Relation-Constrained Decoding for Text Generation

Xiang Chen, Zhixian Yang, Xiaojun Wan
Wangxuan Institute of Computer Technology, Peking University
Center for Data Science, Peking University
The MOE Key Laboratory of Computational Linguistics, Peking University
caspar@pku.edu.cn, yangzhixian@stu.pku.edu.cn, wanxiaojun@pku.edu.cn

Abstract

The dominant paradigm for neural text generation nowadays is seq2seq learning with large-scale pretrained language models. However, it is usually difficult to manually constrain the generation process of these models. Prior studies have introduced Lexically Constrained Decoding (LCD) to ensure the presence of pre-specified words or phrases in the output. However, simply applying lexical constraints has no guarantee of the grammatical or semantic relations between words. Thus, more elaborate constraints are needed. To this end, we first propose a new constrained decoding scenario named Relation-Constrained Decoding (RCD), which requires the model’s output to contain several given word pairs with respect to the given relations between them. For this scenario, we present a novel plug-and-play decoding algorithm named RElation-guided probability Surgery and bEAmlocation (RESEAL), which can handle different categories of relations, e.g., syntactical relations or factual relations. Moreover, RESEAL can adaptively “reseal” the relations to form a high-quality sentence, which can be applied to the inference stage of any autoregressive text generation model. To evaluate our method, we first construct an RCD benchmark based on dependency relations from treebanks with annotated dependencies. Experimental results demonstrate that our approach can achieve better preservation of the input dependency relations compared to previous methods. To further illustrate the effectiveness of RESEAL, we apply our method to three downstream tasks: sentence summarization, fact-based text editing, and data-to-text generation. We observe an improvement in generation quality. The source code is available at https://github.com/CasparSwift/RESEAL.

1 Introduction

Incorporating complex manual constraints into neural text generation is a challenging research topic. One of the most important manual constraints is the relation constraint, i.e., to guarantee that two pre-specified words must appear in the generated text and keep the given relation between them. Such relation constraints have various applications. For instance, data-to-text generation [11, 20] and fact-based text editing [17] aim to ensure the presence of given facts (entities and relations between them) in the output. Moreover, in sentence summarization task [36], there are some key semantic relations that must be preserved to ensure the fluency and factual constituency of the summaries.

The most prominent paradigm for text generation is seq2seq learning by finetuning the large-scale pretrained models [21, 34] and obtaining the outputs by beam search in an autoregressive manner. However, this paradigm often fails to satisfy the complex constraints because there is no explicit mechanism to enforce these constraints. To tackle this problem, previous works [15, 16, 31] propose Lexically Constrained Decoding (LCD) to preserve some given keywords in the output. However,

*Equal contribution.
simply utilizing these lexical constraints still struggles to ensure the word relation constraints. Therefore, another form of constrained decoding is needed to handle the relation constraints.

In this paper, we propose a new constrained decoding scenario named Relation-Constrained Decoding (RCD). Specifically, we adopt the triplets (head, relation, tail) as the relation constraints. At the decoding stage, we aim to force the model output to include these relation constraints. The right part of Figure 1 shows three instances for RCD with different relation types. In an evident way, satisfying relation constraints requires satisfying corresponding lexical constraints. A straightforward solution to this problem is to generate a set of candidate sentences using any LCD algorithm to ensure the keyword preservation, and then rerank them by the number of relation constraints they have met. However, this approach requires to first generate a number of whole sentences, which is inefficient and inflexible. To this end, we propose RESEAL (RELation-guided probability Surgery and bEam ALlocation), a relation-guided decoding algorithm for RCD that can dynamically adjust the choice of words during decoding. RESEAL modifies a conventional LCD method, i.e., Dynamic Beam Allocation (DBA) algorithm [31], and incorporates a high-quality external relation identifier to identify the presence of relation constraints. As illustrated in the left part of Figure 1, based on the relation identifier, RESEAL dynamically adjusts the probability of the candidate constrained words in the generation process.

Among all categories of word relations, the dependency relation is most basic, standard and representative which has various public available datasets for evaluation. Therefore, in this paper, we mainly focus on the dependency relation scenario of RCD. To illustrate the effectiveness of our proposed RESEAL, we construct a benchmark from publicly available treebanks [39] that contain sentences and their dependency trees annotated by human. We randomly sample a subset of dependency triplets as the input constraints and regard the original sentences as reference outputs. We call this task “Dependency Placement”. Experiment results show that our method outperforms baselines and LCD methods on dependency coverage (the ratio of satisfied relation constraints for dependency). After that, to showcase the applicability of this work, we further explore some potential applications of RCD. We conduct extensive experiments on three downstream tasks: sentence summarization (with dependency relations), fact-based text editing (with relations between two entities in the knowledge graph), and data-to-text generation (with relations extracted from knowledge bases). Across different tasks, we observe a consistent improvement over the strong baselines.

Note that the performance of relation identifier is crucial to the generation quality. It’s not so difficult to train an accurate relation identifier. We will further discuss this external dependence issue in Appendix D.3.
Algorithm 2 (line 1-6) shows the process of probability surgery. At time step where $M$ is the output sequence length, $y_i \in \mathcal{V}_T$ and $\mathcal{V}_T$ is the target vocabulary. The conditional probability of $Y$ given $X$ and model parameter $\theta$ can be calculated as follows:

$$p(Y|X; \theta) = \prod_{t=1}^{M} p(y_t|y_{<t}, X; \theta).$$

Eq. 1 usually acts as an objective for beam search. In this paper, we denote each relation constraint as a triplet $(h, r, \tau)$, where $h$ is the head, $\tau$ is the tail, and $r$ is the relation between them. Given an unordered relation constraints set $C = \{(h_l, r_l, \tau_l)\}_{l=1}^{L}$, where $L$ is the number of constraints and $h_l, \tau_l \in \mathcal{V}_T$, we aim to make the output $Y$ satisfy the constraints in $C$ as much as possible. For the model’s output $Y$, we denote $C'(Y) = \{(h'_l, r'_l, \tau'_l)\}_{l=1}^{M}$ be the relation triplets of $Y$, and then we propose to jointly optimize Eq. 1 and $|C \cap C'(Y)|$ as a novel objective for RCD.

2 Methodology

Algorithm 1 gives an overview of RESEAL. To start the decoding, the decoder input is initialized with a single EOS token. At each time step $t$, the model maintains the $k$-best candidate sentences, where $k$ is the beam size. The decoder produces the distribution $p_{\text{vocab}}(w|y_{<t}, X; \theta)$ for each token $w$ in the target vocabulary $\mathcal{V}_T$ (line 3). Each candidate has different $p_{\text{vocab}}$ respectively to produce $k|\mathcal{V}_T$ candidates, then we can select top-$k$ candidates from them by the cumulative log probability (line 5). The decoding ends when candidates contain $k$ finished sentences (line 6-7). Different from the standard beam search, RESEAL follows a two-step approach as follows:

Step 1: Probability Surgery (line 4) RESEAL operates the produced probability distributions according to the result of a relation identifier, which serves as an explicit signal to guarantee the presence of relation constraints.

Step 2: Relation-Guided Top-K (line 5) To satisfy lexical constraints, we replace the standard Top-K operation by the one used in DBA [31]. Furthermore, to satisfy relation constraints, RESEAL dynamically allocates beam by the results of relation identifier instead of the number of satisfied lexical constraints used in DBA.

If removing line 4 and replacing line 5 with a normal Top-K, RESEAL is equivalent to the standard beam search. We will describe the aforementioned two steps in detail in the rest of this section.

3.1 Probability Surgery

Algorithm 2 (line 1-6) shows the process of probability surgery. At time step $t$ during decoding, the model predicts the next token probability $p_{\text{vocab}}(w|y_{<t}, X; \theta)$. To provide an external signal to guarantee the presence of relation constraints, we calculate another probability distribution $p_{\text{rel}}(w|y_{<t}, C)$
We first give a formal definition of $p$ where

Algorithm 2 Probability Surgery and RG-Top-K

1: function PROB_SURGERY($p_{vocab}$, candidates, $C$, $R$)
2:   for all candidate in candidates do
3:     for all unmet lexical constraints $w$ of candidate do
4:       sent $\leftarrow$ candidate.sentence (i.e., $y_{<t} + w$)
5:       Get $p_{trans}$, $p_{type}$ by $R$ and sent, and then calculate $p_{rel}(w|y_{<t}, C)$ by Eq. 3.
6:       Calculate and normalize $\tilde{p}$ by $\tilde{p}(w|y_{<t}, X; \theta) \propto g(p_{rel}(w|y_{<t}, C)) \cdot p_{vocab}(w|y_{<t}, X; \theta)$.
7:   return $\tilde{p}$ of every candidate
8: function RG_TopK($\tilde{p}$, candidates, $C$)
9:   candidates $\leftarrow$ Generate_candidates_by_DBA($\tilde{p}$, candidates)
10:  Initialize relation_counts, bank, and pruned_candidates.
11:  for all candidate in candidates do
12:    Get $n_t$, which is the number of correct relation constraints in this candidate.
13:    Update relation_counts, and add the candidate to bank $n_t$.
14:    bank_sizes $\leftarrow$ BeamAllocate_by_DBA(relation_counts)
15:    for $j$ in $[0, |C|]$ do
16:      Add top-K candidates in bank $j$ to pruned_candidates, where $K = \text{bank_sizes}[j]$.
17: return pruned_candidates

by the external relation identifier. The $p_{rel}(w|y_{<t}, C)$ indicates the probability that the token $w$ satisfies the relation constraints $C$ given previous decoding result $y_{<t}$. The core idea of our proposed probability surgery is to combine $p_{vocab}$ and $p_{rel}$ together to produce an augmented distribution $\tilde{p}$, and then use $\tilde{p}$ instead of $p_{vocab}$ to choose the words.

We first give a formal definition of $p_{rel}(w|y_{<t}, C)$. The $p_{rel}$ depends on two factors: (1) the transition probability $p_{trans}(y_j, y_i)$, the probability that $y_j$ is the head of $y_i$, (2) the relation type probability $p_{type}(y_j, r, y_i)$, the probability that the relation between $y_j$ and $y_i$ falls in type $r$. Both $p_{trans}$ and $p_{type}$ can be obtained during decoding by a relation identifier. For dependency relations, the relation identifier can be a left-to-right dependency parser [9] to better fit the left-to-right manner of autoregressive decoding. For relations between entities, the relation identifier can be any relation extraction model. The function of relation identifier is to predict the head of $y_i$, which produces a series of $p_{trans}(y_j, y_i)$. Additionally, the relation identifier also predict the relation types, which produces a series of $p_{type}(y_j, r, y_i)$. Note that if $y_j$ is not the head of $y_i$, $p_{type}(y_j, r, y_i) = 0$ for all types $r$.

Given the incomplete output sequence $y_{<t} = (y_1, y_2, \ldots, y_{t-1})$ at time step $t$ and the next token $w$, we choose the relation constraints related to $w$ in $C$. Let $C_{\text{head}}(w, t), C_{\text{tail}}(w, t)$ denote the subset of relation constraints $C$ at time step $t$ where the $w$ serves as the head or the tail, respectively:

$$C_{\text{head}}(w, t) = \{(w, r, y)| (w, r, y) \in C \land y \in y_{<t}, \forall w \in V_T, \}$$

$$C_{\text{tail}}(w, t) = \{(y, r, w)| (y, r, w) \in C \land y \in y_{<t}, \forall w \in V_T, \}$$

(2)

We can calculate $p_{rel}(w|y_{<t}, C)$ as follows:

$$p_{rel}(w|y_{<t}, C) = \frac{1}{Z_{w,t}} \left\{ \sum_{(w, r, y) \in C_{\text{head}}(w, t)} [p_{trans}(w, y) + p_{type}(w, r, y)] + \sum_{(y, r, w) \in C_{\text{tail}}(w, t)} [p_{trans}(y, w) + p_{type}(y, r, w)] \right\},$$

(3)

where $Z_{w,t} = 2||C_{\text{head}}(w, t)|| + ||C_{\text{tail}}(w, t)||$ is a normalizing factor. Consequently, the augmented distribution can be calculated as follows:

$$\tilde{p}(w|y_{<t}, X; \theta) \propto g(p_{rel}(w|y_{<t}, C)) \cdot p_{vocab}(w|y_{<t}, X; \theta),$$

(4)

where $g : [0, 1] \rightarrow (0, 1]$ is a gate function to transform $p_{rel}$ to a weight of $p_{vocab}$. We aim to increase the weight when the $p_{rel}$ increases, so $g$ cannot be a monotonically decreasing function. Moreover, the output of this function should not be exactly zero, because assigning zero probability will make the log-likelihood of the whole sentence be negative infinite. Based on these requirements, inspired

\footnote{Note that $p_{rel}(w|y_{<t}, C) = 1$ if $C_{\text{head}}(w, t) = C_{\text{tail}}(w, t) = \emptyset$. Apart from that, we use additive form of $p_{rel}$ instead of a multiplicative manner, we further discuss this in Appendix A.}
by Schick et al. [37], we adopt this parameterized form of $g$ in this paper:

$$g(p_{rel}) = \begin{cases} e^{-\lambda(1-p_{rel})} & \text{if } p_{rel} < \rho \\ 1 & \text{otherwise} \end{cases} \quad (5)$$

where $\lambda, \rho$ are the hyperparameters and $\lambda > 0, \rho \in (0, 1]$. $\lambda$ controls the decay of output value. $\rho$ is a threshold for the probability. We fix $\rho = 0.5$ in this paper. If the $p_{rel}$ of a word $w$ is greater than or equal to this pre-specified threshold $\rho$, it is confident enough to consider $w$ satisfies the relation constraints. Thus we set $g(p_{rel}) = 1$ for this situation to preserve the $p_{vocab}$ of $w$. On the contrary, when $p_{rel}$ is close to 0, $g(p_{rel})$ will be close to a relatively small value $e^{-\lambda}$, which indicates that the model will be less likely to choose the words violating relation constraints.

### 3.2 Relation-Guided Top-K

To ensure the presence of lexical constraints, we adopt the Top-$K$ operation in DBA [31]. DBA firstly generates a set of candidates and then selects $k$ of them through beam allocation. The candidate set generated by DBA (line 8) is the union of three sets: (1) the normal Top-$K$ tokens, (2) all unsatisfied lexical constraints, and (3) the single-best token for each hypothesis in the beam. After that, DBA groups together the candidates with the same number of satisfied lexical constraints into some banks and selects a different number of candidates from different banks. The candidates with fewer lexical constraints will have more chances to be selected. However, the original DBA is not aware of the relations between words. Since we have already processed the candidate sentences by relation identifier in probability surgery, we can now use the processed result to guide the bank allocation. As illustrated in Algorithm 2, we propose to use the number of correct relation constraints of $i$-th candidates (line 10-12) to divide the banks, rather than the number of satisfied lexical constraints used by DBA. This modification can jointly consider both lexical and relation constraints, because one relation constraint is equivalent to two lexical constraints and their relation.

## 4 Experiments on Dependency Placement

There are many kinds of word relations in natural language, so it is necessary to showcase the performance of our proposed RESEAL on different relations. Since syntactic dependency structures serve as the principle of how words are combined to form sentences, dependency relations can be the most basic and important for text generation. Thus, in this section, we mainly focus on the dependency relation scenario of RCD, and evaluate RESEAL on the Dependency Placement task. Besides, we will conduct extensive experiments on three downstream tasks in Section 5.

### 4.1 Task and Dataset

We first define Dependency Placement task: given the constraints $C$ of dependency relations, output a fluent sentence $Y$ which appropriately places these constraints. The model input $X$ is optional, which can be a single ⟨BOS⟩ token, or a sequential transformation of $C$ to provide necessary information. We then construct the dataset for dependency placement task from the English-EWT [39] corpus\(^4\), which contains 16,621 sentences with dependency annotations and standard train/dev/test set split. For each sentence with $m$ words, we randomly sample $n$ dependency triplets $\{(h_i, r_i, \tau_i)\}_{i=1}^n$ as the given constrains $C$, where $n < m$. The original sentences serve as references. We refer to this dataset as English-EWT-Dep. More details about English-EWT-Dep can be found in Appendix B.

### 4.2 Evaluation Metrics

In this section, we discuss the appropriate metrics for the dependency placement task to evaluate an RCD algorithm (including but not limited to our proposed RESEAL). Simply using the automatic evaluation to compare the system outputs with the references is not suitable, because there are too many sentences that can be the correct answer given several dependencies. We mainly focus on whether the relation constraints are satisfied when examining an RCD algorithm. To this end, the output sentences should be processed again by an accurate external parser\(^5\). This parser should

\(^4\)https://universaldependencies.org/

\(^5\)Another choice is to process the outputs by human, which is too expensive.
Table 1: Evaluation result for dependency placement task. “Reference” denotes evaluating the ground truth sentences, which can be viewed as an approximated upper bound of this task. Despite that the BLEU-4 and METEOR is not so accurate to evaluate this task, we still provide it for reference only.

<table>
<thead>
<tr>
<th>Method</th>
<th>Stanza UC↑</th>
<th>spaCy UC↑</th>
<th>BLEU-4↑</th>
<th>METEOR↑</th>
<th>PPL↓</th>
<th>Word%↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base [21]</td>
<td>80.52</td>
<td>71.02</td>
<td>11.92</td>
<td>20.12</td>
<td>865.40</td>
<td>97.11</td>
</tr>
<tr>
<td>Rerank (k = 20)</td>
<td>84.32</td>
<td>74.86</td>
<td>11.66</td>
<td>20.18</td>
<td>346.44</td>
<td>99.88</td>
</tr>
<tr>
<td>CGMH [26]</td>
<td>39.46</td>
<td>24.69</td>
<td>1.47</td>
<td>14.50</td>
<td>2341.83</td>
<td>96.20</td>
</tr>
<tr>
<td>DBA [31]</td>
<td>79.54</td>
<td>68.47</td>
<td>12.23</td>
<td>20.13</td>
<td>436.27</td>
<td>99.82</td>
</tr>
<tr>
<td>DDBA [25]</td>
<td>79.22</td>
<td>70.11</td>
<td>12.22</td>
<td>20.12</td>
<td>796.76</td>
<td>97.01</td>
</tr>
<tr>
<td>NeuroLogic [24]</td>
<td>82.47</td>
<td>72.87</td>
<td>12.23</td>
<td>20.13</td>
<td>527.35</td>
<td>98.93</td>
</tr>
<tr>
<td>RESEAL</td>
<td>86.45</td>
<td>80.66</td>
<td>12.62</td>
<td>20.40</td>
<td>260.80</td>
<td>99.60</td>
</tr>
<tr>
<td>Reference</td>
<td>86.80</td>
<td>81.49</td>
<td>100.00</td>
<td>100.00</td>
<td>527.35</td>
<td>100.00</td>
</tr>
</tbody>
</table>

preferably be different from the one used in an RCD algorithm, which can better examine the generalization ability across the parsers. In this paper, we use two widely-used dependency parsers provided by Stanza [32] and spaCy for evaluation. Let \( C(i) \) denote the relation constraints of \( i \)-th output \( Y(i) \) in test set. Let \( C'(Y(i)) \) denote the dependency relation triplets obtained by the external parser. Let \( W(i) \) and \( W'(Y(i)) \) denote the sets if we omit the dependency relation \( r \) of \( C(i) \) and \( C'(Y(i)) \). Similar to the unlabeled/labeled attachment score (UAS/LAS) used in dependency parsing, we can define the unlabeled/labeled coverage (UC/LC) as follows:

\[
UC = \frac{\sum_i |W(i) \cap W'(Y(i))|}{\sum_i |W(i)|}, \quad LC = \frac{\sum_i |C(i) \cap C'(Y(i))|}{\sum_i |C(i)|}. \tag{6}
\]

Moreover, we report the BLEU-4 [30], METEOR [2], GPT-2 [33] perplexity (PPL) and word coverage (the proportion of lexical constraints that are satisfied).

4.3 Baselines

Since there are no existing work about dependency placement, we design some straightforward baselines to compare with our method:

- **Base**: Use BART\(_{\text{large}}\) [21] as the backbone, and then finetune it on English-EWT-Dep. The input is the concatenation of the triplets of \( C \) separated by special token \#. For example, if \( C = \{(h_1, r_1, \tau_1), (h_2, r_2, \tau_2)\} \), the input sequence \( X = h_1\#r_1\#\tau_1, h_2\#r_2\#\tau_2 \). The target output is the reference. During decoding, we use standard beam search with beam size \( k = 20 \).

- **CGMH** [26]: Use MCMC sampling to generate a sentence by modifying it. We use BERT\(_{\text{large}}\) [6] to produce its replacement probability, and GPT-2\(_{\text{large}}\) [33] as its language model.

- **X-MCMC** [13]: Improve CGMH by using XLNet [46]. X-MCMC-C adds a classifier to instruct the X-MCMC models where and how to modify the candidate sentences.

- **DBA** [31]: Use DBA algorithm to decode on the Base model with beam size \( k = 20 \).

- **DDBA** [25]: A denoised variant of DBA by filtering noisy constraints.

- **NeuroLogic** [24]: A LCD algorithm which support complex lexical constraints in Conjunctive Normal Form (CNF).

- **Rerank**: Preserve all \( k \) sentences generated by DBA, and select the sentence that satisfies the most relation constraints using left-to-right parser [9].

We report the results when applying RESEAL to the Base. More details can be found in Appendix C.1.

\[\text{https://explosion.ai/blog/ud-benchmarks-v3-2#project}\]
Table 2: Result of ablation study for dependency placement task. We remove a single component from the full algorithm to study the individual effect. “w/o prob” denotes without probability surgery (but still with RG-Top-K), “w/o RG-Top-K” denotes using the way of original DBA to allocate beam (but still with probability surgery). “word+rel” denotes using the number of satisfied lexical and (dependency) relation constraints to allocate beam.

<table>
<thead>
<tr>
<th>Method</th>
<th>Stanza UC↑</th>
<th>Stanza LC↑</th>
<th>spaCy UC↑</th>
<th>spaCy LC↑</th>
<th>BLEU-4↑</th>
<th>METEOR↑</th>
<th>PPL↓</th>
<th>Word%↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESEAL</td>
<td>86.45</td>
<td>79.26</td>
<td>86.73</td>
<td>80.66</td>
<td>12.62</td>
<td>20.40</td>
<td>260.80</td>
<td>99.60</td>
</tr>
<tr>
<td>w/o prob</td>
<td>81.70</td>
<td>70.91</td>
<td>82.12</td>
<td>72.21</td>
<td>11.83</td>
<td>20.19</td>
<td>256.88</td>
<td>99.80</td>
</tr>
<tr>
<td>w/o RG-Top-K</td>
<td>82.12</td>
<td>74.27</td>
<td>83.03</td>
<td>75.49</td>
<td>11.89</td>
<td>20.21</td>
<td>361.96</td>
<td>99.76</td>
</tr>
<tr>
<td>word+rel</td>
<td>82.40</td>
<td>74.51</td>
<td>83.34</td>
<td>75.42</td>
<td>11.08</td>
<td>19.81</td>
<td>452.50</td>
<td>99.72</td>
</tr>
</tbody>
</table>

Figure 2: The result when altering the value of $\lambda$ and beam size $k$.

4.4 Discussion

Results Table 1 shows the results for dependency placement. We find that the sampling-based method [13, 26] can achieve better word coverage, but the low UC/LC scores demonstrate that they fail to enforce the relation constraints. RESEAL achieves best UC/LC among all the methods with significant decline of the PPL and competitive word coverage. Specifically, RESEAL gains an improvement of 1.15 on BLEU-4 and 11.87%/12.19% (Stanza/spaCy) on LC compared to DBA. These results illustrate the weakness of existing LCD algorithms to correctly place the multiple relation constraints. Some LCD methods (DBA, DDBA, NeuroLogic) would forcibly add the unsatisfied lexical constraints into the candidate word set. This is the key to ensuring the presence of lexical constraints. However, doing this fails to consider the relations between words, thus would make the generated sentence less fluent. Apart from that, RESEAL outperforms Rerank on almost all the metrics. An intrinsic reason for this may be that RESEAL can dynamically adjust the word selection according to the parsing result during decoding, but Rerank can not make a choice until finishing all the sentences.

Ablation Study Table 2 shows the results of ablation study. We observe a decrease of 0.79 on BLEU-4 and 8.35%/8.45% (Stanza/spaCy) on LC by removing probability surgery. We also observe a decrease of 0.73 on BLEU-4 and 4.99%/5.17% (Stanza/spaCy) on LC by removing RG-Top-K. Probability surgery enables the tokens with correct dependencies to enter the candidate set. RG-Top-K dynamically allocates the beam according to the parsing result to satisfy more relation constraints. These two components are both crucial. Furthermore, we find that adopting another beam allocation strategy (“word+rel” in Table 2) also hurt the performance.

Impact of Hyperparameters For the dependency placement task, the performance would benefit from a larger beam size, which is consistent with the observations of Post and Vilar [31]. The left of Figure 2 shows the labeled coverage (LC) as a function of beam size by different methods on the test set. We observe that RESEAL can achieve better LC scores with smaller beam sizes when compared with Rerank. Apart from that, the decay factor $\lambda$ introduced in Section 3.1 is another important hyperparameter of RESEAL. We investigate its influence on the model performance. Figure 2 shows
the results when varying $\lambda$ from 0 to 10 based on the validation set. If we set $\lambda$ as a relatively large value (> 4), the performance of RESEAL tends to be stable. Specifically, $\lambda = 0$ is equivalent to removing probability surgery (from Eq. 5), which will result in worse performance.

**Case Study** Table 4 shows the sentences generated by all listed methods for dependency placement task. The **Base** model may omit some lexical constraints. DBA and Rerank satisfy all the lexical constraints, but they cannot correctly handle the relations between them. Rerank may generate sentences with repetitions (e.g., the word “charge”). RESEAL can produce sentences that are close to the grammatical structure of the references.

**More Discussions** In Appendix D, we discuss the limitations of RESEAL, including the impact of dependency parsers, time complexity and external dependence issue. In Appendix E, we provide more cases for dependency placement task. In Appendix F, we discuss the social impact of this work.

## 5 Experiments on Downstream Tasks

To further explore the effectiveness of our proposed RESEAL, we conduct experiments on the three downstream tasks: sentence summarization, fact-based text editing, and data-to-text generation. Intuitively, RESEAL can aid these tasks. For the sentence summarization, RESEAL may help to preserve the important relations in the source sentence. For the rest two tasks, RESEAL can help to incorporate the given factual relations.

### 5.1 Sentence Summarization

**Dataset** We conduct experiments on English Gigaword dataset [36], which contains about 3.8M training sentence pairs. We use the validation and test set provided by Zhou et al. [49] with 8,000 and 2,000 sentence pairs, respectively. Following previous work, we also evaluate our model on the test set of DUC2004 [29] (with 500 input sentences) and MSR-ATC [42] (with 785 input sentences).

**Dependency Prediction** To apply RESEAL to sentence summarization, we first need to obtain reasonable dependency triples to construct relation constraints. In this paper, we train a vanilla BERT-base-uncased [6] model to predict which dependencies should be present in the target output. Firstly, we parse the sentences in the dataset by the left-to-right parser [9]. Then we use the intersection of dependencies of source and target sentences as the ground truth to train the dependency predictor. For each triplet $(h, r, \tau)$, we concatenate the contextual embedding of $h$, $\tau$ and label embedding of $r$ to perform binary classification. More details can be found in Appendix C.2.
Table 5: Evaluation result of WebEdit dataset.

<table>
<thead>
<tr>
<th>Methods</th>
<th>BLEU-4</th>
<th>SARI</th>
<th>KEEP</th>
<th>ADD</th>
<th>DELETE</th>
</tr>
</thead>
<tbody>
<tr>
<td>EncDecEditor [17]</td>
<td>71.03</td>
<td>69.59</td>
<td>89.49</td>
<td>43.82</td>
<td>75.48</td>
</tr>
<tr>
<td>FactEditor [17]</td>
<td>75.68</td>
<td>72.20</td>
<td>91.84</td>
<td>47.69</td>
<td>77.07</td>
</tr>
<tr>
<td>Seq2Seq</td>
<td>82.96</td>
<td>73.74</td>
<td>93.62</td>
<td>64.56</td>
<td>69.07</td>
</tr>
<tr>
<td>Seq2Seq+RESEAL</td>
<td>84.12</td>
<td>78.33</td>
<td>96.07</td>
<td>69.63</td>
<td>69.29</td>
</tr>
</tbody>
</table>

Table 6: Evaluation result for WebNLG test set.

<table>
<thead>
<tr>
<th>Methods</th>
<th>BLEU-4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Castro Ferreira et al. [4]</td>
<td>51.68</td>
</tr>
<tr>
<td>Moryossef et al. [27]</td>
<td>47.24</td>
</tr>
<tr>
<td>Zhao et al. [48]</td>
<td>52.78</td>
</tr>
<tr>
<td>Harkous et al. [12]</td>
<td>52.90</td>
</tr>
<tr>
<td>Nan et al. [28]</td>
<td>45.89</td>
</tr>
<tr>
<td>T5-small [34]</td>
<td>56.34</td>
</tr>
<tr>
<td>T5-small + RESEAL</td>
<td>56.87</td>
</tr>
<tr>
<td>T5-base [34]</td>
<td>59.17</td>
</tr>
<tr>
<td>T5-base + RESEAL</td>
<td>59.59</td>
</tr>
</tbody>
</table>

Results Following previous work [18, 49], we report ROUGE F1 [23] on Gigaword and MSR-ATC, and ROUGE recall on DUC2004. Table 3 show the results. BART consistently outperforms the previous models without pretraining across different datasets. Over the strong BART baseline, RESEAL can achieve better ROUGE scores on all datasets with predicted relation constraints. We also investigate the upper bound of BART+RESEAL by using the gold relation constraints, which shows that there is room for improvement with more accurate dependency predictors.

5.2 Fact-Based Text Editing

Dataset Fact-based Text Editing is a novel task proposed by Iso et al. [17]. Given some triplets (facts) from knowledge graphs and a draft text, this task aims to revise the draft text to contain these facts. We adopt the WebEdit dataset provided by Iso et al. [17], which contains 181K/23K/29K instances as train/valid/test set. Note that this dataset can be viewed as a natural scenario for RCD because both the relation constraints (facts) and model input (draft text) are provided. More importantly, based on the results of error analysis, the models trained on this dataset still suffer from missing facts and incorrect relations (see “Qualitative evaluation” section in [17]). RESEAL may alleviate these problems by explicitly enforcing facts and relations.

5.3 Data-to-Text Generation

Models and Results For the relation identifier, we train a simple BiLSTM [14] encoder with biaffine attention [8] on the training set of WebEdit (See Appendix C.3.1 for more details). We use EncDecEditor and FactEditor reported by Iso et al. [17] as our baseline models. EncDecEditor is an encoder-decoder model based on LSTM, with two separate encoders for facts and drafts and a decoder for generating revised texts. FactEditor shares the same encoders with EncDecEditor but has a novel decoder that doesn’t follow the conventional autoregressive decoding manner. Thus we can only apply RESEAL to the EncDecEditor. For a fair comparison, we do not use pretrained models and keep the same architecture setting as EncDecEditor. However, the source code and the training details of EncDecEditor are unreleased, thus we reimplement EncDecEditor using our own training configuration (See Appendix C.3.2). We denote this model as Seq2Seq. Table 5 shows the experiment results of WebEdit. Following Iso et al. [17], we report the BLEU-4 and SARI [45] score (the average F1-score for keep, add and delete operations). Owing to the difference of training settings, Seq2Seq can achieve significant improvement on BLEU-4 (+7.28) and SARI (+1.54) compared to FactEditor. Based on this result, we further adopt RESEAL on Seq2Seq and then observe a substantial improvement on BLEU-4 (+1.16) and SARI (+4.59). The above results demonstrate that RESEAL can achieve better facts and relations preservation over the strong Seq2Seq baseline.

Dataset Data-to-text generation is another direct application for RESEAL. In this paper, we adopt WebNLG dataset [11] which provides facts as inputs and sentences containing these facts as outputs. We use the data provided by Ribeiro et al. [35] which contains 18,102/872/1,862 instances as train/valid/test set. Each test instance has 1-3 references.

Models and Results For WebNLG dataset, the setting of the relation identifier is the same as that for WebEdit. Table 6 shows the experiment results of WebNLG dataset. We adopt our RESEAL on
T5 [34] and report the BLEU-4 score for evaluation. We observe an improvement of 0.53 BLEU-4 on T5-small and 0.42 BLEU-4 on T5-base. The detailed experimental settings can be found in Appendix C.4.

6 Related Work

**Lexically Constrained Decoding (LCD)** Prior explorations for LCD can be summarized into four categories. The first line of studies has proposed some model-agnostic methods which only modify the decoding process. They are independent from the training. Hokamp and Liu [15] propose the grid beam search (GBS) algorithm, a modification to beam search to impose the lexical constraints. Post and Vilar [31] propose the dynamic beam allocation (DBA) algorithm with less time complexity. Vectorized DBA [16] and Denoised DBA [25] are two different DBA variants. The second line of studies requires some modifications to the training process. Augmenting the training data with the lexical constraints is a general approach [5, 7, 40]. Another branch of previous works focuses on adding additional structure to the model [41, 43, 44]. The fourth line of studies applies Markov Chain Monte Carlo (MCMC) to constrained text generation in a refinement manner [13, 26, 38]. Different from these methods, we do not only focus on the isolated lexical constraints. We propose to adopt relation constraints to consider the relationship between words.

**Dependency-Guided Generation** Dependency is a natural way to represent the syntactic or semantic relations between words, so it can be used to guide the text generation. There are few works exploring this. Filippova and Strube [10] propose to compress the dependency graph to guide the sentence fusion. Akoury et al. [1] propose to predict a chunked syntactic parse tree and then generate tokens conditioned on the parse. Jin et al. [18] encode the dependency relations by a graph encoder to improve sentence summarization. Casas et al. [3] propose a language model where the generation is driven by the expansion over the dependency parse tree. Yang and Wan [47] propose a dependency modeling objective to incorporate dependency knowledge. However, one drawback of these methods is the limitation of interpretability and controllability since they only use the dependency as a latent variable during training, and cannot explicitly control the generation at the inference stage.

7 Conclusion

In this paper, we explore the Relation-Constrained Decoding (RCD), a new decoding scenario with a more complex definition of constraints. We propose RESEAL, a novel algorithm for RCD, which can be applied to the decoder of different models to preserve specific relation constraints. We examine two different experiment settings of RCD: dependency placement and downstream tasks. For dependency placement, we construct the benchmark for dependency placement, and the experiment results show the strength of RESEAL for satisfying relation constraints. Furthermore, we apply RESEAL to three downstream tasks as extended experiments for practical applications. Extensive experiments demonstrate the effectiveness and universality of our method.

Acknowledgment

This work was supported by National Key R&D Program of China (2021YFF0901502), National Science Foundation of China (No. 62161160339), State Key Laboratory of Media Convergence Production Technology and Systems and Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology). We appreciate the anonymous reviewers for their helpful comments. Xiaojun Wan is the corresponding author.

References


**Checklist**

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default [TODO] to [Yes], [No], or [N/A]. You are strongly encouraged to include a justification to your answer, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section 4.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper’s contributions and scope? [Yes]  
   (b) Did you describe the limitations of your work? [Yes] See Appendix D  
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Appendix F  
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]

3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix C
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix C

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Section 4 and Section 5
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] See Section 4 and Appendix B
   (d) Did you discuss whether and how consent was obtained from people whose data you’re using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes]

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]
A More Explanations about RESEAL

Both \( C_{\text{head}}(w, t) \) and \( C_{\text{tail}}(w, t) \) can be empty set as \( C \) can be arbitrary. Noted that in this part we simply sum over all \( p_{\text{trans}} \) and \( p_{\text{type}} \) to produce \( p_{\text{rel}} \), which is different from the method we use when combining \( p_{\text{vocab}} \) and \( p_{\text{rel}} \). The reason why we use this form is that if substituting Eq. 5 and 3 into Eq. 4, we can derive a decomposed form:

\[
g(p_{\text{rel}}(w|y_{<t}, C)) = \begin{cases} \prod \left[ g(p_{\text{trans}}(w, y)) \cdot g(p_{\text{type}}(w, y, r)) \right] \cdot g(p_{\text{trans}}(y, w)) \cdot g(p_{\text{type}}(y, w, r)) \right] \cdot Z_{w, t} \end{cases} \tag{7}
\]

Eq. 7 indicates that we assign high probability to \( w \) at time step \( t \) if and only if all the relation constraints in \( C_{\text{head}}(w, t) \cup C_{\text{tail}}(w, t) \) are satisfied. Any violation of the constraints will result in a small value of \( g(p_{\text{rel}}(w|y_{<t}, C)) \) and reduce the augmented probability \( \tilde{p} \) of \( w \) according to Ed. 4.

B Dataset Construction for English-EWT-Dep

In this section, we give more details about the data construction process of English-EWT-Dep introduced in Section 4.1. We only consider the dependency relations whose modifier is an adjective, noun, or verb and randomly sample parts of them (at a ratio of 40%). The sentences which are shorter than 4 words are omitted. Some sentences containing personal information (e.g. phone number or address) are also discarded. The statistics of the English-EWT-Dep dataset is as follows:

<table>
<thead>
<tr>
<th></th>
<th>train</th>
<th>dev</th>
<th>test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sentence</td>
<td>10119</td>
<td>1454</td>
<td>1407</td>
</tr>
<tr>
<td>Avg word count</td>
<td>19.39</td>
<td>15.98</td>
<td>16.19</td>
</tr>
<tr>
<td>Avg number of dependencies</td>
<td>2.43</td>
<td>2.04</td>
<td>2.04</td>
</tr>
<tr>
<td>Max number of dependencies</td>
<td>18</td>
<td>13</td>
<td>11</td>
</tr>
</tbody>
</table>

C Implementation Details

We provided the implementation details for the four tasks included in this work: dependency placement, sentence summarization, fact-based text editing and data-to-text generation. Note that in this paper, we conduct all the experiments on a NVIDIA A40 GPU with 40 GB memory.

C.1 Dependency Placement

For the Base model of dependency placement task, we use BART\textsubscript{large} [21] as our pretrained model. We use label smoothing of 0.1 and optimize the model by Adam [19]. The learning rate is 2e-5 with a warmup rate of 0.1. The weight decay is 0.1. We train the Base model for 10 epochs with a batch size of 32.

C.2 Sentence Summarization

For the sentence summarization task, we also use BART\textsubscript{large} [21] as our pretrained model. The hyperparameter setting is similar to the dependency placement except that we only train for 5 epochs with a batch size of 100. The ROUGE evaluation option for GigaWord and MSR-ATC is -m -n 2 -w 1.2. The ROUGE evaluation option for DUC2004 is -m -b 75 -n 2 -w 1.2.
For the dependency predictor used in this task, we adopt a BERT-base-uncased [6] model. Since the dependency is a word-level relation and BERT uses sub-word tokenization, we use the mean vector of the start and the end token as the representation \( f(w) \) of a word \( w \):

\[
f(w) = \frac{1}{2} \left[ \text{BERT(start token of } w) + \text{BERT(end token of } w) \right].
\]  

(8)

We use an additional embedding layer with the dimension of 200 to represent the dependency label \( r \). Let \( f_L(r) \) denotes the representation of \( r \), the feature of \( (h, r, w) \) is defined as the concatenation of \( f(h), f(w) \) and \( f_L(r) \).

We optimize the dependency prediction model by Adam [19]. The learning rate is 2e-5 with a warmup rate of 0.1. The weight decay is 0.1. We train this model for 10 epochs with a batch size of 200. After training, for each sentence in test set, we select the dependency relation constraints with predicted probability larger than a threshold \( \rho = 0.6 \).

It’s worth noting that our reported performance of sentence summarization can be further improved by training a more accurate dependency predictor, or using a single keyword extractor and preserving the corresponding dependencies of predicted keywords. We leave more in-depth explorations as future work.

### C.3 Fact-Based Text Editing

Different from sentence summarization task, fact-based text editing does not require to predict the relation constraints, because the relation triplets have been provided in the dataset. Therefore, to adopt RESEAL, we only need to train the relation identifier and the LSTM [14] seq2seq model.

#### C.3.1 Relation Identifier for Fact-Based Text Editing

The structure of relation identifier for fact-based text editing is borrowed from the biaffine dependency parser [8]. We use a single layer LSTM with single direction. We use two separate MLPs with one hidden layer to extract the features of heads and tails, respectively. Finally, we use a biaffine attention module to classify the relation types. Moreover, following the setting of Iso et al. [17], we use a simple embedding lookup table to embedding the input words. The detailed hyperparameter setting is shown as follows:

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Embedding size</td>
<td>300</td>
</tr>
<tr>
<td>LSTM hidden size</td>
<td>300</td>
</tr>
<tr>
<td>MLP hidden size</td>
<td>512</td>
</tr>
<tr>
<td>MLP output size</td>
<td>300</td>
</tr>
<tr>
<td>Number of relation categories</td>
<td>229</td>
</tr>
<tr>
<td>Maximum source length</td>
<td>128</td>
</tr>
<tr>
<td>Learning rate</td>
<td>1e-3</td>
</tr>
<tr>
<td>Dropout</td>
<td>0.5</td>
</tr>
<tr>
<td>Batch size</td>
<td>300</td>
</tr>
<tr>
<td>Total epochs</td>
<td>10</td>
</tr>
</tbody>
</table>

The training data for the relation identifier can be heuristically constructed from WebEdit dataset in a distant supervision manner, which is also the mainstream scheme for dataset construction of relation extraction models. We consider the target texts (revised text) in WebEdit as the inputs, and then use the given facts to match the words in revised texts. We omit the instances where the head word and the tail word are not in the same sentence, because in this situation it is difficult to determine whether these two words have given relations. Furthermore, we construct many negative samples from the positive samples by randomly deleting the words between the head words and the tail words. The relations between the head words and the tail words in negative samples are annotated as “no relation”. We add these negative samples into training set and test set to improve the robustness of our relation identifier.

With the above mentioned model and augmented training data, we obtain a robust relation identifier for WebEdit, which can achieve about 93 F1 score in the test set.
C.3.2 Training Configuration

We reimplement the EncDecEditor [17] and denote it as Seq2Seq. The original paper only report the architecture of FactEditor [17]. Therefore, we have to make our Seq2Seq model have the same size as the FactEditor. The detailed hyperparameter setting is shown as follows:

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Embedding size</td>
<td>300</td>
</tr>
<tr>
<td>Table Encoder hidden size</td>
<td>300</td>
</tr>
<tr>
<td>Text Encoder hidden size</td>
<td>300</td>
</tr>
<tr>
<td>Decoder hidden size</td>
<td>600</td>
</tr>
<tr>
<td>Maximum source length</td>
<td>128</td>
</tr>
<tr>
<td>Learning rate</td>
<td>1e-3</td>
</tr>
<tr>
<td>Teacher forcing ratio</td>
<td>0.5</td>
</tr>
<tr>
<td>Dropout</td>
<td>0.5</td>
</tr>
<tr>
<td>Batch size</td>
<td>200</td>
</tr>
<tr>
<td>Total epochs</td>
<td>30</td>
</tr>
</tbody>
</table>

Note that we use the same vocabulary as the corresponding relation identifier. Similar to EncDecEditor and FactEditor, our Seq2Seq model also employs attention and copy mechanism.

C.4 Data-to-Text Generation

In this part, we describe more details about the experiments of data-to-text generation task.

C.4.1 Entity and Relation Prediction

The settings of data-to-text generation is similar with that of fact-based text editing. The major difference is that the triplets given in WebNLG dataset cannot directly serve as relation constraints. The entities given in the source inputs may not has the same form in the references. Therefore, similar to the dependency prediction in sentence summarization task, we need to predict whether an entity should appear in its output.

In this paper, we directly use the encoder of T5 [34] to predict how to preserve the entities. Specifically, for all the triplets \((h, w, \tau)\), we transform them into a sequential form. Then we perform the sequence labeling task to predict which entities should be preserved in the output. This prediction task can be optimized jointly with the standard seq2seq learning. It must be pointed out that according to our experiment result, this prediction task itself does not influence the performance of data-to-text generation, so adding this task is not the reason for our performance gains. We can obtain the entities and their corresponding relations and set them as relation constraints during the inference stage by adopting RESEAL, which achieves better relation preservation and enhance the performance.

C.4.2 Relation Identifier for Data-to-Text Generation

The relation identifier on WebNLG dataset shares the same architecture with the one used in fact-based text editing. The only difference is the vocabulary. Due to the fact that we use T5 as our backbone for WebNLG, we adopt the tokenizer used in T5 to make the tokenizer consistent. The vocabulary size is 32,128.

C.4.3 Training Configuration

For the training settings for the WebNLG dataset, we train the T5-small and T5-base with a learning rate of 5e-5 and a batch size of 20 for 30 epochs. We optimize the model by Adam [19]. No weight decay and label smoothing are applied. The maximum length for both source and target are 384.
D Limitation Discussions

D.1 Impact of the Relation Identifier

For dependency placement task, there are two types of dependency parser. The left-to-right parser [9] is used for decoding. Stanzan [32] and spaCy are used for the evaluation of algorithm.

For the parser which is used for decoding, the result will benefit from a more accurate parser. For the parser which is used for the evaluation, a more accurate one would increase the absolute values of UC and LC. For more precise evaluation, one can use multiple such parsers to produce an average result.

D.2 Time Complexity Analysis

Similar to most of decoding-only LCD algorithms, RESEAL is usually slower than the standard beam search, but the overall runtime is within an acceptable range. The bottleneck of decoding speed is the times of decoder forward propagation. For a sentence of length \( N \), beam search and DBA take \( N \) times forward. Our method should be slower due to the relation identifier. There are at most \( 2|C| \) candidate sentences to be parsed at time step \( t \). Therefore, it will at most take \( N(1 + 2|C|) \) times of forward for RESEAL, which still has \( O(N) \) time complexity. A faster implementation of RESEAL is to save the outputs and hidden states obtained by the relation identifier at time step \( t \), and reuse them at next time step \( t + 1 \). This implementation is left for the future work.

D.3 External Dependence Issue

Our proposed RESEAL relies on the external relation identifier. If the relation identifier is poor, it would somewhat mislead the generation process and hurt the overall generation quality anyway. However, at least in our experiment, the relation identifier is of high quality, and we believe that in the tasks of this work, it’s not so difficult to train a relatively strong relation identifier.

For dependency placement and sentence summarization tasks, we use the left-to-right dependency parser [9] as the relation identifier. Training the parser is easier and faster than training a language model. The training data is from the English Universal Dependencies Treebank, which is publicly available. The parser achieves 90.93 UAS and 88.99 LAS on the test set of this benchmark, which shows its effectiveness for predicting dependency relations.

For fact-based text editing and data-to-text tasks, we use a biaffine attention model [8] as the relation identifier. We use a single layer LSTM with single direction to extract features. We use two separate MLPs with one hidden layer to extract the features of heads and tails, respectively. Finally, we use a biaffine attention module to classify the relation types. Training this model is also easier and faster than training a language model. The training data can be heuristically constructed from the training set of these two tasks: we consider the target sentences as the inputs, and then use the given fact triple (head, relation, tail) to match the words in the target sentences. The relation identifier reaches 93.14 F1 score on the test set, which shows its effectiveness for predicting factual relations.

Thus we believe that all the relation identifiers used in our work are strong enough to produce high-quality results. Training a stronger relation identifier would definitely enhance the performance, but that would be outside the scope of our paper because our paper focuses on the text generation, not the relation extraction task.

E Case Study

Table 8 shows more examples for the dependency placement task.

F Impact Statement

Our work has introduced a generic decoding method for Relation-Constrained Decoding (RCD). Similar to most of text generation techniques, our proposed RESEAL has a potential risk of being deployed to generate human-like fake text. We suggest that the users or the programmers can utilize the relation constraints to carefully control the generation, e.g., avoiding generating the wrong facts. Therefore, we still believe that the societal impacts of RESEAL are limited and under control.
<table>
<thead>
<tr>
<th>Method</th>
<th>Generated Sentences</th>
<th>Predicted/Gold Dependencies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base</td>
<td>If you 've never tried a brick , try the burger .</td>
<td>['try', 'bruger', 'obl']</td>
</tr>
<tr>
<td>DBA</td>
<td>I 'm going to try the burger and the brick .</td>
<td>['try', 'bruger', 'obl']</td>
</tr>
<tr>
<td>Rerank</td>
<td>If you 're looking for a good burger , try the brick .</td>
<td>['try', 'bruger', 'obl']</td>
</tr>
<tr>
<td>RESEAL</td>
<td>I would not try this brick for a burger .</td>
<td>['try', 'brick', 'obj'], ['try', 'bruger', 'obl']</td>
</tr>
<tr>
<td>Ref</td>
<td>Along with the great burger try a brick of onion rings if you are with someone .</td>
<td>['try', 'brick', 'obj'], ['try', 'bruger', 'obl']</td>
</tr>
<tr>
<td>Base</td>
<td>I 'll come back to work when I call .</td>
<td>None</td>
</tr>
<tr>
<td>DBA</td>
<td>I 'll come back to work when I call .</td>
<td>None</td>
</tr>
<tr>
<td>Rerank</td>
<td>I 'll come back to work when I call .</td>
<td>None</td>
</tr>
<tr>
<td>RESEAL</td>
<td>I 'll come back on a call when I 'm back at work .</td>
<td>['come', 'call', 'obl'], ['back', 'work', 'obl']</td>
</tr>
<tr>
<td>Ref</td>
<td>Ps I may have to come back to work for a 615 call .</td>
<td>['come', 'call', 'obl'], ['back', 'work', 'obl']</td>
</tr>
<tr>
<td>Base</td>
<td>I understand that you do not have the approval to leave , but I would like to know if you can give me an update on this .</td>
<td>['understand', 'ccomp', 'have'], ['have', 'obj', 'approval']</td>
</tr>
<tr>
<td>DBA</td>
<td>I understand that you do not have the approval of all of the parties , but I would like to leave it at that .</td>
<td>['understand', 'ccomp', 'have'], ['have', 'obj', 'approval']</td>
</tr>
<tr>
<td>Rerank</td>
<td>I understand that you do not have the approval of the President of the United States , but I would like to leave it at that .</td>
<td>['understand', 'ccomp', 'have'], ['have', 'obj', 'approval']</td>
</tr>
<tr>
<td>RESEAL</td>
<td>I understand that you do not have the approval of all of the parties , but I will leave it at that .</td>
<td>['understand', 'ccomp', 'have'], ['have', 'obj', 'approval'], ['understand', 'conj', 'leave']</td>
</tr>
<tr>
<td>Ref</td>
<td>I understand you may not have credit approval yet so perhaps we can leave the appropriate sections blank in the meantime .</td>
<td>['understand', 'ccomp', 'have'], ['have', 'obj', 'approval'], ['understand', 'conj', 'leave']</td>
</tr>
<tr>
<td>Base</td>
<td>At the time , there were no clients in the facility and the staff rushed out of the building as soon as there was a problem .</td>
<td>['clients', 'facility', 'nmod']</td>
</tr>
<tr>
<td>DBA</td>
<td>The clients of this facility were rushed out of the building by the time I got there and there was no one there to help me .</td>
<td>['clients', 'facility', 'nmmod'], ['rushed', 'time', 'obl'], ['rushed', 'was', 'advcl']</td>
</tr>
<tr>
<td>Rerank</td>
<td>The clients of this facility were rushed out of the building by the time I got there and there was no one there to help me .</td>
<td>['clients', 'facility', 'nmmod'], ['rushed', 'time', 'obl'], ['rushed', 'was', 'advcl']</td>
</tr>
<tr>
<td>RESEAL</td>
<td>When there was clients in facility , they rushed out of the building at the same time as the clients were leaving .</td>
<td>['was', 'clients', 'nsubj'], ['rushed', 'time', 'obl'], ['rushed', 'was', 'advcl'], ['clients', 'facility', 'nmmod']</td>
</tr>
<tr>
<td>Ref</td>
<td>Overpriced and the doctor acted arrogant and rushed at a time when there was very few clients in the facility .</td>
<td>['was', 'clients', 'nsubj'], ['rushed', 'time', 'obl'], ['rushed', 'was', 'advcl'], ['clients', 'facility', 'nmmod']</td>
</tr>
</tbody>
</table>