, and Y. Choi. Counterfactual story reasoning and generation. *arXiv preprint arXiv:1909.04076*, 2019.
- [33] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1 (8):9, 2019.
- [34] J. W. Rae, S. Borgeaud, T. Cai, K. Millican, J. Hoffmann, F. Song, J. Aslanides, S. Henderson, R. Ring, S. Young, et al. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*, 2021.
- [35] M. T. Ribeiro, S. Singh, and C. Guestrin. "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, 2016.
- [36] M. T. Ribeiro, T. Wu, C. Guestrin, and S. Singh. Beyond accuracy: Behavioral testing of nlp models with checklist. *arXiv preprint arXiv:2005.04118*, 2020.
- [37] M. Robeer, F. Bex, and A. Feelders. Generating realistic natural language counterfactuals. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3611–3625, 2021.
- [38] V. Sanh, L. Debut, J. Chaumond, and T. Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.
- [39] X. Sun, X. Li, J. Li, F. Wu, S. Guo, T. Zhang, and G. Wang. Text classification via large language models. *arXiv preprint arXiv:2305.08377*, 2023.
- [40] Z. Tan, A. Beigi, S. Wang, R. Guo, A. Bhattacharjee, B. Jiang, M. Karami, J. Li, L. Cheng, and H. Liu. Large language models for data annotation: A survey. *arXiv preprint arXiv:2402.13446*, 2024.
- [41] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.
- [42] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.
- [43] K. Valmeekam, A. Olmo, S. Sreedharan, and S. Kambhampati. Large language models still can’t plan (a benchmark for llms on planning and reasoning about change). *arXiv preprint arXiv:2206.10498*, 2022.
- [44] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- [45] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.
- [46] A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman. Superglue: A stickier benchmark for general-

- purpose language understanding systems. *Advances in neural information processing systems*, 32, 2019.
- [47] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.
- [48] T. Wu, M. T. Ribeiro, J. Heer, and D. S. Weld. Polyjuice: Generating counterfactuals for explaining, evaluating, and improving models. *arXiv preprint arXiv:2101.00288*, 2021.