Robust Text Perturbation using Sequence-to-Sequence Pre-Training

Nishtha Madaan
IBM Research - India
nishthamadaan@in.ibm.com

Diptikalyan Saha
IBM Research - India
diptsaha@in.ibm.com

Srikanta Bedathur
Indian Institute of Technology Delhi
srikanta@cse.iitd.ac.in

Abstract

Large Transformer-based models have shown great performance in sequence-to-sequence tasks such as machine translation, text summarization etc. While these models perform well on the original task they have been trained on, it is hard to use them for a new but related task. We propose CASPer, a framework to perturb the input-output behavior of the original pre-trained sequence-to-sequence model. CASPer learns a perturbation parameter at test time to modify the behavior of pre-trained model and generates samples that have target characteristics. We apply this framework on a pre-trained text summarization model to alter a given input text such that the generated text has a changed sentiment or other attributes. In experiments, we show that CASPer effectively generates controlled text that preserve the original content, are fluent, diverse and follow the steering provided by the attribute model.

1 Introduction

Large Transformer-based language models (Vaswani et al., 2017; Radford et al., 2019; Lewis et al., 2019; Radford et al., 2018) have shown great success in sequence-to-sequence tasks such as machine translation, text summarization etc. While these models perform well on the original task they have been trained on, it is hard to control them to modify their behavior in order to perform a new but related task. For instance, we would like to use a text summarization model for generating summaries of a given text while also controlling the sentiment, style, or toxicity of the generated output.

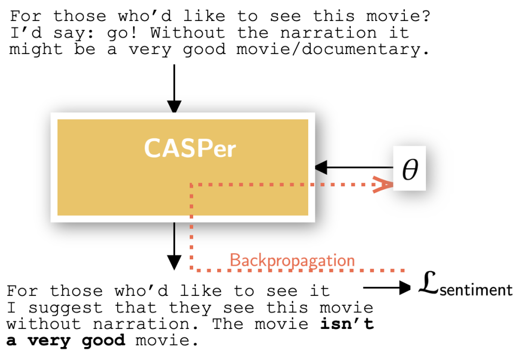
For those who'd like to see this movie? I'd say: go! Without the narration it might be a very good movie/documentary.
CASPer
θ
Backpropagation
$\mathcal{L}_{\text{sentiment}}$
For those who'd like to see it I suggest that they see this movie without narration. The movie **isn't a very good** movie.

Figure 1: Overview: CASPer takes an input text and a perturbation parameter θ and returns a perturbed text output. The loss $\mathcal{L}_{\text{sentiment}}$ provides a steering signal, for instance, to make the sentiment of the text negative. The backpropagation is used to update the perturbation parameter θ.

There has been a rise in the interest in the task of text perturbation. One such work is Checklist (Ribeiro et al., 2020) which provides perturbation goal at test time, for instance, to change the sentiment of a given text. However, this is achieved by using pre-existing templates and dictionaries which require significant domain knowledge. Another related work is GYC (Madaan et al., 2021) which attempts to generate controlled perturbations by making use of GPT-2 decoder. GYC adopts a two-step process where in the first step, it reconstructs the given input text via gradient descent on a learnable parameter. In the second step, it then fine-tunes this learned parameter based on the target attribute model to generate controlled perturbations. However we find that GYC reconstructs well only on short and simple texts and its success depends on the random seed text at the start of the gradient descent. Thus, in general, it fails to perform reconstruction on longer and more complex texts. While GYC provides a useful conceptual solution to the problem of controlled text generation without templates and dictionaries, its lack of robustness hinders its wider use.

In contrast, our proposed model, CASPer, makes use of the auto-encoding framework of BART (Lewis et al., 2019) to infer the representation of the input text rather than through gradient descent. In this way, CASPer can accurately and reliably reconstruct the given input text via the GPT-2 decoder. Through our model CASPer, we seek to control the output of a pre-trained sequence-to-sequence model given an input text such that the output text possesses some target characteristics or simply attributes as shown in Figure 1. To achieve this, we introduce a learnable parameter θ which is added as a perturbation to one of the hidden layers of the pre-trained sequence-to-sequence model. We assume that an attribute-model is available which provides a score on how well the target characteristics are present in the output text. Hence, given an input text, we train the learnable parameter θ to control the mapping from the input text to the output text with the objective of maximizing the score for the target attributes. As shown in Figure 1, the input text having a positive sentiment class has been modified to maximize target attributes, negative sentiment in this case. Crucially, our model is a plug-and-play model which requires no task-specific training of the sequence-to-sequence model for a given target characteristic. Our perturbation parameter θ is learned per-input text at test time to maximize the target attribute in the output text.

In this work, we implement this idea using a pre-trained summarization model BART. Given an input text, we seek to obtain a perturbation of the model so that target attribute is added to the original text. In our experiments we take two kinds of attribute models for controlling the generations. The first is a sentiment classifier and the second is a classifier which returns a true label when a named entity (such as location, person or organization) are present in the generated text. In experiments, we show that our generations are consistent with the control signals from the attribute models. We show that our generations are fluent and effectively preserve content. Because our generation process uses a GPT-2 decoder and samples the output text token by token, hence by drawing multiple text samples, we can obtain diverse set of perturbations for the given input. We also show that our sample diversity outperforms the baselines.

In summary, 1) We propose CASPer, a controlled plug and play perturbation framework which takes an input text and perturbs it to generate target characteristics at test time, 2) We propose an implementation which makes use of BART as a sequence-to-sequence pre-trained model thus enabling fluent and diverse generations, 3) In experiments, we show that we outperform the baselines in generating perturbed text on metrics that measure content preservation, fluency and diversity.

2 Method

In this section we describe our method for generating controlled text. For a given text $\mathbf{x}$ and an attribute model $p(a|\mathbf{y})$, the task is to generate K text perturbations $Y = \{\mathbf{y}^k\}_{k=1}^K$ that maximize $p(a|\mathbf{y})$ for the target attribute a. In other words, our goal is to obtain samples from the true distribution of $p(\mathbf{y}|a, \mathbf{x})$. To achieve this, will first approximate $p(\mathbf{y}|a, \mathbf{x})$ by learning a distribution $q_\theta(\mathbf{y})$. Although the true distribution $p(\mathbf{y}|a, \mathbf{x})$ is not available, using Bayes' rule, we can write $p(\mathbf{y}|a, \mathbf{x})$ to be proportional to the joint distribution $p(\mathbf{y}, a|\mathbf{x})$. We assume that this joint distribution can be factorized as $p(a|\mathbf{y})p(\mathbf{y}|\mathbf{x})$ where $p(\mathbf{y}|\mathbf{x})$ shall be the BART text summarization model and $p(a|\mathbf{y})$ shall be the given attribute model. Thus we can write the true distribution $p(\mathbf{y}|a, \mathbf{x}) \propto p(a|\mathbf{y})p(\mathbf{y}|\mathbf{x})$. As $p(\mathbf{y}|\mathbf{x})$ is taken to be a pre-trained BART text summarization model, each token $\mathbf{y}_t$ in the output text is modeled auto-regressively as $p(\mathbf{y}|\mathbf{x}) = \prod_t p(\mathbf{y}_t|\mathbf{y}_{<t}, \mathbf{x})$. With these, we get:

$$p(\mathbf{y}|a, \mathbf{x}) \propto p(a|\mathbf{y}) \prod_{t=1}^{T} p(\mathbf{y}_t|\mathbf{y}_{<t}, \mathbf{x}).$$

The distribution that we learn, $q_\theta(\mathbf{y})$, is also modeled auto-regressively as $q_\theta(\mathbf{y}) = \prod_{t=1}^{T} q_\theta(\mathbf{y}_t|\mathbf{y}_{<t})$. To learn the distribution $q_\theta(\mathbf{y})$, we minimize the KL divergence between the true distribution $p(\mathbf{y}|a, \mathbf{x})$

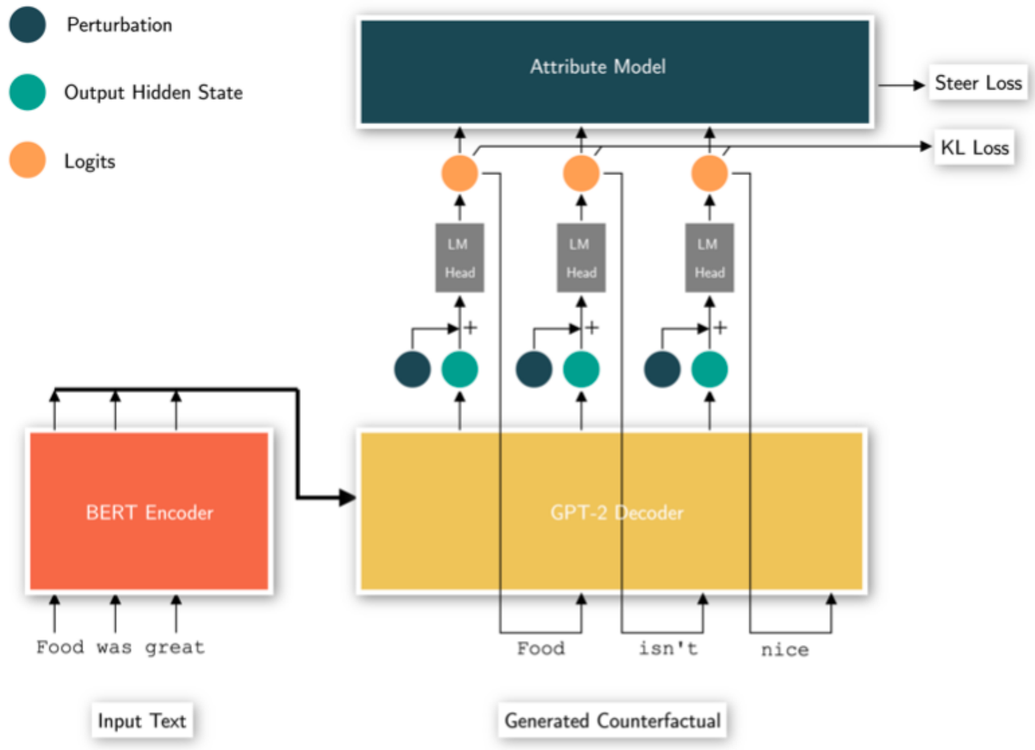

Figure 2: Model Architecture of CASPer. The BERT encoder (shown in orange) takes the input text and returns a representation of the input text as a set of vectors. These are provided to the GPT-2 decoder for cross-attention. At each step of decoding, the decoder returns an output hidden state (shown as green circle). To this, we add a perturbation matrix (shown as blue circle). The perturbed hidden state is then provided to the language model head to obtain the logits over the vocabulary for sampling the next token. These logits are also provided to the attribute model for computing the loss for steering the generation. The KL divergence between the perturbed logits and the unperturbed logits is also computed to keep the semantic content of the generated text close to the original input text.

and our learned approximation $q_\theta(\mathbf{y})$. This KL divergence $D_{\mathrm{KL}}(q_\theta(\mathbf{y}) \parallel p(\mathbf{y}|a, \mathbf{x}))$ objective can be simplified as the following training loss:

$$-\mathbb{E}_{\mathbf{y}} \log p(a|\mathbf{y}) + \sum_{t=1}^{T} \mathbb{E}_{\mathbf{y}_{<t}} D_{\mathrm{KL}}(q_\theta(\mathbf{y}_t|\mathbf{y}_{<t}, \mathbf{x}) || p(\mathbf{y}_t|\mathbf{y}_{<t}, \mathbf{x})).$$

Here the second term ensures that the content is close to that of the original summarization model, while the first term steers the content towards the given target attribute a.

2.1 Transformer-based Implementation

In the framework that we introduced above, we implement $q_\theta(\mathbf{y})$ by taking a pre-trained BART (Lewis et al., 2019) summarization model with a learnable perturbation parameter θ. We choose BART because it is a rich auto-encoding model that can represent complex input text $\mathbf{x}$ and return samples of summaries $\mathbf{y}$ of the given input text. BART auto-encoding framework is implemented using a BERT encoder and and GPT-2 decoder. To implement $q_\theta(\mathbf{y})$, we would like to perturb and control the decoding process of GPT-2 part of BART. For this, our learnable parameter θ is decomposed into a learnable parameter θ_t for every token position t. This is implemented similarly to PPLM (Dathathri et al., 2019) in which θ_t is added to the output layers of the GPT-2 decoder. In this way, we get a steerable auto-encoding model, CASPer. We describe the detailed process of generating text with this model below.

Algorithm 1 CASPer

Input Text $\mathbf{x}$, Output text length T, Target attribute value a, Attribute Classifier Model $f_{\text{attribute}}(\text{text})$ Generated Sample $\mathbf{y}$

```
# 1. Initialize perturbation as a learned parameter
$\boldsymbol{\theta} = \{\theta\}_{t=1}^{T}$

# 2. Encode input text using BART encoder
$\mathbf{e} = f_{\text{BERT}}(\mathbf{x})$

# 3. Initialize a start token
$\mathbf{y}_0 = \texttt{start\_token}$

# 3. Run decoder auto-regressively token-by-token
for $t = 1$ to $T$ do
    # 3.1 Get output hidden state from GPT-2
    $\mathbf{h}_t = f_{\text{GPT-2}}(\mathbf{e}, \mathbf{y}_{<t})$

    # 3.2 Add perturbation vector to the output hidden state
    $\tilde{\mathbf{h}}_t = \mathbf{h}_t + \theta_t$

    # 3.3 Obtain next token logits from the perturbed hidden state
    $\tilde{\mathbf{o}}_t = f_{\text{LM}}(\tilde{\mathbf{h}}_t)$

    # 3.4 Obtain next token logits from the unperturbed hidden state
    $\mathbf{o}_t = f_{\text{LM}}(\mathbf{h}_t)$

    # 3.5 Compute KL divergence between perturbed and unperturbed logits
    $\mathcal{L} = \mathcal{L} + D_{\text{KL}}(\tilde{\mathbf{o}}_t || \mathbf{o}_t)$

    # 3.6 Sample the next token from the perturbed distribution
    $\mathbf{y}_t \sim \text{Categorical}(\tilde{\mathbf{o}}_t)$
end for

# 4. Compute Attribute-Based Loss for Steering
$\mathcal{L} = \mathcal{L} + \texttt{CE}(a, f_{\text{attribute}}(\mathbf{y}_{1:T}))$
```

Given an input text, an attribute model and a target attribute label value a, we first initialize perturbation θ as a learnable parameter. Next, we encode the input text using the BERT encoder f_{BERT} of BART and obtain a encoding $\mathbf{e}$ of the input text. This is then provided to the GPT-2 decoder $f_{\text{GPT-2}}$ along with a start token y_0. The GPT-2 then generates the text auto-regressively. At every step t in this auto-regressive generation, we get an output hidden-state $\mathbf{h}_t$ from the GPT-2 decoder. We add the perturbation θ_t to this hidden-state to get a perturbed hidden state $\tilde{\mathbf{h}}_t$. We use this perturbed hidden state to get log-probabilities of a categorical distribution $\tilde{\mathbf{o}}_t$ to sample the next token. Using this categorical distribution we then sample the next token. Simultaneously, we use the unperturbed hidden state $\mathbf{h}_t$ to directly compute the unperturbed next-token log-probabilities $\mathbf{o}_t$. We shall use the unperturbed log-probabilities to evaluate the KL term of our learning objective which ensures that the perturbed text has a content that is close to the original text. Formally, we compute the KL between perturbed and unperturbed logits as $D_{\text{KL}}(\tilde{\mathbf{o}}_t || \mathbf{o}_t)$. The loss function for our BART-based implementation for each input text $\mathbf{x}$ can be described as follows :

$$\mathcal{L} = \sum_{t=1}^{T} D_{\text{KL}}(\tilde{\mathbf{o}}_t || \mathbf{o}_t) + \texttt{CE}(a, f_{\text{attribute}}(\mathbf{y}_{1:T})).$$

The detailed algorithm is shown in Algorithm 1. In practice, we run this controlled generation process as a batch hence generating K samples in parallel on GPU. Because the sample of the generated text $\mathbf{y}_{1:T}$ is discrete and not differentiable with respect to sampling logits, we use REINFORCE to propagate the learning gradient to the model from the second term of the learning objective.

Model		Masked LM	CheckList	CASPer
Content Preservation	YELP	0.070	0.141	**0.169**
	IMDB	0.091	**0.193**	0.165
Perplexity	YELP	250.42	342.97	**136.97**
	IMDB	230.90	312.48	**205.88**
BLEU-4	YELP	0.903	0.530	**0.120**
	IMDB	0.9027	0.909	**0.102**

Table 1: Comparison between models on the YELP and IMDB dataset. The model used for steering is a pre-trained sentiment classification model.

3 Related Work

The task of generating controlled perturbations has been well studied in literature. (Hu et al., 2017) propose a model aims to generate plausible sentences conditioned on representation vectors with semantic structure. Another work (Ye et al., 2020) focuses on generating perturbations, however unlike the previous work (Hu et al., 2017), the conditioning need not be simply a class label. The conditioning can be a data structure such as a table. The model is trained end-to-end similarly to the objective of (Hu et al., 2017). PPLM (Dathathri et al., 2019) combine a pre-trained LM with an attribute classifier to perform controlled language generation. The idea is to use the attribute classifier to steer the text generation process without further training of any of the two models.

A recent work by Ribeiro et al., 2020 employs a tool Checklist which is one of the attempts to come up with generalized perturbations. For generation, Checklist uses a set of pre-defined templates, lexicons, general-purpose perturbations, and context-aware suggestions. Polyjuice (Wu et al., 2021) requires paired dataset in the form of text and their perturbed versions for different control codes. Therefore the mapping between text and perturbed version is learned through supervision. Another parallel work Tailor (Ross et al., 2021) generates perturbations designed for different control codes by making use of a combination of semantic roles and content keywords. And thereby require supervision for different controls. In contrast, CASPer does not require any task-specific or control-code specific training and can be used to work with different control code models given input text.
Another closely related work, GYC (Madaan et al., 2021) generates K text samples $Y = \{\mathbf{y}^k\}_{k=1}^K$ given a text $\mathtt{x}$ and controlling `condition` that specify the scope of the generated text. In GYC, the inference of latent representation of the input text with respect to a GPT-2 decoder is done directly via gradient descent. This approach fails to achieve good inference for long and complex text. As the inferred latent is inaccurate, the reconstructed text from the decoder often fails to preserve the content of the original text. CASPer amortizes this inference process by introducing an encoder that directly returns an accurate latent representation with respect to the GPT-2 decoder.

MiCE (Ross et al., 2020) proposes a technique to generate counterfactual explanations which are human interpretable and user-centric. They fine-tune T5 to generate counterfactual text and use them as explanations of the model but lack feature-attributions. Another work (Ross et al., 2021) tries to generate perturbation with semantic controls but rely on specific templates derived using semantic roles and labelling heuristics. CATGen (Wang et al., 2020) tries to generate attribute-specific text but it requires task-specific training of sequence to sequence model.

4 Experiments

The goal of the experiments is to: *1)* evaluate the ability of our model to generate text on long and complex text and *2)* evaluate how well the generated perturbations can act as data-augmentation samples in order to make a downstream classification task performance more robust.

Inputs	Generated Controlled Perturbations
Steering Goal : To maximize the probability of the positive class label with respect to a pre-trained sentiment classifier. **Input Text**: `We know from other movies that the writers can't solve problems, like this one. The actors are also good.` **Initial Label** : Negative	We know from other movies that the actors are good but they cannot help the movie. The premise was not too bad and is enjoyable. **New Label** : Positive We know from other documentaries that the acting was quite strong. We can't say the premise was good but the execution was good. **New Label** : Positive We know from other movies that the actors are good but they must win over the fans. The premise was not too bad. **New Label** : Positive We know from other movies that the actors are good. They cannot help the movie. The premise was not too bad. **New Label** : Positive
Steering Goal: To make the sentence contain at least one location named-entity. **Input Text**: `A wonderful little production.The filming technique is very unassuming- very old-time-BBC fashion.` **Initial State** : No location named-entity is present in the text	A wonderful first look at this film from the UK. The filming is unassertive. </s>io is also one of the best productions. **Location Named-Entity**: UK A wonderful little Theatre is an old-time BBC production, set in an old London apartment block. The film is set in London landmarks **Location Named-Entity**: London A wonderful little series about a BBC crew in Africa. Well-executed by the team who are experienced in film and rigging procedures. **Location Named-Entity**: Africa A wonderful little show about BBC's wartime bombing of Libya contributed by Peter Bradshaw in 1959. **Location Named-Entity**: Libya
Steering Goal : To maximize the probability of the positive class label with respect to a pre-trained sentiment classifier. **Input Text**: `I had the terrible misfortune of having to view this "b-movie" in it's entirety. All I have to say is-- save your time and money!!` **Initial Label** : Negative	I have the wonderful misfortune of having to view this New Girl in it's entirety. To view it positively, I must open up and say. **New Label** : Positive I had the terrible misfortune of having to view this ẗransform into... in it's entirety. It's truly amazing how some programs remain.. **New Label** : Positive I had the terrible misfortune of having to view this movie in it endearing ways. It's a wonderful salute wives lenders acknowledge.. **New Label** : Positive I had the terrible misfortune of having to view this in it's full glory. All I have to say is wow. everyone agrees. **New Label** : Positive
Steering Goal : To maximize the probability of the negative class label with respect to a pre-trained sentiment classifier. **Input Text**: `For those who'd like to see this movie? I'd say: go! Without the narration it might be a very good movie/documentary.` **Initial Label** : Positive	For those who'd like to see it I suggest that they see this movie without narration. The movie isn't a very good movie. **New Label** : Negative For those want to get to know the movie through its narration? I'd say: go! Without the narration it might be a very bad movie. **New Label** : Negative

Table 2: Generated controlled perturbations from the proposed model CASPer.

4.1 Datasets

To focus on the ability of our model to deal with long and complex text, we evaluate the models on the following data sets text.

1. **YELP Sentiment Dataset.** To evaluate how our model is able to change the sentiment of the original text and achieve the target sentiment, we use the YELP sentiment dataset. This dataset is also characterized by informal text which can be seen in realistic user inputs.
2. **IMDB Sentiment Dataset.** To further evaluate how our model is able to change the sentiment of the original input text, we test on IMDB Sentiment Dataset. This dataset is also characterized by long and complex text. Because of this, it also tests the model in how well the generated text preserve important information of the original text.

4.2 Controlled Text Generation with Attribute Steering

We first evaluate the quality of our generated text with respect to the steering signal. We expect our generated text to preserve the semantic content and syntactic structure of the input text while being fluent and diverse as we steer the text towards the target attribute. The steering signal we evaluate in this work is to make the sentiment of the target text from negative to positive. In the setting of long and complex text, obtaining text with these characteristics has been challenging.

4.2.1 Baselines

To compare with state-of-the-art template-based methods relying on token substitutions via dictionaries we compare with Checklist (Ribeiro et al., 2020). In comparison to this baseline, we expect ours to generate more fluent and diverse text samples that is free from the restrictions of the pre-specified templates. We also compare against Masked-LM (Devlin et al., 2018), which is a dictionary-free approach but still relies on masking a specific token in the input text and and letting the model fill-in the masked token. The randomness in this filling-in process leads to generation of counter-factual samples. Hence this model generates only token-level substitutions and does not generate fluent sentence-level text. We expect CASPer to address this limitation of purely token-level substitutions. Lastly we omit the comparison with GYC as we found that GYC failed to reconstruct 98% of the input text samples from our dataset and hence generated no output. Failure of this reconstruction is a critical problem in GYC. GYC relies on a curriculum in which the reconstruction happens first and then reconstructed text is perturbed for task specific generation.

4.2.2 Metrics

To assess the quality of generated counterfactual text we focus on evaluating content preservation, fluency, diversity and syntactic similarity. We use the following metrics to measure the above characteristics.

1. **Content Preservation.** By measuring content preservation, we assess the similarity between input text and the counterfactual text samples. For this, we use the transformer model proposed in (Reimers & Gurevych, 2019). While higher content preservation is desirable in general, this metric alone does not provide the complete evaluation. Therefore for proper evaluation, we will introduce a second metric that measures sample diversity.
2. **Diversity.** This metric evaluates how different are the generated samples from each other. We find the BLEU-4 score between the input text and the generated text. Hence, if this score is lower, then the generated counterfactual samples have a high diversity at the token level.
3. **Fluency.** Fluency of the generated samples is important to evaluate because the samples must come from a distribution that the test model is likely to see when it is deployed. This is computed by finding the perplexity score of the generated output. We take a GPT model for computing the perplexity. Lower perplexity implies that the generated text is more fluent.

4.2.3 Quantitative Results

In Tab. 1 we note that in terms of content preservation, our model is competitive with rule-based token substitution methods like Checklist. This shows that CASPer is able effectively preserve the content in its samples. In terms of perplexity score, CASPer outperforms the baselines achieving the lower perplexity score. This shows that the samples generated by CASPer are fluent and plausible. Lastly, in terms of BLEU-4 score, we note that our model again outperforms the baselines with our generated samples achieving the lowest values of BLEU-4 score with the original input text. This shows that our model indeed generates diverse samples with that have low token-level match with the input text. Overall, this shows that CASPer is able to effectively generate samples that preserve the original content, are fluent and diverse in comparison to the baselines.

4.2.4 Qualitative Analysis

In Tab. 2, we show samples of text generated by CASPer. We show two experiments. In the first experiment, our steering goal was to take an input text and perturb it so that the probability of its sentiment becomes large. The probability of the sentiment is estimated using a pre-trained sentiment classification model. The initial label of the text was negative. On the right we note that CASPer has successfully perturbed the text to change its sentiment label to positive. Furthermore, note that the generated sample have good content preservation as all the samples talk about movies and actors. Furthermore, the model makes some important changes to the content that result in a change in the sentiment of the text. We also note that each sample is different from the other thus producing diverse samples. Lastly, we note that the samples are fluent and plausible text samples.

In the second experiment, our steering goal was to take a sentence that does not contain a location named-entity and perturb it so that contains a location named-entity. We see that CASPer produces samples that contain a location named-entity tag. We also note that the named entity that the model introduces are diverse and are used in a variety of contexts in the generated text. As before, the text samples are fluent and preserve the content of the original input text.

Note that these samples clearly do not change the sentiment of the text and only introduce some location entities. Because we expect the actual location (i.e. UK or Libya) should not be a causal term in prediction of the sentiment of the text, these samples can act as effective samples for augmenting the training data when we train a downstream sentiment model. While a model that is biased may predict different labels based on the actual location token used, this kind of data augmentation will regularize the model to be more robust to such changes which should ideally not affect the predicted label of the test model.

4.2.5 Computational Cost

Furthermore, GYC is orders of magnitude slower requiring up to 200 or more iterations to generate text and often failing to reconstruct the original text ($\sim$98% text samples) while CASPer can generate good quality perturbed text and converging in around 100 iterations. However, in our experiments, we ran CASPer for 200 iterations to generate the samples for evaluation.

5 Conclusion

In this paper, we introduced CASPer, a plug-and-play counterfactual text generation framework. We showed that our model can gracefully handle long and complex text. Our generated controlled perturbations preserve the content of the original text while also being fluent, diverse and effective in terms of the provided steering signal. We showed that samples generated by CASPer can act as effective candidates for training data augmentation and improve the robustness of the target model and preventing the target model from modeling spurious correlations between the target label and non-causal aspects of the input text. As future work, we would like to investigate how well the model can perform in low-resource domains.

References

Dathathri, S., Madotto, A., Lan, J., Hung, J., Frank, E., Molino, P., Yosinski, J., and Liu, R. Plug and play language models: a simple approach to controlled text generation. *arXiv preprint arXiv:1912.02164*, 2019.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Hu, Z., Yang, Z., Liang, X., Salakhutdinov, R., and Xing, E. P. Toward controlled generation of text. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1587–1596. JMLR. org, 2017.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

Madaan, N., Padhi, I., Panwar, N., and Saha, D. Generate your counterfactuals: Towards controlled counterfactual generation for text. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 13516–13524, 2021.

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. Improving language understanding by generative pre-training (2018), 2018.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. 2019.

Reimers, N. and Gurevych, I. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. URL `http://arxiv.org/abs/1908.10084`.

Ribeiro, M. T., Wu, T., Guestrin, C., and Singh, S. Beyond accuracy: Behavioral testing of nlp models with checklist. *arXiv preprint arXiv:2005.04118*, 2020.

Ross, A., Marasović, A., and Peters, M. E. Explaining nlp models via minimal contrastive editing (mice). *arXiv preprint arXiv:2012.13985*, 2020.

Ross, A., Wu, T., Peng, H., Peters, M. E., and Gardner, M. Tailor: Generating and perturbing text with semantic controls. *arXiv preprint arXiv:2107.07150*, 2021.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 5998–6008. Curran Associates, Inc., 2017. URL `http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf`.

Wang, T., Wang, X., Qin, Y., Packer, B., Li, K., Chen, J., Beutel, A., and Chi, E. Cat-gen: Improving robustness in nlp models via controlled adversarial text generation. *arXiv preprint arXiv:2010.02338*, 2020.

Wu, T., Ribeiro, M. T., Heer, J., and Weld, D. S. Polyjuice: Automated, general-purpose counterfactual generation. *arXiv preprint arXiv:2101.00288*, 2021.

Ye, R., Shi, W., Zhou, H., Wei, Z., and Li, L. Variational template machine for data-to-text generation. *arXiv preprint arXiv:2002.01127*, 2020.