Think While You Write: Hypothesis Verification Promotes Faithful Knowledge-to-Text Generation

Yifu Qiu1*, Varun Embar2, Shay B. Cohen1, Benjamin Han2
University of Edinburgh1, Apple2
{yifu.qiu, scohen}@ed.ac.uk, {v_embar, ben.b.han}@apple.com

Abstract
Knowledge-to-text generators often struggle to faithfully generate descriptions for the input facts: they may produce hallucinations that contradict the input, or describe facts not present in the input. To reduce hallucinations, we propose a decoding-only method, TWEAK (Think While Effectively Articulating Knowledge), which can be integrated with any generator without retraining. TWEAK treats the generated sequences at each decoding step and its future sequences as hypotheses, and ranks each generation candidate based on the extent to which their hypotheses are supported by the input facts using a Hypothesis Verification Model (HVM). We first demonstrate the effectiveness of TWEAK by using a Natural Language Inference (NLI) model as the HVM and report improved faithfulness with a minimal impact on the quality. We then replace the NLI model with a task-specific HVM trained with a first-of-a-kind dataset, FATE (Fact-Aligned Textual Entailment), which pairs input facts with their original and perturbed descriptions. We test TWEAK with two generators, and the best TWEAK variants improve on average for the two models by 2.24/7.17 points in faithfulness (FactKB) in in/out-of-distribution evaluations, respectively, and with only a 0.14/0.32-point decline in quality (BERTScore)1.

1 Introduction
Knowledge-to-text generation (K2T) aims to generate precise and fluent textual descriptions which are consistent with the input facts (Gardent et al., 2017; Perez-Beltrachini and Lapata, 2018; Agarwal et al., 2021; Colas et al., 2021). Although the neural generators are capable of generating fluent and high-quality texts on various tasks (Ribeiro et al., 2021a; Zhou et al., 2021; Liu et al., 2022; Chen et al., 2022; Qiu and Cohen, 2022), one major challenge remains to be hallucination (Zhao et al., 2020; Maynez et al., 2020; Dziri et al., 2022; Daheim et al., 2023; Xu et al., 2023), i.e., the tendency of the models to produce outputs that contradict or are not supported by the inputs.

In this paper, we address the hallucination problem with a model-agnostic decoding method, TWEAK (Think While Effectively Articulating Knowledge). Different from previous works such as (Hashem et al., 2023), we tweak only the decoding process without requiring re-training of the generative models, thus making our approach easily integratable with any K2T generator. The existing decoding methods of a generative model, such as beam search, sample candidates only from the predicted likelihood without any consideration on the faithfulness implication of these candidates. The problem of exposure bias of autoregressive generation only makes the matter worse once any deviation from a faithful generation occurs, since these errors accumulate and become unrecoverable (Schmidt, 2019; Zhang et al., 2023). TWEAK mitigates this problem by verifying the faithfulness of the candidates at each decoding step to reduce hallucinations. As the example illustrated in Fig. 1, for each candidate at a decoding step, TWEAK treats the sequence generated so far as the backward and the forward hypothesis (inspired by Lu et al.), respectively, and feeds them into a Hypothesis Verification Model (HVM) to estimate the candidate’s faithfulness score, a measure indicating how well the candidate supports the input facts. The candidates are then ranked considering both their generation scores and faithfulness scores.

We first deploy a natural language inference (NLI) model (Nie et al., 2020) as the HVM for experimentation, and observe that this approach, TWEAK-NLI, indeed improves the faithfulness of the output compared to the baseline (beam search).
by a significant margin. The distribution shift between NLI and faithfulness assessment tasks, however, may result in reduced output quality (Kryscinski et al., 2020; Laban et al., 2022; Qiu et al., 2023). We therefore experiment with a second variation, TWEAK-HVM, where we propose a task-specific HVM trained with a first-of-a-kind dataset, FATE (Fact-Aligned Textual Entailment). This dataset pairs and aligns input facts with their original and perturbed descriptions. We mimic the autoregressive decoding process where we expand the generation process one token at a time until completion to synthesize the triple-hypothesis pairs with their faithfulness labels. The HVM is then trained to predict all triple-hypothesis labels in a tabular form (Wang et al., 2021; Fatahi Bayat et al., 2022). Experimental results on WebNLG (Gardent et al., 2017) and two out-of-distribution datasets, Tek-Gen (Agarwal et al., 2021) and GenWiki (Perez-Beltrachini and Lapata, 2018), confirm the advantages of TWEAK-HVM. It also greatly reduces computation as it encodes both input facts and hypotheses simultaneously.

We summarize our contributions as follows,

- We propose a model-agnostic decoding strategy, TWEAK, which incorporates an HVM for candidate ranking, and show that the approach improves faithfulness of K2T generation when using an NLI model as the HVM.
- We propose a new dataset, FATE, which pairs and aligns input facts with their original and perturbed descriptions at word level.
- We train a task-specific HVM with FATE and demonstrate its advantages over the NLI-based method in output faithfulness and quality.

2 Related Work

Knowledge-to-text generation tasks involve the transformation of structured data or knowledge into natural language texts (Gardent et al., 2017; Perez-Beltrachini and Lapata, 2018; Colas et al., 2021). Previous works encode the structured input explicitly as models’ representations (Schmitt et al., 2021; Marcheggiani and Perez-Beltrachini, 2018; Guo et al., 2019; Rebuffel et al., 2020; Koncel-Kedziorski et al., 2019). A usual way is to serialize the structured input first and use a pre-trained model to directly generate its description (Ribeiro et al., 2021b; Li et al., 2021; Su et al., 2021). However, a notable challenge is hallucinations – models produce claims that are not supported by inputs (Hashem et al., 2023; Wang et al., 2023; Yang et al., 2022). Previous work has explored methods including plan-before-generate pipelines (Puduppully and Lapata, 2021; Puduppully et al., 2019, 2022), architecting models to be explicitly fact-aware (Wang et al., 2022; Ji et al., 2023), and augmenting the training data with self-supervised learning (Han and Shareghi, 2022; Wang et al., 2023; Hashem et al., 2023). Mitigating hallucinations in decoding, however, has received relatively less attention, despite its advantages in model-agnostic applications (Xiao and Wang, 2021; Lu et al., 2022; Wan et al., 2023).

Comparing to a recent work (Wan et al., 2023), where effect of different decoding strategies on faithfulness of abstractive summarization is investigated, and a faithfulness re-ranking method is proposed to improve output, our work is unique in that we target a different task (K2T), use hypothesis verification instead of a faithfulness composite metric.
to guide the ranking, and train a task-specific HVM based on our novel dataset to bring improvement to both faithfulness and quality.

3 Knowledge-to-Text: Task Definition

K2T task concerns generating a natural language description \( y \) for a list of input facts \( x = \langle \ldots, x_t, \ldots \rangle \), where \( x_t \) is a fact triple represented as \( \langle \text{subj}, \text{rel}, \text{obj} \rangle \) indicating a relation, \text{rel}, holds between the subject entity, \text{subj}, and the object entity, \text{obj}. Two complementary requirements exist for an ideal generation: a high-quality generation should describe all of the input facts in a grammatical and readable fashion, while a faithful generation should not add any additional claim or contradict any input fact.

We use an autoregressive language model \( p_0 \) with parameters \( \theta \) to estimate the probability of the token sequence \( y = \langle \ldots, y_t, \ldots \rangle \), i.e., \( p_0(y \mid x) = \prod_{t=1}^{|y|} p_0(y_t \mid y_{<t}, x) \). To decide on the final output, a decoding process finds the optimal sequence by solving \( y^* = \arg\max_{y \in Y} F(y) \), where \( Y \) is the set of all possible sequences, and \( F \) is an objective function. This can be accomplished by selecting the top \( k \) candidates generated from vocabulary \( \mathcal{V} \) using an \( F \)-approximating scoring function \( f \) one token \( y_t \) at a time:

\[
Y' = \{ y_{t} \circ y_t \mid y_{t-1} \in Y_{t-1}, y_t \in \mathcal{V} \}, \\
Y_t = \operatorname{argtopk} \{ f(y_{t}, y_t, x) \}.
\] (1)

Common decoding strategies, such as greedy and beam search, set \( f \) to \( \log p_0(y_{t} \mid x) \). In greedy search \( k \) is set to 1. In Sec. 4.1 we describe our scoring function that promotes faithful generation via hypothesis verification.

4 TWEAK

We now describe our approach in Sec. 4.1, the FATE dataset in Sec. 4.2, and our task-specific HVM trained with the dataset in Sec. 4.3.

4.1 Decoding with Hypothesis Verification

TWEAK is a model-agnostic decoding method that incorporates faithfulness objective into the decoding process. As shown in Fig. 1, at each decoding step we rank a candidate not only by its predicted likelihood from the generator, i.e., \( \log p_0(y_{t} \mid x) \), but also by its faithfulness score. To assess the faithfulness for a single candidate, we ask the model to look ahead and generate the future sequence until the end (Lu et al., 2022), and we approximate the candidate’s faithfulness based on the sequence generated to the current step, the backword hypothesis, and the future sequence, the forward hypothesis, using a HVM.

More specifically, we instantiate the scoring function \( f(y_{t}, y_t, x) \) in Eq. (1) as follows:

\[
f(y_{t}, y_t, x) = \log p_0(y_{t} \mid x) + \alpha \cdot f_{\text{faith}}(\cdot),
\]

\[
f_{\text{faith}}(\cdot) = w_1 \cdot h(x, y_{t}) + (1 - w_2)h(x, y_{t}).
\] (2)

The overall score \( f \) is thus a weighted sum of the generator’s predicted likelihood and faithfulness \( f_{\text{faith}} \). The latter, weighted by \( \alpha \), scores how likely a backward and forward hypothesis, \( y_{\leq t} \) and \( y_{\leq t} \), respectively, supports the input facts via the hypothesis scoring function \( h \), and returns a weighted sum of the faithfulness scores of the two hypotheses. Depending on the implementation of \( h \), we have different instantiations for \( y_{\leq t} \) and weight \( w_1 \), as described in Sec. 4.1.1 and Sec. 4.1.2.

4.1.1 Hypothesis Verification via NLI

One simple way to implement an HVM is to treat the concatenated input facts as a premise and the (possibly partial) generated sequence as the hypothesis, then use an NLI model’s prediction as the faithfulness score. We thus instantiate Eq. (2) as:

\[
h(x, y_{t}) = \text{NLI}(x_1 \circ \ldots \circ x_m, y_{t}),
\]

\[
y_{t} = y_{\leq t} \circ g(y_{\leq t}, x),
\]

\[
g(y_{\leq t}, x) = \arg\max_{y_{\leq t}} \prod_{t'=t+1} \prod_{y_{t'} \in Y_{t'}} p_0(y_{t'} | y_{<t'}, x),
\] (3)

\[
w_{t} = \begin{cases} 1 & \text{for TWEAK-NLI-B} \\ 0 & \text{for TWEAK-NLI-F} \\ \frac{t}{m} & \text{for TWEAK-NLI-B+F} \end{cases}
\]

The hypothesis scoring function in the above is simply an NLI model returning a score indicating how likely the hypothesis is supported by the premise.\(^3\) The forward hypothesis \( y_{t} \) is a complete sequence concatenating the sequence generated so far and a possible future sequence. Function \( g \) is a greedy generator producing a future sequence from time step \((t + 1)\) on. We experiment with three NLI-based variants: TWEAK-NLI-B uses only the

\(^3\)We omit function arguments as ‘\( \cdot \)’ if context is clear.

\(^4\)The weight \( \alpha \) can be determined on a validation set such that a desirable balance between output quality and faithfulness is achieved. See Fig. 3 for example.

\(^5\)We only use the entailment score and discard the scores of neutral and contradiction.
backward hypothesis with \( w_t \) set to 1, TWEAK-NLI-F uses only the forward hypothesis with \( w_t \) set to 0, and TWEAK-NLI-B+F uses both, with \( w_t \) dynamically set to the ratio of the lengths of the backward and the forward hypotheses at time step \( t \). We call this last weighing scheme dynamic aggregation (DA), and the intuition is to place less weight on the relatively incomplete backward hypothesis at the early stage of decoding.

4.1.2 Hypothesis Verification via HVM
Alternatively, we train a task-specific HVM to score hypotheses, and instantiate Equ. (2) as:

\[
\begin{align*}
    h(x, y) &= \text{HVM}(x, y), \\
    y_t &= g(y_{\leq t}; x), \\
    w_t &= \frac{t}{t + |y_t|}.
\end{align*}
\]

Comparing to the NLI-based hypothesis scoring function in Equ. (3), here we use HVM to compute a score indicating how well sequence \( y \) supports input facts \( x \). We also consider only the future sequence as \( y_t \), and the weight \( w_t \) is computed entirely dynamically, similar to TWEAK-NLI-B+F. More details of HVM are discussed in Sec. 4.3.

4.2 Fact-Aligned Textual Entailment Dataset
To train the task-specific HVM (see Sec. 4.3), we construct a novel dataset called FATE, where each instance is a tuple \((F^+, F^-, T^+, T^-)\): \(F^+, F^-\) are fact triples and their perturbed version, and \(T^+, T^-\) are their respective descriptions. We take \(F^+\) and \(T^+\) from WebNLG (Gardent et al., 2017), and employ a large language model (LLM)\(^5\) to perturb one triple in \(F^-\) to construct \(F^-\). The perturbation may happen in any position in a fact triple, i.e., subject, object, or relation. We then ask the LLM to generate description \(T^-\) for \(F^-\) that is as close to \(T^+\) as possible. The perturbed span is then identified and clearly marked with tag “\(<S I>“” in both \(T^+\) and \(T^-\), where \(I\) indicates the perturbed triple corresponding to the span. We present an instance in Table 1 and the dataset statistics in Appendix B.

4.3 A Task-specific HVM
There are two disadvantages when using an NLI model as the HVM in TWEAK: 1) the NLI model concatenates all triples into a single premise, losing the entailment relationship between each individual triple and a hypothesis, and 2) NLI models often perform poorly in faithfulness classification due to their inability to generalize to a different target task (Utama et al., 2022; Kryscinski et al., 2020).

To address these problems, we train a task-specific HVM using our dataset FATE described in Sec. 4.2. As depicted in Fig. 2, we first assemble fact triples and the corresponding pair of backward and forward hypotheses as input. We then encode the input via a language model (RoBERTa; Liu et al. 2019) and use average pooling over all tokens to obtain the representations of each triple and hypothesis. A biaffine attention layer is then used to predict a 2D table representing the pair-wise faithfulness relations (unsupported/supported) between each triple-hypothesis pair. Our model is then trained to minimize a table-form objective (Wang et al., 2021; Fatahi Bayat et al., 2022),

\[
L = -\frac{1}{2|X|} \sum_{x \in X} \sum_{y \in \{y_{\leq t}, y_t\}} \log P(\hat{B}_{x,y} = B_{x,y} | x, y),
\]

where \( x \) is the set of fact triples in an instance, \( y_{\leq t} \) and \( y_t \) are a corresponding backward and forward hypotheses, and \( B_{x,y} \) and \( \hat{B}_{x,y} \) are the ground-truth label and the biaffine model prediction for the triple-hypothesis pair, respectively. For inference, we instantiate the function HVM in Equ. (4) as:

\[
\text{HVM}(x, y) = \frac{1}{|X|} \sum_{x \in X} \log P(\hat{B}_{x,y} = 1 | x, y).
\]

\(^5\)We use text-davinci-003. The prompt templates we use for manipulating triple and description are in Appendix E.

\(^6\)1 is the supported label.
To train the task-specific HVM with our FATE dataset, for each training instance we randomly set a decoding position and break its original and perturbed descriptions in two parts to simulate possible backward and forward hypotheses: a hypothesis derived from a perturbed description that overlaps with the marked perturbed span receives unsupported label as the ground truth, and all of the others receive supported. We present an example synthetic pair of backward and forward hypotheses in Table 1. Finally, we up-sample the supported hypotheses to balance the labels.

### 5 Experiments and Results

**Datasets and Models.** We train two base generation models BART-large (Lewis et al., 2020) and T5-large (Raffel et al., 2020), following the hyperparameter settings from (Ribeiro et al., 2021a), and evaluate our decoding strategy on WebNLG (Gardent et al., 2017), TekGen (Agarwal et al., 2021), and GenWiki (Jin et al., 2020).

**Metrics.** We assess the models on faithfulness and quality. Faithfulness metrics measure how much semantic distortion the output contains with respect to the input, while quality metrics measure how close a model output is to the reference. For the former we employ FactKB (Feng et al., 2023), a state-of-the-art reference-free metric constructed via factuality pre-training. For the latter we employ the three metrics previously used by Ribeiro et al. (2021a): BLEU (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005), and BERTScore (Zhang et al., 2020).

**Baseline Decoding Strategies.** As baselines we test two basic decoding strategies: greedy search and beam search (Sec. 3). For our TWEAK decoding strategy, we first test it with an off-the-shelf NLI model (Nie et al., 2020) for hypothesis verification. Three variations are tested: TWEAK-NLI-B, TWEAK-NLI-F, and TWEAK-NLI-B+F, using only backward, only forward, and both hypotheses, respectively. We then replace the NLI model with our task-specific HVM trained with FATE dataset (Sec. 4.2 & 4.3) as TWEAK-HVM variant. More implementation details are in Appendix A.1.

### 5.1 Main Results in WebNLG

Our main results are shown in Table 2. Overall the best TWEAK variants improve on average +2.24 points on faithfulness (FactKB), with only -0.14 points degradation in quality (BERTScore).

**Baseline Decoding Results.** Looking at the results of the two baseline decoding strategies, we observe that beam search consistently outperforms greedy search on both faithfulness and quality metrics. This suggests that increasing the beam size during decoding widens the exploration and generates a more faithful and higher quality output.

**TWEAK Decoding with NLI.** Comparing our TWEAK-NLI variants to the baselines, we find that all of them outperform beam search on faithfulness (FactKB), with TWEAK-NLI-B on BART-large

---

**Table 1:** FATE’s example (upper panel) and the synthesized hypotheses derived from it (bottom panel).

<table>
<thead>
<tr>
<th>FATE Instance</th>
<th>Type</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Ireland, largest_city, Dublin)</td>
<td>PTs</td>
<td>-</td>
</tr>
<tr>
<td>(Ireland, national_capital, Dublin)</td>
<td>NTs</td>
<td>-</td>
</tr>
<tr>
<td>Dublin is Ireland’s &lt;S0&gt; largest city &lt;/S0&gt;</td>
<td>PD</td>
<td>-</td>
</tr>
<tr>
<td>Dublin is Ireland’s &lt;S0&gt; national capital &lt;/S0&gt;</td>
<td>ND</td>
<td>-</td>
</tr>
</tbody>
</table>

**Synthesized Hypotheses (at 10th Decoding Step)**

<table>
<thead>
<tr>
<th>Hypothesis</th>
<th>Supported/BH</th>
<th>Forward/FH</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dublin is Ireland’s largest city.</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Dublin is Ireland’s national capital.</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

**Table 2:** Results of decoding baselines and our TWEAK decoding variants measured by faithfulness metric (FACTKB) and quality metrics (BLEU, METEOR, BS = BERTScore) on WebNLG dataset. Numbers in bold are the highest scores among the baselines (greedy and beam) or among the TWEAK variants.

<table>
<thead>
<tr>
<th>Decoding</th>
<th>FKB</th>
<th>BLEU</th>
<th>MET</th>
<th>BS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greedy</td>
<td>27.74</td>
<td>51.3</td>
<td>66.79</td>
<td>94.2</td>
</tr>
<tr>
<td>Beam</td>
<td>28.91</td>
<td>54.23</td>
<td>67.55</td>
<td>94.35</td>
</tr>
<tr>
<td>TWEAK-NLI-F</td>
<td>30.46</td>
<td>52.02</td>
<td>67.17</td>
<td>94.2</td>
</tr>
<tr>
<td>TWEAK-NLI-B</td>
<td>30.59</td>
<td>49.68</td>
<td>65.88</td>
<td>94.12</td>
</tr>
<tr>
<td>TWEAK-NLI-B+F</td>
<td>30.47</td>
<td>51.62</td>
<td>66.84</td>
<td>94.19</td>
</tr>
<tr>
<td>TWEAK-HVM</td>
<td>31.34</td>
<td>53.14</td>
<td>67.38</td>
<td>94.25</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Decoding</th>
<th>FKB</th>
<th>BLEU</th>
<th>MET</th>
<th>BS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greedy</td>
<td>30.14</td>
<td>57.71</td>
<td>68.71</td>
<td>94.84</td>
</tr>
<tr>
<td>Beam</td>
<td>31.29</td>
<td>58.93</td>
<td>69.38</td>
<td>94.86</td>
</tr>
<tr>
<td>TWEAK-NLI-F</td>
<td>33.03</td>
<td>53.51</td>
<td>67.8</td>
<td>94.39</td>
</tr>
<tr>
<td>TWEAK-NLI-B</td>
<td>31.49</td>
<td>44.96</td>
<td>65.02</td>
<td>93.93</td>
</tr>
<tr>
<td>TWEAK-NLI-B+F</td>
<td>32.71</td>
<td>51.71</td>
<td>66.73</td>
<td>94.19</td>
</tr>
<tr>
<td>TWEAK-HVM</td>
<td>33.34</td>
<td>57.31</td>
<td>69.02</td>
<td>94.68</td>
</tr>
</tbody>
</table>
Table 3: Generalization results on out-of-distribution (OOD) test sets TekGen and GenWiki. BS = BERTScore. Numbers in bold are the highest scores among the baselines (greedy and beam) or among the TWEAK variants.

<table>
<thead>
<tr>
<th>Model</th>
<th>Decoding</th>
<th>TekGen</th>
<th>GenWiki</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FactKB</td>
<td>BLEU</td>
<td>METEOR</td>
</tr>
<tr>
<td>BART-large</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Greedy</td>
<td>9.44</td>
<td>22.42</td>
<td>44.21</td>
</tr>
<tr>
<td>Beam</td>
<td>11.57</td>
<td>23.34</td>
<td>43.86</td>
</tr>
<tr>
<td>TWEAK-NLI-F</td>
<td>14.77</td>
<td>15.92</td>
<td>38.48</td>
</tr>
<tr>
<td>TWEAK-NLI-B</td>
<td>12.8</td>
<td>20.22</td>
<td>42.62</td>
</tr>
<tr>
<td>TWEAK-NLI-B+F</td>
<td>15.2</td>
<td>17.57</td>
<td>38.79</td>
</tr>
<tr>
<td>TWEAK-HVM</td>
<td>13.24</td>
<td>19.26</td>
<td>40.48</td>
</tr>
<tr>
<td>T5-large</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Greedy</td>
<td>9.12</td>
<td>21.09</td>
<td>43.09</td>
</tr>
<tr>
<td>Beam</td>
<td>11.64</td>
<td>21.35</td>
<td>42.97</td>
</tr>
<tr>
<td>TWEAK-NLI-F</td>
<td>16.51</td>
<td>8.57</td>
<td>37.48</td>
</tr>
<tr>
<td>TWEAK-NLI-B</td>
<td>12.12</td>
<td>19.98</td>
<td>41.32</td>
</tr>
<tr>
<td>TWEAK-NLI-B+F</td>
<td>15.86</td>
<td>10.66</td>
<td>38.44</td>
</tr>
<tr>
<td>TWEAK-HVM</td>
<td>13.44</td>
<td>21.51</td>
<td>41.61</td>
</tr>
</tbody>
</table>

Table 4: Human evaluation on NLI vs. Beam and HVM vs. Beam. Numbers are win-rates over the non-similar output. Highest numbers in each aspect are bolded.

<table>
<thead>
<tr>
<th></th>
<th>Faithfulness</th>
<th>Completeness</th>
<th>Readability</th>
</tr>
</thead>
<tbody>
<tr>
<td>NLI vs Beam</td>
<td>56.06%</td>
<td>56.67%</td>
<td>36.07%</td>
</tr>
<tr>
<td>HVM vs Beam</td>
<td>59.09%</td>
<td>56.06%</td>
<td>45.83%</td>
</tr>
</tbody>
</table>

improving +1.68 points, and TWEAK-NLI-F on T5-large improving +1.74 points over beam search. This demonstrates the effectiveness of performing hypothesis verification during decoding to improve output faithfulness. For each generator, a different variant achieves the best faithfulness result while the combo approach, TWEAK-NLI-B+F, is always in the middle. This indicates that simply combining the scores obtained from both hypotheses does not guarantee an optimal gain in faithfulness.

On the quality front, all TWEAK-NLI variants score lower on all metrics, with TWEAK-NLI-F showing the least regression. A manual analysis reveals that the more faithful generations exhibit a higher divergence from the reference (see Appendix A.2 for an example). This is also consistent with Wan et al. (2023) who show that optimizing faithfulness can lead to lower textual similarity with reference. We also note that since quality metrics require reference while the faithful metric does not, any noise present in the reference may lead to a lower score even if the output is reasonable.

**TWEAK Decoding with HVM.** Comparing the TWEAK-HVM variant (Sec. 4.3) to the baselines, TWEAK-HVM significantly outperforms in faithfulness: its FactKB score reaches 31.34 (+2.43 points improvement) and 33.34 (+2.05 points) on BART-large and T5-large over beam search, respectively. TWEAK-HVM is also more faithful than the most faithful TWEAK-NLI variant, demonstrating the advantage of a task-specific HVM and the benefits of performing triple-specific entailment classification.

On output quality, TWEAK-HVM still fares lower than beam search, but it scores higher than all TWEAK-NLI variants on all metrics, therefore significantly closing the gaps to be almost on par with beam search, with only 0.1/0.18 decline in BERTScore for BART/T5, respectively. In summary, TWEAK-HVM is more faithful than the baselines with almost as good quality.

### 5.2 Out-of-distribution Evaluation

We have demonstrated that performing hypothesis verification during decoding can significantly enhance faithfulness without losing much of the overall quality on an in-distribution (ID) test set. To evaluate the out-of-distribution (OOD) effectiveness of our approach, we conducted experiments on two additional datasets that the HVM is not trained on: TekGen (Agarwal et al., 2021) and GenWiki (Jin et al., 2020). We show the results for BART and T5-large in Table 3. Overall the best TWEAK variants improve BART and T5 on average +7.17 points on faithfulness (FactKB), with only -0.32 points degradation on quality (BERTScore).

TWEAK-HVM still outperforms the best baseline (beam search) on faithfulness, yielding an average relative improvement of 14.95%/14.98%
Table 5: Effect of dynamic aggregation (DA) with TWEAK-NLI-B+F on WebNLG and BART-large. MET and BS stand for METEOR and BERTScore.

<table>
<thead>
<tr>
<th></th>
<th>FactKB</th>
<th>BLEU</th>
<th>MET</th>
<th>BS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BART</td>
<td>30.47</td>
<td>51.62</td>
<td>66.84</td>
<td>94.19</td>
</tr>
<tr>
<td>w/o DA</td>
<td>-0.12</td>
<td>-2.63</td>
<td>-1.34</td>
<td>-0.12</td>
</tr>
<tr>
<td>T5</td>
<td>32.71</td>
<td>51.71</td>
<td>66.73</td>
<td>94.19</td>
</tr>
<tr>
<td>w/o DA</td>
<td>-0.79</td>
<td>-12.21</td>
<td>-2.21</td>
<td>-0.49</td>
</tr>
</tbody>
</table>

On TekGen/GenWiki, respectively. However, the best NLI variant outperforms TWEAK-HVM on faithfulness by an average relative margin of 18.82%/46.35% on TekGen/GenWiki, respectively. Since the NLI model is trained with OOD datasets, it appears to be more generalizable than our task-specific HVM in the OOD setup, as expected.

On the quality front, all TWEAK variants score lower than the best baseline, similar to the ID setting (Sec. 5.1). if we compare NLI vs HVM by picking first the most faithful TWEAK-NLI variant, it always performs worse on quality than the TWEAK-HVM variant. For example, on TekGen with BART-large, comparing TWEAK-NLI-B+F, which has the highest FactKB score among all NLI variants, to TWEAK-HVM using BLEU, the HVM variant outperforms by 1.69 absolute points. It appears TWEAK-HVM is able to strike a better balance between faithfulness and quality.

5.3 Human Evaluation

We also conduct human evaluation on WebNLG for the output of the beam search, TWEAK-NLI, and TWEAK-HVM decoding. The graders are asked to compare side-by-side NLI vs. Beam and HVM vs. Beam on three aspects: faithfulness (whether an output contains only claims supported by the input), completeness (whether an output captures all of the input), and readability (whether an output is grammatical and easy to understand), and can choose between four grades: better than, similar to, worse than, and can’t decide. We use T5-large as the base model, and sample the output uniformly across different numbers of input facts (1 to 7), resulting in 127 instances. The result is shown in Table 4 in terms of the win-rates over the output that are not marked as similar. Overall, consistent with the main results discussed in Sec. 5.1, both TWEAK variants outperform the beam search baseline on faithfulness and completeness, but underperform on readability. In particular, TWEAK-HVM outperforms more than TWEAK-NLI on faithfulness and readability, with nearly identical completeness.

6 Analysis

We report additional experiments and analyses in this section.

Dynamic Aggregation. As observed in Table 2, different models achieve peak faithfulness using either backward or forward hypotheses (BART favors backward while T5 favors forward). This implies both types of hypotheses can be useful in improving faithfulness of the output, which is borne out again by the OOD results reported in Table 3 where we observe that TWEAK-NLI-B+F, using both backward and forward hypotheses via dynamic aggregation (DA; see Sec. 4.1.1), becomes the most faithful variant. To assess DA’s impact, we examine TWEAK-NLI-B+F without DA on WebNLG in Table 5, revealing a clear performance drop in both faithfulness and quality. This underscores the importance of adapting weights placed on forward/backward hypotheses throughout the decoding process, as incomplete hypothesis verification can be less reliable.

Weighting Effects. As described in Equ. (2), we combine the generative score and the faithfulness score weighted by \( \alpha \) to rank the candidates. We are therefore interested in the effect of choosing \( \alpha \). In Fig. 3 we plot the resulting quality score (BLEU) and faithfulness score (FactKB) with different \( \alpha \), with 0 being equivalent to beam search. The experiments are done with WebNLG test set and BART-large, using TWEAK-NLI-B+F and TWEAK-HVM variants.

We observe that increasing the weight on faithfulness score improves faithfulness in almost all settings at the cost of reduced quality. HVM out-
performs NLI on quality at all $\alpha$ values, and HVM also outperforms NLI on faithfulness when $\alpha \geq 6$. This clearly demonstrates the advantages of HVM in the ID setting (see Sec. 5.2).

**Number of Input Facts.** The number of input fact triples is an important factor in determining K2T output quality: the more triples in the input, the more challenging for a model to generate a faithful and high-quality output. To investigate the correlation, we split the WebNLG test set into three groups: Short (one input triple), Medium (2-4 triples), and Long (5-7 triples). We then test both TWEAK-NLI-B+F and TWEAK-HVM variants with BART-large on these three groups. The results are shown in Table 6.

On generative quality (BLEU) we observe that TWEAK-HVM outperforms TWEAK-NLI-B+F by a similar amount across the three groups. On faithfulness (FactKB), however, TWEAK-HVM’s improvement over TWEAK-NLI-B+F is positively correlated with the number of input triples, climbing from +0.28%, +2.39%, to +5.47% from Short, Medium, to Long. We attribute this growing advantage to HVM’s ability to model each triple-hypothesis relation, whereas TWEAK-NLI-B+F concatenates all triples into a single premise and may misclassify with more triples in the input.

**Exploring Larger Beam Size.** If our TWEAK decoding strategy can promote a lower-ranked candidate based on its faithfulness score, can we further improve its effectiveness by increasing the beam size, i.e., letting in more candidates to be evaluated by TWEAK? To answer this question, we run beam search, TWEAK-NLI-B+F, and TWEAK-HVM side-by-side on WebNLG test set and BART-large, and plot their quality (BLEU) and faithfulness (FactKB) differences in Fig. 4.

Comparing TWEAK-HVM with beam search (blue bars), we observe that TWEAK-HVM improves on faithfulness, with improvement growing with beam size. In terms of quality, however, TWEAK-HVM underperforms beam search, but the drop stabilizes after beam size = 4.

Comparing TWEAK-HVM with TWEAK-NLI-B+F (red bars), we observe that on quality, TWEAK-HVM steadily becomes better than TWEAK-NLI-B+F as beam size increases. On faithfulness, TWEAK-HVM starts out being slightly worse at beam size = 2, but then steadily becomes better over TWEAK-NLI-B+F with in-

<table>
<thead>
<tr>
<th>#Triples</th>
<th>Short</th>
<th>Medium</th>
<th>Long</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Sample</td>
<td>908</td>
<td>2196</td>
<td>620</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BLEU</th>
<th>HVM</th>
<th>NLI</th>
<th>Δ</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>64.18</td>
<td>50.22</td>
<td>1.09%</td>
</tr>
<tr>
<td></td>
<td>46.26</td>
<td>45.23</td>
<td>2.36%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FactKB</th>
<th>HVM</th>
<th>NLI</th>
<th>Δ</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>18.11</td>
<td>33.47</td>
<td>0.28%</td>
</tr>
<tr>
<td></td>
<td>43.17</td>
<td>40.81</td>
<td>5.47%</td>
</tr>
</tbody>
</table>

Table 6: TWEAK decoding performance on WebNLG with increasing number of input triples. We split the WebNLG test set into three groups: Short (1 triples), Medium (2-4 triples) and Long (5-7 triples).

Figure 4: Performance differences ($\Delta$) on quality (BLEU) and faithfulness (FactKB) between TWEAK-HVM, TWEAK-NLI-B+F and beam search on various beam sizes {2, 4, 6, 8, 10, 15}. All experiments are done on WebNLG with BART-large.

Figure 5: The distributions of the relative positions where negative predictions (i.e., possible hallucination) happen during the decoding process. 0 and 1 along the horizontal axis represent the start and end of the decoding. The upper and bottom panel represent TWEAK-HVM and TWEAK-NLI-B+F running on WebNLG with BART-large, respectively.
The leader of Russia is Valentina Matviyenko. The country is the location of the club FC Torpedo Moscow which is managed by Valery Petrakov. The club is affiliated with...
Ethical Considerations

This paper focuses on accurate knowledge-to-text generation, crucial for reducing errors in natural language generation. Our goal is to minimize mistakes and misinformation synthesized in texts produced by the generative language models. Within this scope, TWEAK provides a method to manipulate text generation to be more faithful without altering the trained generative model.

In our human evaluation, all participants are employed as full-time workers within our internal data annotation team. To guarantee evaluation quality, we ensure that all participants possess a native-speaker level of proficiency in English. Each participant receives fair compensation, commensurate with standard wages in the United States. All participants are explicitly informed that the annotated data would be used for research purposes. Additionally, this study has undergone review and approval by our internal ethical panel.

Limitations

The authors wish to note the following limitations:

• The proposed TWEAK decoding strategy imposes additional cost at inference time compared to the baseline approaches such as beam search.
• The reported results indicate while all TWEAK variants outperform the baseline in ID settings, in OOD settings the results are more nuanced. On faithfulness, TWEAK-HVM still outperforms the baselines in OOD settings, but it underperforms the more costly variant TWEAK-NLI in some settings (see Sec. 5.2). A future exploration is to further improve the robustness of HVM as discussed in Sec. 6.
• Our proposed approach has only been tested in English language. The authors expect the approach to work reasonably well in non-English languages, provided adequate datasets and base models are available.

Acknowledgements

We thank the reviewers for their useful feedback. We are grateful for the 2023 Apple Scholarship awarded to Yifu Qiu.

References


Pennsylvania, USA. Association for Computational Linguistics.


Zhuoer Wang, Marcus Collins, Nikhita Vedula, Simone Flicie, Shervin Malmasi, and Oleg Rokhlenko. 2023. Faithful low-resource data-to-text generation through


A Appendix

A.1 Implementation Details.

We implement all of our methods with the transformers package (Wolf et al., 2020). We mainly follow Ribeiro et al. to train and test our models in all experiments. In this section we describe all hyperparameters used for reproducibility.

We train BART-large and T5-large as the base generators. They have 406M and 770M parameters, respectively.

A.1.1 WebNLG

BART. Following (Ribeiro et al., 2021a), we add the special tokens <H>, <R>, and <T> to the models’ vocabulary, insert them before the subject, relation, and object, respectively, before concatenating them all into a triple string. We then concatenate all triple strings within an instance to form the input. We train a BART-large model (Lewis et al., 2020) as our generator with 2 epochs and a batch size 4. We set the learning rate to be $3 \cdot 10^{-5}$. We use beam search as the baseline and set the beam search size to 5. The best checkpoint is selected based on the validation BLEU score (Papineni et al., 2002). We set the max generation length to 384.

T5. We perform the same preprocessing as above for T5’s input. We additionally append a prefix, “translate from Graph to Text.” at the beginning of an input. We train a T5-large generator with 10 epochs and batch size 4. We use the same learning rate as suggested in (Ribeiro et al., 2021a) at $3 \cdot 10^{-5}$. We also use a linearly decreasing learning rate schedule without warm-up. Again, the beam search is used as the baseline but the beam size is set to 3. The best checkpoint is again selected based on the validation BLEU score (Papineni et al., 2002). We set the max generation length also to 384.

TWEAK Decoding For both models, when applying TWEAK decoding, we set the beam size to 4, and generate forward hypotheses using greedy decoding for efficiency. The weighting parameter $\alpha$ is set to 8. We also set the max generation length to 384.

A.1.2 TekGen

We use the same hyperparameters as we do for WebNLG to train and test for both BART-large
and T5-large generators. When applying TWEAK decoding, we set $\alpha$ to 8 and 1 for BART and T5, respectively. We use the same beam size as the beam search baseline, where beam size is 5 and 3 for BART-large and T5-large, respectively. Max generation length is also set to 384.

### A.1.3 GenWiki

We use the same hyperparameters as we do for WebNLG to train and test for both BART-large and T5-large generators, except we set a larger batch size at 32. We also raise the batch size for training T5-large to 16, and keep the other hyperparameters the same as when we train T5-large on WebNLG.

When applying TWEAK decoding, we set $\alpha$ to 8 and 2 for BART-large and T5-large, respectively. We use the same beam size as the beam search baseline, where the beam size is 5 and 3 for BART-large and T5-large, respectively. Max generation length is also set to 384.

### A.2 Example: Faithful Output is Worded Differently

The following shows the output from BART-large using the baseline beam search decoding and the TWEAK-NLI-B+F variant decoding strategy. Although the output of the latter produces a higher faithfulness score (FactKB), it is worded more differently with respect to the reference, resulting in a lower quality score.

- **Facts:**
  - (Aston Martin V8, related Mean Of Transportation, Aston Martin DBS)
  - (Aston Martin V8, engine, 5.3 litres)
  - (Aston Martin V8, assembly, United Kingdom)

- **Reference:** The Aston Martin V8 is assembled in the United Kingdom and has an engine volume of 5.3 litres. The Aston Martin V8 and Aston Martin DBS are a related means of transport.

- **Beam search:** Aston Martin V8, which is 5.3 litres and made in the United Kingdom, is related to the Aston Martin DBS.

- **TWEAK-NLI-B+F:** The United Kingdom is the location of the assembly of the Aston Martin V8 which has a 5.3 litres engine and is related to the DBS.

---

### B FATE Dataset Statistics

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Subj</th>
<th>Rel</th>
<th>Obj</th>
<th>Triples</th>
<th>Entity Avg.</th>
<th>Triples</th>
<th>Words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>423</td>
<td>235</td>
<td>1499</td>
<td>922</td>
<td>4.54</td>
<td>19.8</td>
<td></td>
</tr>
<tr>
<td>Perturbed</td>
<td>432</td>
<td>1666</td>
<td>3118</td>
<td>7368</td>
<td>17.05</td>
<td>20.0</td>
<td></td>
</tr>
</tbody>
</table>

Table 7: Dataset statistics for our curated FATE. Both the original and the perturbed sets contain 18,102 instances. All numbers are counts of unique instances.

### C Statistics of Evaluation Benchmarks

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Subj</th>
<th>Rel</th>
<th>Obj</th>
<th>Triples</th>
<th>Entity Avg.</th>
<th>Triples</th>
<th>Words</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebNLG</td>
<td>Train</td>
<td>430</td>
<td>246</td>
<td>1613</td>
<td>2090</td>
<td>4.8</td>
<td>19.8</td>
</tr>
<tr>
<td></td>
<td>Test</td>
<td>575</td>
<td>300</td>
<td>1882</td>
<td>2331</td>
<td>4.0</td>
<td>19.5</td>
</tr>
<tr>
<td>TekGen</td>
<td>Train</td>
<td>20K</td>
<td>1K</td>
<td>13K</td>
<td>34K</td>
<td>1.7</td>
<td>21.0</td>
</tr>
<tr>
<td></td>
<td>Test</td>
<td>1000</td>
<td>200</td>
<td>1176</td>
<td>1783</td>
<td>1.7</td>
<td>21.4</td>
</tr>
<tr>
<td>GenWiki</td>
<td>Train</td>
<td>713K</td>
<td>287</td>
<td>273K</td>
<td>1754K</td>
<td>2.4</td>
<td>29.2</td>
</tr>
<tr>
<td></td>
<td>Test</td>
<td>817</td>
<td>157</td>
<td>2150</td>
<td>1783</td>
<td>3.9</td>
<td>18.6</td>
</tr>
</tbody>
</table>

Table 8: Dataset statistics for WebNLG, TekGen, and GenWiki. All numbers are counts of unique instances.

### D Weighting Effects in Out-of-distribution Evaluations

We have discussed the effect of manipulating weighting coefficient $\alpha$ in in-distribution experiments in Sec. 6. We further plot the weighting effect on out-of-distribution (OOD) datasets in Figure 7. On the two other datasets, HVM underperforms NLI on faithfulness due to distribution shift, but maintains higher quality scores than NLI at all $\alpha$ values. This shows HVM maintains the quality edge over NLI even in the OOD settings.

### E Prompt Template to Generate FATE

In this section we show the prompt templates we use with the LLM to create our FATE dataset (Sec. 4.2): one prompt is used to perturb a fact triple (Figure 8), and the other is to generate the description for the perturbed fact triple (Figure 9). Different from the majority of the similar methods deployed in the literature focusing solely on subjects or objects, we allow perturbations to happen at all possible positions – subject, relation, and object – in order to obtain more diverse datapoints.

### F More Examples

We show three more examples in Table 9:

---

### FATE Dataset Statistics

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Subj</th>
<th>Rel</th>
<th>Obj</th>
<th>Triples</th>
<th>Entity Avg.</th>
<th>Triples</th>
<th>Words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>423</td>
<td>235</td>
<td>1499</td>
<td>922</td>
<td>4.54</td>
<td>19.8</td>
<td></td>
</tr>
<tr>
<td>Perturbed</td>
<td>432</td>
<td>1666</td>
<td>3118</td>
<td>7368</td>
<td>17.05</td>
<td>20.0</td>
<td></td>
</tr>
</tbody>
</table>

Table 7: Dataset statistics for our curated FATE. Both the original and the perturbed sets contain 18,102 instances. All numbers are counts of unique instances.

### C Statistics of Evaluation Benchmarks

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Subj</th>
<th>Rel</th>
<th>Obj</th>
<th>Triples</th>
<th>Entity Avg.</th>
<th>Triples</th>
<th>Words</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebNLG</td>
<td>Train</td>
<td>430</td>
<td>246</td>
<td>1613</td>
<td>2090</td>
<td>4.8</td>
<td>19.8</td>
</tr>
<tr>
<td></td>
<td>Test</td>
<td>575</td>
<td>300</td>
<td>1882</td>
<td>2331</td>
<td>4.0</td>
<td>19.5</td>
</tr>
<tr>
<td>TekGen</td>
<td>Train</td>
<td>20K</td>
<td>1K</td>
<td>13K</td>
<td>34K</td>
<td>1.7</td>
<td>21.0</td>
</tr>
<tr>
<td></td>
<td>Test</td>
<td>1000</td>
<td>200</td>
<td>1176</td>
<td>1783</td>
<td>1.7</td>
<td>21.4</td>
</tr>
<tr>
<td>GenWiki</td>
<td>Train</td>
<td>713K</td>
<td>287</td>
<td>273K</td>
<td>1754K</td>
<td>2.4</td>
<td>29.2</td>
</tr>
<tr>
<td></td>
<td>Test</td>
<td>817</td>
<td>157</td>
<td>2150</td>
<td>1783</td>
<td>3.9</td>
<td>18.6</td>
</tr>
</tbody>
</table>

Table 8: Dataset statistics for WebNLG, TekGen, and GenWiki. All numbers are counts of unique instances.
Figure 7: The effect on quality (BLEU) and faithfulness (FactKB) from choosing different $\alpha$ in Equ. (2), with $\alpha = 0$ being equivalent to beam search. The results are obtained using TWEAK-NLI-B+F and TWEAK-HVM variants on WebNLG, TekGen and GenWiki with BART.

- **First example:** All three decoding strategies produce faithful textual descriptions for the given triples. TWEAK-HVM’s output is arguably more readable than the others, but in terms of textual similarity with the reference, both TWEAK variants produce less similar output than beam search.

- **Second example:** Beam search generates an output asserting "Costa Crociere is the parent company of Carnival Corporation", which directly contradicts the input fact 2 ("The parent company of Costa Crociere is Carnival Corporation"). Both TWEAK variants avoid this mistake.

- **Third example:** Beam search misses the fact "nearest city" which is captured by our TWEAK variants.
Fact triples:

1. <H> Accademia di Architettura di Mendrisio <R> country <T> Switzerland
2. <H> Accademia di Architettura di Mendrisio <R> dean <T> Mario Botta
3. <H> Accademia di Architettura di Mendrisio <R> city <T> Mendrisio
4. <H> Accademia di Architettura di Mendrisio <R> established <T> 1996
5. <H> Accademia di Architettura di Mendrisio <R> academic Staff Size <T> 100
6. <H> Accademia di Architettura di Mendrisio <R> location <T> Ticino

Reference: Accademia di Architettura di Mendrisio in Mendrisio, Switzerland has 100 employees, was established in 1996 and is overseen by Dean Mario Botta. The school’s location is Ticino.

Beam Search: The Accademia di Architettura di Mendrisio is located in Ticino, Mendrisio, Switzerland. It was established in 1996 and its dean is Mario Botta. It has 100 academic staff.

TWEAK-NLI: Mario Botta is Dean of the Accademia di Architettura di Mendrisio in Mendrisio, Ticino, Switzerland. The Accademia was established in 1996 and has 100 academic staff.

TWEAK-HVM: The dean of the Accademia di Architettura di Mendrisio is Mario Botta. The Accademia was established in 1996 in the city of Mendrisio, Ticino, Switzerland and has 100 academic staff.

Fact triples:

1. <H> Costa Crociere <R> location <T> Genoa
2. <H> Costa Crociere <R> parent Company <T> Carnival Corporation & plc
3. <H> AIDAstella <R> operator <T> AIDA Cruises
4. <H> AIDAstella <R> builder <T> Meyer Werft
5. <H> AIDAstella <R> owner <T> Costa Crociere

Reference: Carnival Corporation & plc is the parent company of Costa Crociere in Genoa, who own the AIDAstella. AIDAstella was built by Meyer Werft and is operated by AIDA Cruises.

Beam Search: AIDAstella, built by Meyer Werft and owned by Costa Crociere, is operated by AIDA Cruises. Costa Crociere is the parent company of Carnival Corporation & plc and is located in Genoa.

TWEAK-NLI: AIDAstella, built by Meyer Werft, is operated by AIDA Cruises. It is owned by Costa Crociere, whose parent company is Carnival Corporation & plc and is located in Genoa.

TWEAK-HVM: The parent company of Costa Crociere, located in Genoa, is Carnival Corporation & plc. AIDAstella, operated by AIDA Cruises, was built by Meyer Werft and is owned by Costa Crociere.

Fact triples:

1. <H> 14th New Jersey Volunteer Infantry Monument <R> district <T> Monocacy National Battlefield
2. <H> Monocacy National Battlefield <R> nearest City <T> Frederick Maryland

Reference: The 14th New Jersey Volunteer Infantry Monument is near Frederick, Maryland in the district of the Monocacy National Battlefield.

Beam Search: The 14th New Jersey Volunteer Infantry Monument is located in the district of the Monocacy National Battlefield, Frederick, Maryland.

TWEAK-NLI: Frederick, Maryland is the nearest city to Monocacy National Battlefield where the 14th New Jersey Volunteer Infantry Monument is located.

TWEAK-HVM: The 14th New Jersey Volunteer Infantry Monument is in the district of the Monocacy National Battlefield and the nearest city is Frederick, Maryland.

Table 9: Three examples taken from T5-large’s outputs on WebNLG.
Prompt template for perturbing original facts.

Using your commonsense knowledge to edit the predicate in the old triple to make it counter-factual. Note that you should not always use predicate negation.

Old triple: ('Aarhus Airport', 'operating Organisation', 'Aarhus Lufthavn A/S')
New triple: ('Aarhus Airport', 'leader Name', 'Aarhus Lufthavn A/S')

Old triple: ('Aarhus Airport', 'location', 'Tirstrup')
New triple: ('Aarhus Airport', 'country', 'Tirstrup')

Old triple: ('Aarhus Airport', 'location', 'Tirstrup')
New triple: ('Aarhus Airport', 'birthday', 'Tirstrup')

Old triple: ('jamaica at the fifa world cup', 'subclass of', 'jamaica national football team')
New triple: ('jamaica at the fifa world cup', 'president of', 'jamaica national football team')

Old triple: ('kentucky louisville rivalry', 'participating team', 'louisville cardinals')
New triple: ('kentucky louisville rivalry', 'beat', 'louisville cardinals')

Old triple: {$old_triple}
New triple: {$new_triple}

Figure 8: Prompt template we used for perturbing a fact triple. We allow the model to perturb both subject/object and predicate.

Prompt template for editing description to align with the new fact.

Minimally edit the following sentence so it supports the new fact triple instead of the old fact triple, while highlighting your edited text spans with '[' and ']'.

Sentence: Aarhus Airport serves the city of Aarhus, Denmark.
Old fact: ('Taylor County Texas', 'city Served', 'Aarhus Denmark')
New fact: ('Taylor County Texas', 'city Served', 'Aarhus Denmark')
Revised: [Taylor County Texas] services the city of Aarhus, Denmark

Sentence: Aarhus Airport is operated by Aarhus Lufthavn A/S.
Old fact: ('Aarhus Airport', 'operating Organisation', 'Aarhus Lufthavn A/S')
New fact: ('Aarhus Airport', 'death Date', 'Aarhus Lufthavn A/S')
Revised: Aarhus Airways [death date is] Aarhus Lufthavn A/S

Sentence: The location of Aarhus Airport is Tirstrup.
Old fact: ('Aarhus Airport', 'location', 'Tirstrup')
New fact: ('Aarhus Airport', 'leader Name', 'Tirstrup')
Revised: The [leader name of] Aarhus Airport is Tirstrup.

Sentence: {$sentence}
Old fact: {$old_triple}
New fact: {$new_triple}
Revised: 

Figure 9: Prompt template we used for editing textual description to align with the edited fact, while annotating the edited span.