Improving Faithfulness by Augmenting Negative Summaries from Fake Documents

Abstract

Current abstractive summarization systems tend to hallucinate content that is unfaithful to the source document, posing a risk of misinformation. To mitigate hallucination, we must teach the model to distinguish hallucinated summaries from faithful ones. However, the commonly used maximum likelihood training does not disentangle factual errors from other model errors. To address this issue, we propose a back-translation-style approach to augment negative samples that mimic factual errors made by the model. Specifically, we train an elaboration model that generates hallucinated documents given the reference summaries, and then generate summaries from the fake documents. We incorporate the negative samples into training through a controlled generator, which produces faithful/unfaithful summaries conditioned on the control codes. Additionally, we find that adding textual entailment data through multitasking further boosts the performance. Experiments on XSum, Gigaword, and WikiHow show that our method consistently improves faithfulness without sacrificing informativeness according to both human evaluation and automatic metrics.1

1 Introduction

Despite the fast progress on fluency and coherence of text summarization systems, a common challenge is that the generated summaries are often unfaithful to the source document, containing hallucinated, non-factual content (Cao et al., 2018; Falke et al., 2019). Current summarization models are usually trained by maximum likelihood estimation (MLE), where unfaithful and faithful summaries are penalized equally if they both deviate from the reference. As a result, if the model fails to imitate the reference, it is likely to “over-generalize” and produce hallucinated content.

In this work, we address the issue by explicitly teaching the model to discriminate between positive (groundtruth) and negative (unfaithful) summaries. The key challenge is to generate realistic negative samples. Existing work on negative data augmentation mostly focuses on corrupting the reference (e.g., replacing entities) or sampling low-probability model outputs (Cao and Wang, 2021; Kryscinski et al., 2020a; Kang and Hashimoto, 2020). However, the synthetic data often does not resemble actual hallucinations from the model (Goyal and Durrett, 2021) and many methods rely on external tools such as NER taggers.

To generate unfaithful summaries, we propose a simple method inspired by back-translation (Sennrich et al., 2016) (Fig. 1). Specifically, we first generate fake documents using an elaboration model that is trained to produce a document given the summary. We then generate summaries from the fake documents, which are assumed to be unfaithful since they are likely to contain hallucinated information in the fake documents. Given the reference summaries and the augmented negative samples, we train a controlled generation model that generates either faithful or unfaithful summaries conditioned on a faithfulness control code. At inference time, we control the model to generate only faithful summaries. We call our approach CoFE (Controlled Faithfulness via Elaboration). The controlled generation framework also makes it easy to incorporate additional data: we show that jointly training on natural language inference (NLI) datasets to generate entailed (faithful) and non-entailed (unfaithful) hypothesis further improves the result.

We evaluate CoFE on three summarization datasets. Both automatic metrics and human evaluation show that our method consistently outperforms
prior methods in terms of faithfulness and content
similarity to the reference, without sacrificing ab-
stractiveness (Ladhak et al., 2021).

2 Approach
To learn a summarization model, the commonly
used MLE aims to imitate the reference and does
not distinguish different types of errors, thus the
model may be misaligned with the desired behavior
in downstream applications. For example, a faith-
ful summary missing a detail would be preferred
over a summary with hallucinated details, even if
both have low likelihood under the data distribu-
tion. Therefore, additional inductive bias is needed
to specify what unfaithful summaries are. There-
fore, we augment negative examples and jointly
model the distributions of both faithful and un-
faithful summaries. At decoding time, we generate the
most likely faithful summary.

Negative data augmentation. The key chal-
gen in generating negative summaries is to simul-
ate actual model errors. Prior approaches largely
focus on named entities errors. However, differ-
ent domains exhibit diverse hallucination errors
(Goyal and Durrett, 2021); in addition, certain do-
mins may not contain entities that can be easily
detected by off-the-shelf taggers (e.g., stories or in-
structions). Our key insight is that the reverse sum-
marization process—expanding a summary into a
document—requires the model to hallucinate de-
tails, thus provides a domain-general way to pro-
duce unfaithful information. Instead of manipulat-
ing the reference summary directly, we expand it
into a fake document, and generate negative sum-
maries from it using the summarization model.

More formally, given a set of document-
summary pairs \((x, y)\), we train a backward elab-
oration model \(p_{\text{back}}(x \mid y)\) as well as a forward
summarization model \(p_{\text{for}}(y \mid x)\). Then, given
a reference summary \(y\), we first generate a fake
document \(\tilde{x}\) from \(p_{\text{back}}\), then generate the negative
sample \(y_{\text{neg}}\) from \(\tilde{x}\) using \(p_{\text{for}}\), forming a pair of
positive and negative samples \((x, y)\) and \((x, y_{\text{neg}})\).
To avoid data leakage (i.e. training models and gen-
nerating summaries on the same data), we split the
training data into \(K\) folds; the negative examples
in each fold are generated by elaboration and sum-
marization models trained on the rest \(K - 1\) folds.
We use \(K = 5\) in the experiments.

Controlled generation. Given the positive and
negative samples, we would like the model to learn
to discriminate faithful summaries from unfaithful
ones. Inspired by controlled generation methods
(Keskar et al., 2019), we train the model to generate
faithful or unfaithful summaries conditioned on a
control code. In practice, we prepend a prefix at
the beginning of the document ([ENT] for posi-
tive examples and [CON] for negative examples).
At inference time, we always prepend [ENT] to
generate faithful summaries.

Training. Our training data consists of positive
examples (i.e. the original dataset) and gener-
ated negative samples, marked with different pre-
fixes. Let \(L_{\text{pos}}, L_{\text{neg}}\) denote negative log-likelihood
(NLL) losses on the positive and negative examples.
We use a multitasking loss that is a weighted sum
of the two losses to balance the contribution from
different types of examples: \(\mathcal{L} = L_{\text{pos}} + \lambda L_{\text{neg}}\).

Adding NLI datasets. We hypothesize that in-
corporating NLI data through multitasking would
transfer knowledge about entailment to the gener-
ator, allowing it to better model faithful and un-
faithful summaries. Specifically, an entailed hy-
pothesis could be considered as the faithful sum-
mary, and non-entailed hypothesis as the unfaith-
ful summary. Thus, the NLI sentence pairs can be
naturally incorporated into our training frame-
work. Let $\mathcal{L}_{\text{NLI}}$ denote the NLL loss on the auxiliary NLI examples. The loss function becomes:

$$\mathcal{L} = \mathcal{L}_{\text{pos}} + \lambda_1 \mathcal{L}_{\text{neg}} + \lambda_2 \mathcal{L}_{\text{NLI}}.$$ 

3 Experiments

Datasets. We evaluate our approach on 3 datasets, including: (i) XSum (Narayan-Chen et al., 2019), a dataset of BBC news articles paired with one-sentence summaries; (ii) Gigaword (Rush et al., 2015), a headline generation dataset compiled from the Gigaword corpus (Graff et al., 2003); and (iii) Wikihow (Koupaee and Wang, 2018), a dataset of how-to articles compiled from WikiHow, each paired with paragraph headlines as the summary. For the auxiliary NLI data, we use SNLI (Bowman et al., 2015) and MultiNLI (Williams et al., 2018), both containing pairs of premise and hypothesis sentences.

Baselines. We compare with three baselines: (i) MLE, the standard training algorithm; (ii) Loss Truncation (LT) (Kang and Hashimoto, 2020) that adaptively removes high-loss examples which are assumed to be noisy/unfaithful; and (iii) CLIFF (Cao and Wang, 2021), a contrastive learning method based on generated negative samples.

Implementation. All generation models (including the baselines) are fine-tuned from BART-large (Lewis et al., 2019). We decode from all models using beam search with a beam size of 6. For CoFE, we train the model using Fairseq (Ott et al., 2019) with a learning rate of 3e-5. We generate one negative sample for each document in the original dataset. To ensure that negative examples are different from the references, we remove the top 10% summaries ranked by their edit distances to the reference. To train the controlled generator, we set coefficients ($\lambda_1$, $\lambda_2$) of the loss terms such that the reweighted number of examples in the original dataset, the negative samples, and optionally the NLI datasets have the ratio $1:0.5:0.5$. Details for other baselines are in Appendix B.

Metrics. A good summary must cover important content, be faithful to the document, and be succinct. We evaluate the generated summaries from the following aspects:

- Content selection. We use similarity to the reference as a proxy measure, and report ROUGE (Lin, 2004) and BertScore (Zhang et al., 2020).

- Faithfulness. For automatic evaluation, we use QuestEval (Scialom et al., 2021), a QA-based metric; and FactCC (Kryscinski et al., 2020b), a learned faithfulness predictor. For human evaluation, we randomly selected 100 examples from each dataset. Given a document with the generated summaries from all systems (including the references), we ask annotators from Amazon Mechanical Turk to evaluate whether each summary is supported by the document. Each output is evaluated by 3 annotators. If two or more annotators vote “supported”, then we consider the output faithful. More details are described in Appendix B.

- Extractiveness. Ladhak et al. (2021) show that it is important to measure the extractiveness of the summaries to determine whether a method improves faithfulness mainly by copying from the document. Therefore, we also report coverage and density that measure the percentage of the words and the average length of text spans copied from the document (Grusky et al., 2020).

Results. Table 1 shows our main results. CoFE outperforms the baselines in human evaluated faithfulness accuracy on 2 out of the 3 datasets. On Gigaword, LT performs the best but it also incurs the largest drop in ROUGE and BertScore and increase in copying. CLIFF is good at fixing entity errors, but it has less advantage on datasets like WikiHow that contain fewer entities detectable by off-the-shelf taggers. On average, CoFE is less extractive than CLIFF and LT, indicating that our faithfulness improvements are not simply due to more copying (Ladhak et al., 2021). Finally, we find that adding NLI brings a marginal improvement on top of our negative samples.

Are generated negative summaries really unfaithful? Our method relies on the assumption that elaboration of summaries introduces hallucinations, which results in unfaithful summaries. To verify this, we assess whether our generated negative samples are true negatives. We randomly sample 1000 documents for each dataset and compare the negative samples generated by our method vs. CLIFF. We report the QuestEval scores as well as human-annotated faithfulness scores on a sub-
Table 1: Main results. The best result per metric for each datasets is **bolded**. For “Extractiveness”, lower is better. RL and BS denotes ROUGE-L and BertScore-P. For human evaluation, we report the percentage of faithful summaries based on majority vote (Human Acc) and the total number of votes for faithfulness (# Votes). CoFE outperforms the baselines on average without decreasing overlap with the reference or increasing copying.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Method</th>
<th>Ref. Similarity (↑)</th>
<th>Faithfulness (↑)</th>
<th>Extractiveness (↑)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RL</td>
<td>BS</td>
<td>Human Acc / # Votes</td>
<td>QuestEval</td>
</tr>
<tr>
<td><strong>XSUM</strong></td>
<td>MLE</td>
<td>37.21</td>
<td>45.36</td>
<td>64% / 192</td>
</tr>
<tr>
<td></td>
<td>LT</td>
<td>35.77</td>
<td><strong>47.39</strong></td>
<td>61% / 188</td>
</tr>
<tr>
<td></td>
<td>CLIFF</td>
<td>36.41</td>
<td>52.78</td>
<td>68% / 192</td>
</tr>
<tr>
<td></td>
<td>CoFE</td>
<td>36.38</td>
<td>52.09</td>
<td>68% / 194</td>
</tr>
<tr>
<td></td>
<td>CoFE + NLI</td>
<td>36.98</td>
<td>52.90</td>
<td><strong>70% / 196</strong></td>
</tr>
<tr>
<td><strong>Gigaword</strong></td>
<td>MLE</td>
<td>33.95</td>
<td>27.77</td>
<td>70% / 206</td>
</tr>
<tr>
<td></td>
<td>LT</td>
<td>34.22</td>
<td>26.35</td>
<td>76% / 204</td>
</tr>
<tr>
<td></td>
<td>CLIFF</td>
<td><strong>35.59</strong></td>
<td><strong>30.78</strong></td>
<td>73% / 201</td>
</tr>
<tr>
<td></td>
<td>CoFE</td>
<td>35.53</td>
<td>30.70</td>
<td><strong>73% / 210</strong></td>
</tr>
<tr>
<td></td>
<td>CoFE + NLI</td>
<td>34.02</td>
<td>27.77</td>
<td>74% / 211</td>
</tr>
<tr>
<td><strong>WikiHow</strong></td>
<td>MLE</td>
<td>37.93</td>
<td>43.55</td>
<td>87% / 233</td>
</tr>
<tr>
<td></td>
<td>LT</td>
<td>38.01</td>
<td>43.61</td>
<td>83% / 228</td>
</tr>
<tr>
<td></td>
<td>CLIFF</td>
<td>37.29</td>
<td>42.73</td>
<td>83% / 233</td>
</tr>
<tr>
<td></td>
<td>CoFE</td>
<td>37.86</td>
<td><strong>43.67</strong></td>
<td>84% / 232</td>
</tr>
<tr>
<td></td>
<td>CoFE + NLI</td>
<td><strong>38.23</strong></td>
<td>43.08</td>
<td><strong>88% / 238</strong></td>
</tr>
</tbody>
</table>

Table 2: Quality of generated negative samples. Lower number is better (more likely to be true negatives).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Method</th>
<th>QuestEval (↑)</th>
<th>Human Acc (↑)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>XSUM</strong></td>
<td>CoFE</td>
<td>24.34</td>
<td>19%</td>
</tr>
<tr>
<td></td>
<td>CLIFF</td>
<td>27.65</td>
<td>60%</td>
</tr>
<tr>
<td><strong>Gigaword</strong></td>
<td>CoFE</td>
<td>33.69</td>
<td>34%</td>
</tr>
<tr>
<td></td>
<td>CLIFF</td>
<td>39.42</td>
<td>40%</td>
</tr>
<tr>
<td><strong>WikiHow</strong></td>
<td>CoFE</td>
<td>24.72</td>
<td>32%</td>
</tr>
<tr>
<td></td>
<td>CLIFF</td>
<td>28.31</td>
<td>39%</td>
</tr>
</tbody>
</table>

**Is faithfulness controllable?** We use the controlled generator to model distributions of both faithful and unfaithful summaries. To verify the effect of the control code, we measure the change in ROUGE scores on XSum after toggling the control code from faithful ([ENT]) to unfaithful ([CON]). As expected, we observe that R1/R2 drops from 45.26/22.19 to 37.29/15.82, indicating that the model has learned to discriminate faithful and unfaithful summaries.

4 **Related Work**

Recent work in abstractive summarization has shown that state-of-the-art models sometimes generate non-factual information that is not consistent with the article (Falke et al., 2019; Cao et al., 2018). This has spurred efforts in building automated metrics for factuality (Kryscinski et al., 2020a; Durmus et al., 2020; Wang et al., 2020; Goyal and Durrett, 2020) and more faithful systems (Xu et al., 2020; Filippova, 2020).

Prior work has proposed to filter the training dataset to remove noisy examples to improve faithfulness. For example, Kang and Hashimoto (2020) drop high-loss examples from training observing that these examples are usually of lower quality. Nan et al. (2021) discard sentences from gold summaries if there is an entity that does not match the entities in the document. Goyal and Durrett (2021) take a more fine-grained approach, and use a dependency arc-based entailment metric (Goyal and Durrett, 2020) to filter noisy tokens from the summary.

On modeling, prior work has incorporated additional information such as relation triplets (Cao et al., 2018), knowledge graph of relations (Zhu et al., 2021) and topical information (Aralikatte et al., 2018), knowledge graph of relations (Zhu et al., 2020; Wang et al., 2020; Goyal and Durrett, 2020) and more faithful systems (Xu et al., 2020; Filippova, 2020).
References


Ziqiang Cao, Furu Wei, Wenjie Li, and Sujian Li. 2018. Faithful to the original: Fact aware neural abstractive summarization. Proceedings of the AAAI Conference on Artificial Intelligence, 32(1).


Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling.


A Human-Evaluation Setup

We use Amazon Mechanical Turk as human-evaluations platform. The prompt is shown in Fig. 2. We only hire annotators in U.S. and with more than 98% hit receive rate.

B Experiment Detail

Model details. For both the summarization model, the elaboration model, and the controlled generator, we fine-tune a pre-trained BART model (Lewis et al., 2019) using Fairseq (Ott et al., 2019) and the default learning rate $3 \times 10^{-5}$. All summaries are generated using beam search with a beam size of 6. Linear-scale the max update steps of learning-rate scheduler according to the number of samples in the training data.

For hyperparameters, we follow the setting of fine-tuning BART on XSUM (Lewis et al., 2019), which uses 8 cards, UPDATE_FREQ is 4, TOTAL_NUM_UPDATES is 20000. Linear scale the max-update-step by the number of negative data and NLI data. For the weights of different tasks, an intuitive idea is to fix "the ratio of the product of the number of samples and their weights for different tasks". We set Product_summarization : Product_negative : Product_NLI = 1 : 0.5 : 0.5. For example, if we have 1000 positive and 1000 negative samples in training set, the weight of positive data is 1, the weight of negative data is 0.5. If we filter half negative samples out, reduce it into 500 samples, then the weight of two tasks is 1.

Other baselines: For MLE, the repository of BART releases hyperparameters and checkpoint for XSUM. Based on the hyperparameters for xsum, we scale the max-update-step linearly according to the size of training set of gigaword and wikihow.

For Loss-truncation, besides the hyperparameters in MLE, there are some hyperparameters for the loss function. We follow the settings in their paper. For CLIFF, we only use "SysLowCon" as the negative data augmentation method, which is the best single method they claimed in the paper. They release the checkpoints of XSUM and hyperparameters in their github repository. We only re-scale the max-update-step.

Computational Resources and Model Size.

CoFE on one dataset requires training 11 models, including 10 models for generating negative samples, since each fold needs an elaborator and a summarizer. On a 4 RTX8000 GPU node, each model needs 2 hours to fine-tune. It takes 22 hours to get the final generated output. BART-large has 400M parameters.