FELIX: Flexible Text Editing Through Tagging and Insertion

Jonathan Mallinson[*] Google Research
Aliaksei Severyn[*] Google Research
Eric Malmi Google Research
Guillermo Garrido Google Research
{jonmall,severyn,emalmi,ggarrido}@google.com

Abstract

We present FELIX – a flexible text-editing approach for generation, designed to derive maximum benefit from the ideas of decoding with bi-directional contexts and self-supervised pre-training. In contrast to conventional sequence-to-sequence (seq2seq) models, FELIX is efficient in low-resource settings and fast at inference time, while being capable of modeling flexible input-output transformations. We achieve this by decomposing the text-editing task into two sub-tasks: *tagging* to decide on the subset of input tokens and their order in the output text and *insertion* to in-fill the missing tokens in the output not present in the input. The *tagging* model employs a novel Pointer mechanism, while the *insertion* model is based on a Masked Language Model (MLM). Both of these models are chosen to be non-autoregressive to guarantee faster inference. FELIX performs favourably when compared to recent text-editing methods and strong seq2seq baselines when evaluated on four NLG tasks: Sentence Fusion, Machine Translation Automatic Post-Editing, Summarization, and Text Simplification.

1 Introduction

The ideas of text in-filling coupled with self-supervised pre-training of deep Transformer networks on large text corpora have dramatically changed the landscape in Natural Language Understanding. BERT (Devlin et al., 2019) and its successive refinements RoBERTa (Liu et al., 2019), ALBERT (Lan et al., 2019) implement this recipe and have significantly pushed the state-of-the-art on multiple NLU benchmarks such as GLUE (Wang et al., 2018) and SQuAD (Rajpurkar et al., 2016). More recently, masked or in-filling style objectives for model pretraining have been applied to seq2seq tasks, significantly pushing the state-of-the-art

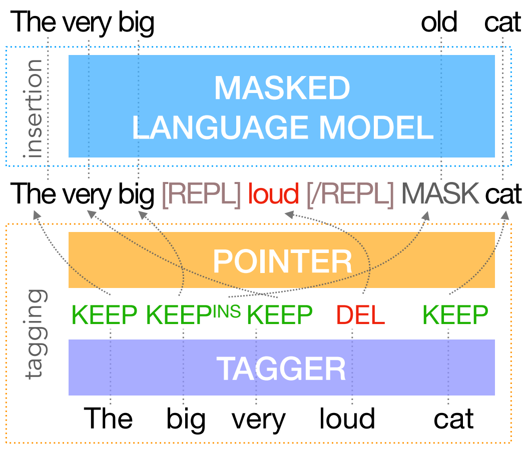

Figure 1: FELIX transforms the source *"The big very loud cat"* into the target text *"The very big old cat"*.

on a number of text generation tasks, e.g, KERMIT (Chan et al., 2019), MASS (Song et al., 2019), Bert2Bert (Rothe et al., 2020), BART (Lewis et al., 2020) and T5 (Raffel et al., 2019).

While seq2seq frameworks offer a generic tool for modeling almost any kind of text-to-text transduction, there are still many real-world tasks where generating target texts completely from scratch—as is done with seq2seq approaches—can be unnecessary. This is especially true for monolingual settings where input and output texts have relatively high degrees of overlap. In such cases a natural approach is to cast conditional text generation as a text-editing task, where the model learns to reconstruct target texts by applying a set of edit operations to the inputs. Typically, the set of edit operations is fixed and pre-defined ahead of time, which on one hand limits the flexibility of the model to reconstruct arbitrary output texts from their inputs, but on the other leads to higher sample-efficiency as the limited set of allowed operations significantly reduces the search space. Based on this observation, text-editing approaches have recently re-gained sig-

[*] Equal contribution.

nificant interest (Gu et al., 2019; Dong et al., 2019; Awasthi et al., 2019; Malmi et al., 2019).

In this paper we present a novel text-editing framework, FELIX, which is heavily inspired by the ideas of bi-directional decoding (slot in-filling) and self-supervised pre-training. In particular, we have designed FELIX with the following requirements:

Sample efficiency. Training a high-precision text generation model typically requires large amounts of high-quality supervised data. Self-supervised techniques based on text in-filling have been shown to provide a crucial advantage in low-resource settings. Hence, we focus on approaches able to benefit from already existing pre-trained language models such as BERT, where the final model is directly fine-tuned on the downstream task. We show that this allows us to train on as few as 450 datapoints.

Fast inference time. Achieving low latencies when serving text-generation models typically requires specialized hardware and finding a trade-off between model size and accuracy. One major reason for slow inference times is that text-generation models typically employ an autoregressive decoder, i.e., output texts are generated in a sequential non-parallel fashion. To ensure faster inference times we opt for keeping FELIX fully *non-autoregressive*, resulting in two orders of magnitude speedups.

Flexible text editing. While simplifying the learning task, text-editing models are not as powerful as general purpose sequence-to-sequence approaches when it comes to modeling arbitrary input-output text transductions. Hence, we strive to strike a balance between the complexity of learned edit operations and the percentage of input-output transformations the model can capture.

We propose to tackle text editing by decomposing it into two sub-problems: *tagging* and *insertion* (see Fig. 1). Our *tagger* is a Transformer-based network that implements a novel Pointing mechanism (Vinyals et al., 2015). It decides which source tokens to preserve and in which order they appear in the output, thus allowing for arbitrary word reordering.

Target words not present in the source are represented by the generic slot predictions to be in-filled by the *insertion* model. To benefit from self-supervised pre-training, we chose our *insertion* model to be fully compatible with the BERT architecture, such that we can easily re-use a publicly-available pre-trained checkpoint.

By decomposing text-editing tasks in this way we redistribute the complexity load of generating an output text between the two models: the source text already provides most of the building blocks required to reconstruct the target, which is handled by the *tagging* model. The missing pieces are then in-filled by the *insertion* model, whose job becomes much easier as most of the output text is already in place. Moreover, such a two-step approach is the key for being able to use completely non-autoregressive decoding for both models and still achieve competitive results compared to fully autoregressive approaches.

We evaluate FELIX on four distinct text generation tasks: Sentence Fusion, Text Simplification, Summarization, and Automatic Post-Editing for Machine Translation and compare it to recent text-editing and seq2seq approaches. Each task is unique in the editing operations required and the amount of training data available, which helps to better quantify the value of solutions we have integrated into FELIX[1].

2 Model description

FELIX decomposes the conditional probability of generating an output sequence $\mathbf{y}$ from an input $\mathbf{x}$ as follows:

$$p(\mathbf{y}|\mathbf{x}) \approx p_{\text{ins}}(\mathbf{y}|\mathbf{y}^m) p_{\text{tag}}(\mathbf{y}^t, \pi|\mathbf{x}) \qquad (1)$$

where the two terms correspond to the *tagging* and the *insertion* model. Term $\mathbf{y}^t$ corresponds to the output of the *tagging* model and consists of a sequence of tags assigned to each input token $\mathbf{x}$ and a permutation π, which reorders the input tokens. Term $\mathbf{y}^m$ denotes an intermediate sequence with masked spans and is fed into the *insertion* model. Given this factorization, both models can be trained independently.

2.1 Tagging Model

The tagging model is composed of three steps: (1) **Encoding**, the source sentence is first encoded using a 12-layer BERT-base model. (2) **Tagging**, a tagger is applied on top of the encoder and tags each source token. (3) **Pointing**, a pointer network, using attention applied to the encoders hidden states, re-orders the source tokens. FELIX is

[1]The code is publicly available at: https://felixmodel.page.link/code

	Src:	The	big	very	loud	cat		
Mask	$\mathbf{y}^t$**:**	KEEP	DEL	DEL	DEL$^{\texttt{INS\_2}}$	KEEP		
	$\mathbf{y}^m$**:**	The	[REPL] big very loud [/REPL]	MASK	MASK	cat		
	Pred:	The		noisy	large	cat		
Infill	$\mathbf{y}^t$**:**	KEEP	DEL	DEL	DEL$^{\texttt{INS}}$	KEEP		
	$\mathbf{y}^m$**:**	The	[REPL] big very loud [/REPL]	MASK	MASK	MASK	MASK	cat
	Pred:	The		noisy	large	PAD	PAD	cat

Figure 2: An example of two ways to model inputs to the *insertion* model: via token *masking* (Mask) or *infilling* (Infill). In the former case the *tagging* model predicts the number of masked tokens (INS_2), while in the latter it is delegated to the *insertion* model, which replaces the generic INS tag with a fixed length span (length 4), the *insertion* model then predicts a special PAD symbol to mark the end of the predicted span. Replacements are modeled by keeping the deleted spans between the [REPL] tags. For simplicity we do not show reordering.

trained to optimize both the tagging and pointing loss:

$$\mathcal{L} = \mathcal{L}_{pointing} + \lambda \mathcal{L}_{tagging} \quad (2)$$

where λ is a hyperparameter.

Tagging. The tag sequence $\mathbf{y}^t$ is constructed as follows: source tokens that must be copied are assigned the KEEP tag, tokens not present in the output are marked by the DELETE tag, token spans present in the output but missing from the input are modeled by the INSERT (INS) tag. This tag is then converted into masked token spans in-filled by the *insertion* model.

Tags are predicted by applying a single feed-forward layer f to the output of the encoder $\mathbf{h}^L$. We define: $p(y^t|x) = \prod_i p(\mathbf{y}_i^t|\mathbf{x})$, where i is the index of the source token. The model then is trained to minimize the cross-entropy loss. During decoding we use argmax to determine the tags, $\mathbf{y}_i^t = \text{argmax}(f(\mathbf{h}_i^L))$.

Pointing. FELIX explicitly models word reordering to allow for larger global edits, as well as smaller local changes, such as swapping nearby words, *John and Mary* $\rightarrow$ *Mary and John*. Without this word reordering step a vanilla editing model based on just tagging such as (Malmi et al., 2019; Dong et al., 2019), would first need to delete a span (*and Mary*) and then insert *Mary and* before *John*. FELIX is able to model this without the need for deletions or insertions. Given a sequence $\mathbf{x}$ and the predicted tags $\mathbf{y}^t$, the re-ordering model generates a permutation π so that from π and $\mathbf{y}^t$ we can reconstruct the insertion model input $\mathbf{y}^m$. Thus we have:

$$P(\mathbf{y}^m|\mathbf{x}) \approx \prod_i p(\pi(i)|\mathbf{x}, \mathbf{y}^t, i) p(\mathbf{y}_i^t|\mathbf{x}). \quad (3)$$

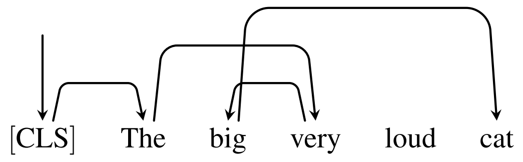

Figure 3: Pointing mechanism to transform *"the big very loud cat"* into *"the very big cat"*.

We highlight that each $\pi(i)$ is predicted independently, non auto-autoregressivly. The output of this model is a series of predicted pointers (source token $\rightarrow$ next target token). $\mathbf{y}^m$ can easily be constructed by daisy-chaining the pointers together, as seen in Fig. 3. As highlighted by this figure, FELIX's reordering process is similar to non-projective dependency parsing Dozat and Manning (2017), where head relationships are non-autoregressively predicted to form a tree. Similarly FELIX predicts next word relationship and instead forms a sequence.

Our implementation is based on a pointer network (Vinyals et al., 2015), where an attention mechanism points to the next token. Unlike previous approaches where a decoder state attends over an encoder sequence, our setup applies intra-attention, where source tokens attend to all other source tokens.

The input to the Pointer layer at position i is a combination of the encoder hidden state $\mathbf{h}_i^L$, the embedding of the predicted tag $e(\mathbf{y}_i^t)$ and the positional embedding $e(\mathbf{p}_i)$[2] as follows: $\mathbf{h}_i^{L+1} = f([\mathbf{h}_i^L; e(\mathbf{y}_i^t); e(\mathbf{p}_i)])$.

The pointer network attends over all hidden states, as such:

$$p(\pi(i)|\mathbf{h}_i^{L+1}) = \text{attention}(\mathbf{h}_i^{L+1}, \mathbf{h}_{\pi(i)}^{L+1}) \quad (4)$$

[2] Voita et al. (2019) have shown that models trained with masked language modeling objectives lose positional information, a property we consider important for reordering.

Attention between hidden states is calculated using a query-key network with a scaled dot-product:

$$\text{Attention}(\mathbf{Q},\mathbf{K}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}), \quad (5)$$

where $\mathbf{K}$ and $\mathbf{Q}$ are linear projections of $\mathbf{h}^{L+1}$ and d_k is the hidden dimension. We found the optional inclusion of an additional Transformer layer prior to the query projection increased the performance on movement-heavy datasets. The model is trained to minimize cross-entropy loss of the pointer network.

To realize the pointers, we use constrained beam search (Post and Vilar, 2018). Like Figure 3, we create the output by daisy chaining pointers, starting with [CLS], and finding the most probable pointer path, a token at a time. We ensure no loops are formed by preventing source token from being pointed to twice, and ensure that all source tokens not tagged with delete are pointed to[3]. We note that when using argmax, loops are only form in $< 3\%$ of the cases.

Dataset construction. When constructing the training dataset, there are many possible combinations of π and $\mathbf{y}^t$ which could produce $\mathbf{y}$. For instance, all source tokens could be replaced by MASK tokens. However, we wish to minimize the number of edits, particularly minimizing the amount of inserted tokens. To do so we greedily apply the following rules, iterating through the target tokens:

1. If the target token appears within the source sentence, point to it and tag it with keep. In the case, the target token appears multiple times in the source sentence, point to the nearest source token, as determined by the previously pointed to source token.

2. If a source token is already pointed to, then it cannot be pointed to again.[4]

3. If a target token does not appear within the source sentence, then it must be inserted. The previously pointed to source token is tagged with insert.

4. If a source token is not pointed to, then it is tagged with delete.

[3]We fix the beam size to 5. For a batch size of 32 and maximum sequence length of 128, beam search incurs an additional penalty of about 12ms when run on a Xeon CPU@3.7GHz.

[4]As each word has at most one out-going edge, having two incoming edges would form a loop.

2.2 Insertion Model

An input to the *insertion* model $\mathbf{y}^m$ contains a subset of the input tokens in the order determined by the *tagging* model, as well as masked token spans that it needs to in-fill.

To represent masked token spans we consider two options: *masking* and *infilling* (see Fig. 2). In the former case the *tagging* model predicts how many tokens need to be inserted by specializing the `INSERT` tag into `INS_k`, where `k` translates the span into `k` `MASK` tokens.

For the *infilling* case the *tagging* model predicts a generic `INS` tag, which signals the *insertion* model to infill it with a span of tokens of an arbitrary length. If we were to use an autoregressive *insertion* model, the natural way to model it would be to run the decoder until it decides to stop by producing a special *stop* symbol. Since by design we opted for using a non-autoregressive model, to represent variable-length insertions we use a `PAD` symbol to pad all insertions to a fixed-length[5] sequence of `MASK` tokens.

Note that we preserve the deleted span in the input to the *insertion* model by enclosing it between `[REPL]` and `[/REPL]` tags. Even though this introduces an undesired discrepancy between the pretraining and fine-tuning data that the *insertion* model observes, we found that making the model aware of the text it needs to replace significantly boosts the accuracy of the *insertion* model.

FELIX as Insertion Transformer. Another intuitive way to picture how FELIX works is to draw a connection with Insertion Transformer (Stern et al., 2019). In the latter, the decoder starts with a blank output text (canvas) and iteratively infills it by deciding which token and in which position should appear in the output. Multiple tokens can be inserted at a time thus achieving sub-linear decoding times. In contrast, FELIX trains a separate *tagger* model to pre-fill[6] the output canvas with the input tokens in a single step. As the second and final step FELIX does the insertion into the slots predicted by the *tagger*. This is equivalent to a single decoding step of the Insertion Transformer. Hence, FELIX requires significantly fewer (namely, two) decoding steps than Insertion Transformer, and through the *tagging/insertion* decomposition of the task it is straightforward

[5]A length of 8 was sufficient to represent over 99% of insertion spans.

[6]This corresponds to more than 80% of the output tokens.

to directly take advantage of existing pre-trained MLMs.

Similar to the *tagger*, our *insertion* model is also based on a 12-layer BERT-base and is initialized from a public pretrained checkpoint.

When using the *masking* approach, the *insertion* model is solving a masked language modeling task and, hence, we can directly take advantage of the BERT-style pretrained checkpoints. This is a considerable advantage, especially in the low-resource settings, as we do not waste training data on learning a language model component of the text-editing model[7]. With the task decomposition where *tagging* and *insertion* can be trained disjointly it essentially comes for free.

Switching from *masking* to *infilling* shifts the complexity of modeling the length of inserted token spans from the *tagging* model to the *insertion* model. Depending on the amount of training data available it provides interesting trade-offs between the accuracy of the *tagging* and *insertion* models. We compare these approaches in Sec. 3.4; for all other tasks we use the masking approach.

3 Experiments

We evaluate FELIX on four distinct text editing tasks: Sentence Fusion, Text Simplification, Summarization, and Automatic Post-Editing for Machine Translation. In addition to reporting previously published results for each task[8], we also compare to a recent text-editing approach LASERTAGGER (Malmi et al., 2019), which combines editing operations with a fixed vocabulary of additional phrases which can be inserted. We follow their setup and set the phrase vocabulary size to 500 and run all experiments using their most accurate autoregressive model. To decode a batch of 32 on a Nvidia Tesla P100, LASERTAGGER takes 1,300ms, FELIX takes 300ms and a a similarly sized seq2seq model takes 27,000ms (Malmi et al., 2019).

For all tasks we run an ablation study, examining the effect of an open vocabulary with no reordering (FELIXINSERT), and a fixed vocabulary[9] with reordering model (FELIXPOINT).

[7]We still fine-tune the insertion model to accommodate for the additional token spans between the `[REPL]` and `[/REPL]`

[8]To ensure fairness, unless otherwise stated, we recalculate all scores using our evaluation scripts.

[9]For simplicity we use the LASERTAGGER phrase vocabulary.

Task analysis. The chosen tasks cover a diverse set of edit operations and a wide range of dataset sizes. Table 1 provides dataset statistics including: the size, sentence length, and the translation error rate (TER) (Snover et al., 2006) between the source and target sentences. We use TER to highlight unique properties of each task. The summarization dataset is a deletion-heavy dataset, with the highest number of deletion edits and the largest reduction in sentence length. It contains moderate amounts of substitutions and a large number of shift edits, caused by sentence re-ordering. Both the simplification and post-editing datasets contain a large number of insertions and substitutions, while simplification contains a greater number of deletion edits. Post-editing, however, is a much larger dataset covering multiple languages. Sentence fusion has the lowest TER, indicating that obtaining the fused targets requires only a limited number of local edits. However, these edits require modeling the discourse relation between the two input sentences, since a common edit type is predicting the correct discourse connective (Geva et al., 2019). Additionally, within Table 2 we provide coverage statistics (the percentage of training instances for which an editing model can fully reconstruct the output) and MASK percentages (the percentage of output tokens which the insertion model must predict). As both FELIX and FELIXINSERT use an open vocabulary, they cover 100% of the data, whereas FELIXPOINT and LASERTAGGER often cover less than half. For every dataset FELIXPOINT covers a significantly higher percentage than LASERTAGGER, with the noticeable case being summarization, where there is a 3x increase in coverage. This can be explained by the high number of shift edits within summarization (Table 1), something FELIXPOINT is explicitly designed to model. We found that the difference in coverage between FELIXPOINT and LASERTAGGER correlates strongly (correlation 0.99, $p<0.001$) with the number of shift edits. Comparing MASK percentages, we see that FELIX always inserts ($\sim$50%) fewer MASKs than FELIXINSERT.

3.1 Summarization

Summarization is the task that requires systems to shorten texts in a meaning-preserving way.

Data. We use the dataset from (Toutanova et al., 2016), which contains 6,168 short input texts (one or two sentences) and one or more human-written

Dataset	Size	L_{src}	L_{tgt}	TER	Ins	Del	Sub	Shft
Post-editing	5M	18.10	17.74	24.97	04.24	06.25	**11.30**	02.69
Simplification	296K	22.61	21.65	26.02	**04.75**	08.97	09.90	02.41
Summarization	**26K**	**32.48**	22.16	**43.23**	00.29	**32.06**	09.34	**10.71**
Sentence fusion	4.5M	30.51	**30.04**	10.92	02.49	04.91	03.75	00.62

Table 1: Statistics across tasks: size of the dataset (Size), source length in tokens (L_{src}), target length in tokens (L_{tgt}), and TER scorse, including number of insertions (Ins), deletions (Del), substitutions (Sub), and shifts (Shft).

Dataset	Coverage % ↑		MASK % ↓	
	LASERTAGGER	FELIXPOINT	FELIXINSERT	FELIX
Postediting	35.10	**40.40**	42.39	**17.30**
Simplification	36.87	**42.27**	18.23	**13.85**
Summarization	16.71	**48.33**	15.92	**11.91**
Sentence fusion	85.39	**95.25**	14.69	**09.20**

Table 2: Coverage and MASK statistics. FELIXINSERT and FELIX have 100% coverage.

	SARI	ADD	DEL	KEEP	Rouge	BLEU
SEQ2SEQ$_{\text{BERT}}$	32.10				52.70	08.30
LASERTAGGER	40.23	06.10	54.48	60.12	81.36	35.05
FELIXPOINT	41.61*	06.80	58.67*	59.36	80.58	32.90
FELIXINSERT	41.99*	06.80	**61.65***	57.53	77.78	29.68
FELIX	**42.78***	**08.11***	57.62*	**62.62***	**83.48***	**35.85**

Table 3: Summarization results. All models copied the source less than 2% of the time. Models significantly different from LASERTAGGER are marked with * ($p < 0.01$). Significance tests were performed using a student t-test.

summaries, resulting in 26,000 total training pairs. The human experts were not restricted to just deleting words when generating a summary, but were allowed to also insert new words and reorder parts of the sentence.

Metrics. We report SARI (Xu et al., 2016), which computes the average F1 scores of the added, kept, and deleted n-grams, as well as breaking it down into each component `KEEP`, `DELETE`, and `ADD`, as we found the scores were uneven across these metrics. We also include ROUGE-L and BLEU-4, as these metrics are commonly used in the summarization literature.

Results. In Table 3 we compare against LASERTAGGER and SEQ2SEQ$_{\text{BERT}}$ from (Malmi et al., 2019), a seq2seq model initialized using BERT. The results show that FELIX achieves the highest SARI, ROUGE and BLEU scores. All ablated models achieve higher SARI scores than all other models.

3.2 Simplification

Sentence simplification is the problem of simplifying sentences such that they are easier to understand. Simplification can be both lexical, replacing or deleting complex words; or syntactic, replacing complex syntactic constructions.

Data. Training is performed on WikiLarge, (Zhang and Lapata, 2017a) a large simplification corpus which consists of a mixture of three Wikipedia simplification datasets collected by (Kauchak, 2013; Woodsend and Lapata, 2011; Zhu et al., 2010). The test set was created by Xu et al. (2016) and consists of 359 source sentences taken from Wikipedia, and then simplified using Amazon Mechanical Turkers to create eight references per source sentence.

Metrics. We report SARI, a readability metric FleschKincaid grade level (FKGL), and the percentage of unchanged source sentences (copy).

Results. In Table 4 we compare against: Three state-of-the-art SMT-based simplification systems: (1) PBMT-R (Wubben et al., 2012), a phrase-based machine translation model; (2) Hybrid (Narayan and Gardent, 2014), a model which performs sentence splitting and deletions and then simplifies with PBMT-R; (3) SBMT-SARI (Xu et al., 2016), a syntax-based translation model trained on PPDB and then tuned using SARI. Four neural seq2seq approaches: (1) DRESS (Zhang and Lapata, 2017b), an LSTM-based seq2seq trained with reinforcement learning; (2) DRESS-Ls, a variant of DRESS which has an additional lexical simplification component; (3) NTS (Nisioi et al., 2017), a seq2seq model; and (4) DMASS (Zhao et al., 2018), a transformer-based model enhanced with simplification rules from PPDB. And two neural editing models: (1) LASERTAGGER and (2) EditNTS (Dong et al., 2019), an autoregressive LSTM-based approach for text simplification, using KEEP/DELETE tags and open vocabulary predictions.

WikiLarge	SARI	ADD	DEL	KEEP	FKGL	Copy
SBMT-SARI	37.94	**05.60**	37.96	70.27	8.89	0.10
DMASS+DCSS	37.01	05.16	40.90	64.96	9.24	0.06
PBMT-R	35.92	05.44	32.07	70.26	10.16	0.11
HYBRID	28.75	01.38	**41.45**	43.42	**7.85**	**0.04**
NTS	33.97	03.57	30.02	68.31	9.63	0.11
DRESS	33.30	02.74	32.93	64.23	8.79	0.22
DRESS-LS	32.98	02.57	30.77	65.60	8.94	0.27
EDITNTS	34.94	03.23	32.37	69.22	9.42	0.12
LASERTAGGER	32.31	03.02	33.63	60.27	9.82	0.21
FELIXPOINT	34.37	02.35	34.80	65.97	9.47	0.18
FELIXINSERT	35.79	04.03	39.70	63.64	8.14	0.09
FELIX	**38.13**	03.55	40.45	**70.39**	8.98	0.08

Table 4: Sentence Simplification results on WikiLarge.

FELIX achieves the highest overall SARI score and the highest SARI-KEEP score. In addition, all ablated models achieve higher SARI scores than LASERTAGGER. While FELIXINSERT achieves a higher SARI score than EditNTS, FELIXPOINT does not; this can in part be explained by the large number of substitutions and insertions within this dataset, with FELIXPOINT achieving a low SARI-ADD score.

3.3 Post-Editing

Automatic Post-Editing (APE) is the task of automatically correcting common and repetitive errors found in machine translation (MT) outputs.

Data. APE approaches are trained on triples: the source sentence, the machine translation output, and the target translation. We experiment on the WMT17 EN-DE IT post-editing task[10], where the goal is to improve the output of an MT system that translates from English to German and is applied to documents from the IT domain. We follow the procedures introduced in (Junczys-Dowmunt and Grundkiewicz, 2016) and train our models using two synthetic corpora of 4M and 500K examples merged with a corpus of 11K real examples oversampled 10 times. The models that we study expect a single input string. To obtain this and to give the models a possibility to attend to the English source text, we append the source text to the German translation. Since the model input consists of two different languages, we use the multilingual Cased BERT checkpoint for FELIX and LASERTAGGER.

Metrics. We follow the evaluation procedure of WMT17 APE task and use TER as the primary metric and BLEU as a secondary metric.

[10]http://statmt.org/wmt17/ape-task.html

	TER ↓	BLEU ↑
COPY	24.48	62.49
TRANSFORMER	22.1	67.2
LASERTAGGER	24.29	63.83
LEVT	21.9	66.9
SOTA (Lee et al., 2019)	**18.13**	**71.80**
FELIXPOINT	22.51	65.61
FELIXINSERT	29.09	57.42
FELIX	21.87	66.74

Table 5: WMT17 En→De post-editing results.

Results. We consider the following baselines: COPY, which is a competitive baseline given that the required edits are typically very limited; LASERTAGGER (Malmi et al., 2019); LEVENSHTEIN TRANSFORMER (LEVT) (Gu et al., 2019), a partially autoregressive model that also employs *deletion* and *insertion* mechanisms; a standard TRANSFORMER evaluated by (Gu et al., 2019); and a state-of-the-art method by (Lee et al., 2019). Unlike the other methods, the last baseline is tailored specifically for the APE task by encoding the source separately and conditioning the MT output encoding on the source encoding (Lee et al., 2019).

Results are shown in Table 5. First, we can see that using a custom method (Lee et al., 2019) brings significant improvements over generic text transduction methods. Second, FELIX performs very competitively, yielding comparative results to LEVT which is a partially autoregressive model, and outperforming the other generic models in terms of TER. Third, FELIXINSERT performs considerably worse than FELIX and FELIXPOINT, suggesting that the pointing mechanism is important for the APE task. This observation is further supported by Table 2 which shows that without the pointing mechanism the average proportion of masked tokens in a target is 42.39% whereas with pointing it is only 17.30%. This suggests that, removing the pointing mechanism shifts the responsibility too heavily from the tagging model to the insertion model.

3.4 Sentence Fusion

Sentence Fusion is the problem of fusing independent sentences into a coherent output sentence(s).

Data. We use the balanced Wikipedia portion of the DiscoFuse dataset (Geva et al., 2019) and also study the effect of the training data size by creating four smaller subsets of DiscoFuse: 450,000 (10%), 45,000 (1%), 4,500 (0.1%) and 450 (0.01%) data

Model	Insertion		Oracle		SARI	Exact	10%	1%	0.1%	0.01%
	Mask	Infill	TAG	INS						
BERT2BERT					**89.52**	**63.90**	54.45	42.07	03.35	00.00
SEQ2SEQ$_{\text{BERT}}$					85.30	53.60	52.80	43.70	00.00	00.00
LASERTAGGER					85.45	53.80	47.31	38.46	25.74	12.32
FELIXPOINT					88.20	60.76	53.75	44.90	31.87	13.82
FELIXINSERT		●	●			82.91	77.25	71.49	57.94	36.61
		●		●		75.00	71.97	66.87	57.08	38.89
		●			88.44	60.80	52.82	46.09	34.11	15.34
	●		●			72.91	64.00	55.45	39.71	18.89
	●			●		88.86	84.11	81.76	75.88	61.68
	●				88.72	63.37	**56.67**	**48.85**	33.32	13.99
FELIX		●	●			70.32	71.78	64.28	51.20	28.42
		●		●		78.37	75.56	72.24	65.95	55.97
		●			87.69	58.32	55.11	48.84	**38.01**	**20.49**
	●		●			67.78	59.62	52.74	41.48	17.30
	●			●		87.52	86.45	83.13	79.79	67.60
	●				88.78	61.31	52.85	45.45	36.87	16.96

Table 6: Sentence Fusion results on DiscoFuse using the full and subsets 10%, 1%, 0.1% and 0.01% of the training set. We report three model variants: FELIXPOINT, FELIXINSERT and FELIX using either Mask or Infill insertion modes. Rows in gray background report scores assuming oracle *tagging* (TAG) or *insertion* (INS) predictions.

points.

Metrics. Following Geva et al. (2019), we report two metrics: *Exact score*, which is the percentage of exactly correctly predicted fusions, and SARI.

Results. Table 6 includes additional BERT-based seq2seq baselines: SEQ2SEQ$_{\text{BERT}}$ and BERT2BERT from (Rothe et al., 2020). For all FELIX variants we further break down the scores based on how the `INSERTION` is modelled: via token-masking (Mask) or Infilling (Infill). Additionally, to better understand the contribution of *tagging* and *insertion* models to the final accuracy, we report scores assuming oracle *insertion* and *tagging* predictions respectively (highlighted rows).

The results show that FELIX and its variants significantly outperform the baselines LASERTAGGER and SEQ2SEQ$_{\text{BERT}}$, across all data conditions. Under the 100% condition BERT2BERT achieves the highest SARI and Exact score, however for all other data conditions FELIX outperforms BERT2BERT. Both seq2seq models perform poorly with less than 4500 (0.1%) datapoints, whereas all editing models achieve relatively good performance.

When comparing FELIX variants we see on the full dataset FELIXINSERT outperforms FELIX, however we note that for FELIXINSERT we followed Malmi et al. (2019) and used an additional sentence re-ordering tag, a hand crafted feature tailored to DiscoFuse which swaps the sentence order. It was included in Malmi et al. (2019) and resulted in a significant (6% Exact score) increase. However, in the low resource setting, FELIX outperforms FELIXINSERT, suggesting that FELIX is more data efficient than FELIXINSERT.

Ablation. We first contrast the impact of the *insertion* model and the *tagging* model, noticing that for all models Infill achieves better tagging scores and worse insertion scores than Mask. Secondly, FELIX achieves worse tagging scores but better insertion scores than FELIXINSERT. This highlights the amount of pressure each model is doing, by making the tagging task harder, such as the inclusion of reordering, the insertion task becomes easier. Finally, the *insertion* models, even under very low data conditions, achieve impressive performance. This suggests that under low data conditions most pressure should be applied to the *insertion* model.

4 Related work

Seq2seq models (Sutskever et al., 2014) have been applied to many text generation tasks that can be cast as monolingual translation, but they suffer from well-known drawbacks (Wiseman et al., 2018): they require large amounts of training data, and their outputs are difficult to control. Whenever input and output sequences have a large overlap, it is reasonable to cast the problem as a text editing task, rather than full-fledged sequence-to-sequence generation. Ribeiro et al. (2018) argued that the general problem of string transduction can be reduced to sequence labeling. Their approach applied only to character deletion and insertion and was based on simple patterns. LaserTagger (Malmi et al., 2019) is a general approach that has been shown to perform well on a number of text editing tasks, but it has two limitations: it does not allow for arbitrary reordering of the input tokens; and insertions are restricted to a fixed phrase vocabulary that is derived from the training data. Similarly, EditNTS (Dong et al., 2019) and PIE (Awasthi et al., 2019) are two other text-editing models developed specifically for simplification and grammatical error correction, respectively.

Pointer networks have been previously proposed as a way to copy parts of the input in hybrid seq2seq models. Gulcehre et al. (2016) and Nallapati et al. (2016) trained a pointer network to specifically deal with out-of-vocabulary words or named entities. Chen and Bansal (2018) proposed a summarization model that first selects salient sentences and then rewrites them abstractively, using a pointer mechanism to directly copy some out-of-vocabulary words.

Previous approaches have proposed alternatives to autoregressive decoding (Gu et al., 2018; Lee et al., 2018; Chan et al., 2019; Wang and Cho, 2019). Instead of the left-to-right autoregressive decoding, Insertion Transformer (Stern et al., 2019) and BLM (Shen et al., 2020) generate the output sequence through insertion operations, whereas LEVT (Gu et al., 2019) additionally incorporates a deletion operation. These methods produce the output iteratively, while FELIX requires only two steps: *tagging* and *insertion*.

The differences between the proposed model, FELIX, its ablated variants, and a selection of related works is summarized in Table 7.

	Type	Non-autoregressive	Pretrained	Reordering	Open vocab
TRANSFORMER					✓
+ COPYING	seq2seq			✓	✓
T5			✓	(✓)	✓
LEVT		(✓)	✓		✓
PIE		✓	✓		✓
EDITNTS					✓
LASERTAGGER	Text edit	✓	✓		
FELIXINSERT		✓	✓		✓
FELIXPOINT		✓	✓	✓	
FELIX		✓	✓	✓	✓

Table 7: Model comparison along five dimensions: model *type*, whether the model: is *non-autoregressive* (LEVT is partially autoregressive), uses a *pretrained* checkpoint, uses a word *reordering* mechanism (T5 uses a reordering pretraining task but does not have a copying mechanism), able to generate any possible output (*Open vocab*).

5 Conclusions and Future Work

We have introduced FELIX, a novel approach to text editing, by decomposing the task into *tagging* and *insertion* which are trained independently. Such separation allows us to take maximal benefit from the already existing pretrained masked-LM models. FELIX works extremely well in low-resource settings and it is fully non-autoregressive which favors faster inference. Our empirical results demonstrate that it delivers highly competitive performance when compared to strong seq2seq baselines and other recent text editing approaches.

In the future we plan to investigate the following ideas: (i) how to effectively share representations between the *tagging* and *insertion* models using a single shared encoder, (ii) how to perform joint training of *insertion* and *tagging* models instead of training them separately, (iii) strategies for unsupervised pre-training of the *tagging* model. which appears to be the bottleneck in highly low-resource settings, and (iv) distillations recipes.

Acknowledgments

We thank Aleksandr Chuklin, Daniil Mirylenka, Ryan McDonald, and Sebastian Krause for useful discussions, running early experiments and paper suggestions.

References

Abhijeet Awasthi, Sunita Sarawagi, Rasna Goyal, Sabyasachi Ghosh, and Vihari Piratla. 2019. Parallel iterative edit models for local sequence transduction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4260–4270. Association for Computational Linguistics.

William Chan, Nikita Kitaev, Kelvin Guu, Mitchell Stern, and Jakob Uszkoreit. 2019. Kermit: Generative insertion-based modeling for sequences.

Yen-Chun Chen and Mohit Bansal. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 675–686. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Yue Dong, Zichao Li, Mehdi Rezagholizadeh, and Jackie Chi Kit Cheung. 2019. EditNTS: An neural programmer-interpreter model for sentence simplification through explicit editing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3393–3402. Association for Computational Linguistics.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Mor Geva, Eric Malmi, Idan Szpektor, and Jonathan Berant. 2019. DiscoFuse: A large-scale dataset for discourse-based sentence fusion. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3443–3455. Association for Computational Linguistics.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O.K. Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In *International Conference on Learning Representations*.

Jiatao Gu, Changhan Wang, and Junbo Zhao. 2019. Levenshtein transformer. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 11179–11189. Curran Associates, Inc.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 140–149. Association for Computational Linguistics (ACL), Association for Computational Linguistics.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2016. Log-linear combinations of monolingual and bilingual neural machine translation models for automatic post-editing. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 751–758, Berlin, Germany. Association for Computational Linguistics.

David Kauchak. 2013. Improving text simplification language modeling using unsimplified text data. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1537–1546. Association for Computational Linguistics.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1173–1182. Association for Computational Linguistics.

WonKee Lee, Junsu Park, Byung-Hyun Go, and Jong-Hyeok Lee. 2019. Transformer-based automatic post-editing with a context-aware encoding approach for multi-source inputs. *arXiv preprint arXiv:1908.05679*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pretraining for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Eric Malmi, Sebastian Krause, Sascha Rothe, Daniil Mirylenka, and Aliaksei Severyn. 2019. Encode, tag, realize: High-precision text editing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5054–5065. Association for Computational Linguistics.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar Gulçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290. Association for Computational Linguistics.

Shashi Narayan and Claire Gardent. 2014. Hybrid simplification using deep semantics and machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 435–445. Association for Computational Linguistics.

Sergiu Nisioi, Sanja Štajner, Simone Paolo Ponzetto, and Liviu P. Dinu. 2017. Exploring neural text simplification models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 85–91. Association for Computational Linguistics.

Matt Post and David Vilar. 2018. Fast lexically constrained decoding with dynamic beam allocation for neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1314–1324, New Orleans, Louisiana. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392. Association for Computational Linguistics.

Joana Ribeiro, Shashi Narayan, Shay B. Cohen, and Xavier Carreras. 2018. Local string transduction as sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1360–1371. Association for Computational Linguistics.

Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. 2020. Leveraging pre-trained checkpoints for sequence generation tasks. *Transactions of the Association for Computational Linguistics*, pages 264–280.

Tianxiao Shen, Victor Quach, Regina Barzilay, and Tommi Jaakkola. 2020. Blank language models.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of association for machine translation in the Americas*, volume 200.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. Mass: Masked sequence to sequence pre-training for language generation.

Mitchell Stern, William Chan, Jamie Kiros, and Jakob Uszkoreit. 2019. Insertion transformer: Flexible sequence generation via insertion operations. volume 97 of *Proceedings of Machine Learning Research*, pages 5976–5985, Long Beach, California, USA. PMLR.

I Sutskever, O Vinyals, and QV Le. 2014. Sequence to sequence learning with neural networks. *Advances in NIPS*.

Kristina Toutanova, Chris Brockett, Ke M. Tran, and Saleema Amershi. 2016. A dataset and evaluation metrics for abstractive compression of sentences and short paragraphs. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 340–350. Association for Computational Linguistics.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in neural information processing systems*, pages 2692–2700.

Elena Voita, Rico Sennrich, and Ivan Titov. 2019. The bottom-up evolution of representations in the transformer: A study with machine translation and language modeling objectives. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4396–4406. Association for Computational Linguistics.

Alex Wang and Kyunghyun Cho. 2019. BERT has a mouth, and it must speak: BERT as a Markov random field language model. In *Proceedings of the Workshop on Methods for Optimizing and Evaluating Neural Language Generation*, pages 30–36. Association for Computational Linguistics.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355. Association for Computational Linguistics.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2018. Learning neural templates for text generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3174–3187. Association for Computational Linguistics.

Kristian Woodsend and Mirella Lapata. 2011. Learning to simplify sentences with quasi-synchronous grammar and integer programming. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 409–420. Association for Computational Linguistics, Association for Computational Linguistics.

Sander Wubben, Antal van den Bosch, and Emiel Krahmer. 2012. Sentence simplification by monolingual machine translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1015–1024, Jeju Island, Korea. Association for Computational Linguistics.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Linguistics*, 4:401–415.

Xingxing Zhang and Mirella Lapata. 2017a. Sentence simplification with deep reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 584–594. Association for Computational Linguistics.

Xingxing Zhang and Mirella Lapata. 2017b. Sentence simplification with deep reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 584–594. Association for Computational Linguistics.

Sanqiang Zhao, Rui Meng, Daqing He, Andi Saptono, and Bambang Parmanto. 2018. Integrating transformer and paraphrase rules for sentence simplification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3164–3173. Association for Computational Linguistics.

Zhemin Zhu, Delphine Bernhard, and Iryna Gurevych. 2010. A monolingual tree-based translation model for sentence simplification. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 1353–1361. Association for Computational Linguistics, Coling 2010 Organizing Committee.