LEWIS: Levenshtein Editing for Unsupervised Text Style Transfer

Machel Reid[*]
The University of Tokyo
machelreid@weblab.t.u-tokyo.ac.jp

Victor Zhong
University of Washington
vzhong@cs.washington.edu

Abstract

Many types of text style transfer can be achieved with only small, precise edits (e.g. sentiment transfer from `I had a terrible time...` to `I had a great time...`). We propose a coarse-to-fine editor for style transfer that transforms text using Levenshtein edit operations (e.g. insert, replace, delete). Unlike prior single-span edit methods, our method concurrently edits multiple spans in the source text. To train without parallel style text pairs (e.g. pairs of +/- sentiment statements), we propose an unsupervised data synthesis procedure. We first convert text to style-agnostic templates using style classifier attention (e.g. `I had a SLOT time...`), then fill in slots in these templates using fine-tuned pretrained language models. Our method outperforms existing generation and editing style transfer methods on sentiment (YELP, AMAZON) and politeness (POLITE) transfer. In particular, multi-span editing achieves higher performance and more diverse output than single-span editing. Moreover, compared to previous methods on unsupervised data synthesis, our method results in higher quality parallel style pairs and improves model performance.[1]

1 Introduction

In text style transfer, a model changes the style of a source text (e.g. sentiment, politeness) into a target style, while otherwise changing as little as possible about the input. Many types of style transfer can be performed with only small, precise edits instead of generation from scratch. Consider the task of transforming a negative sentiment sentence such as `the worst ribs I've ever had!` to a positive sentence such as `probably the best ribs`

---

[*]Corresponding author
[1]Code and data can be found at https://github.com/machelreid/lewis

ever!. Here, we need only invert the negative sentiment phrase around `worst` — the references to `ribs` should be left as-is. Recent and concurrent work on text style transfer propose single-span editing (for insertion and replacement) (Wu et al., 2019; Malmi et al., 2020) as an alternative to generating the target text from scratch (Prabhumoye et al., 2018; He et al., 2020b; John et al., 2019; Shen et al., 2017; Fu et al., 2018).

We introduce a more flexible and powerful multi-span editing method that identifies multiple style-specific components of the text and concurrently edits them into the target style. Given a source text, we first predict the sequence of coarse-grain Levenshtein edit types (e.g. insert, replace, delete) that transform the source text to the target text, then fill insertion and replacement edits using a generator. In the previous example, the operations correspond to inserting the word `probably` before `the`, replacing `worst` with `best`, and removing the words `I've` and `had`. This example is illustrated in detail in Figure 1.

Learning to edit requires supervised source-target text pairs. How do we learn high-quality editors when no such supervised parallel data exists? Given a style text, we synthesize its pair by identifying style-specific content and replacing it with samples from style-specific masked language-models. In our sentiment transfer example, the style-specific content of the sentence `I had a great time at the theatre` is `had a great time`. We can replace this phrase by `saw a fantastic movie today` to synthesize an alternative positive-sentiment sentence, or by `got ripped off today` to synthesize a negative-sentiment sentence. Figure 3 illustrates this example in detail.

We evaluate our editing and synthesis framework, which we call LEWIS (Levenshtein editing with unsupervised synthesis), on three style transfer tasks in sentiment (YELP, AMAZON) and polite-

3932

Source text x		the	worst	ribs	I've	ever	had	!
Coarse edit types c	`<ins>`	`<keep>`	`<repl>`	`<keep>`	`<del>`	`<keep>`	`<del>`	`<keep>`
Generator input	`<MASK>`	the	`<MASK>`	ribs		ever		!
Target text y	probably	the	best	ribs		ever		!

Coarse-grain edit type tagger
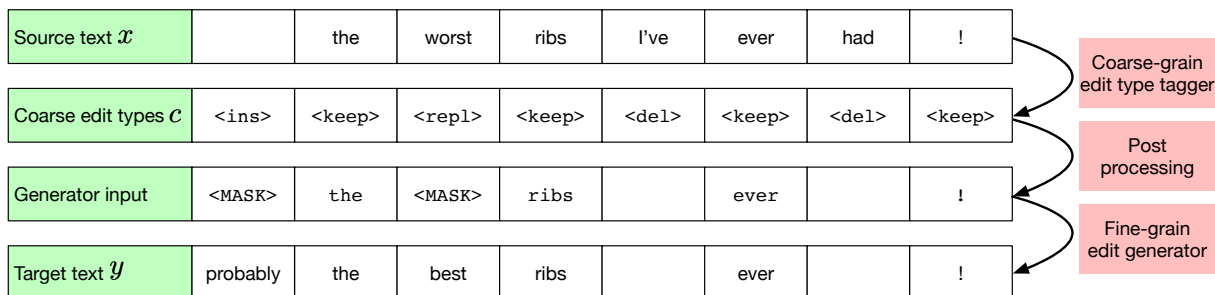Post processing
Fine-grain edit generator

Figure 1: Coarse-to-fine Levenshtein editor. Given the source text, the two-step editor first generates coarse edit types via a tagger. A subsequent generator fills in insertions and replacements while taking into account the source text and the edit types.

ness (POLITE) transfer, and achieve state-of-the-art results in terms of retention of style-agnostic content, similarity to the annotated target text, and transfer accuracy. LEWIS significantly outperforms prior state-of-the-art methods by 2.6-13.5% accuracy depending on the task. In further analyses, we show that (1) compared to concurrent work on editing for style transfer, our editor achieves 33.3% higher accuracy when trained on the same data; (2) compared to a competitive BART (Lewis et al., 2020) pure generation baseline, our editor achieves 5.8% higher accuracy when trained on the same data; (3) compared to concurrent work on unsupervised synthesis of style transfer data, our synthesis procedure improves performance by 9.5 BLEU when used to train the same model. Our experiments show that our editor significantly outperforms both pure generation and editing prior methods, that our editor yields more diverse text transfer and that training on our synthesized data improves performance more than prior synthesis methods.

2 LEWIS

LEWIS consists of coarse-to-fine editing and data synthesis. The editing component, shown in Figure 1, performs local, precise edits of style-specific content of the source text to produce the target text. The data synthesis component, shown in Figure 3, produces supervised source-target text pairs, which do not exist in naturally, to train the editor. To apply our method to transfer text from a source style to a target style, we first train style-specific masked language models, with which we synthesize source-target text pairs. We then compute Levenshtein operations for these source-target text pairs and train the coarse-to-fine editor to reproduce these operations. The full LEWIS is shown in Figure 2. For

ease of exposition, we first describe the editor, then describe how to synthesize parallel data to train the editor.

2.1 Style transfer via Levenshtein editing

We propose a coarse-to-fine editor that first predicts coarse-grain Levenshtein edit types (Levenshtein, 1966), then fills in fine-grain edits with a generator. Figure 1 illustrates the editor.

Suppose we are to transfer text from a source style into a target style. Let x denote the source text, which we would like to edit into the target text y. In the example shown in Figure 1, we transform the source text `the worst ribs I've ever had!` into `probably the best ribs ever!` Our approach has two parts: The source text is first tagged with a sequence of coarse-grain Levenshtein transition types c that transform x into y. A generator then fills in phrases for insertion and replacement operations. The set of coarse Levenshtein transition types are `insert`, `keep`, `replace`, and `delete`. In the running example, the sequence of operations are to insert before `the`, replace `worst`, and delete `I've` and `had`.

First, we train a RoBERTa-tagger (Liu et al., 2019) to generate these coarse edit types, which produces coarse edit types for each token in the source text. To accommodate the insertion operation, we produce two tags for each token. The first tag is a binary indicator of whether an additional phrase should be inserted before this token. The second tag is the non-insertion operation to take for this token. In the previous example, for instance, the word `the` triggers both insertion and keep operations.

$$c = \text{RoBERTa}_c(x; \Phi_c) \qquad (1)$$

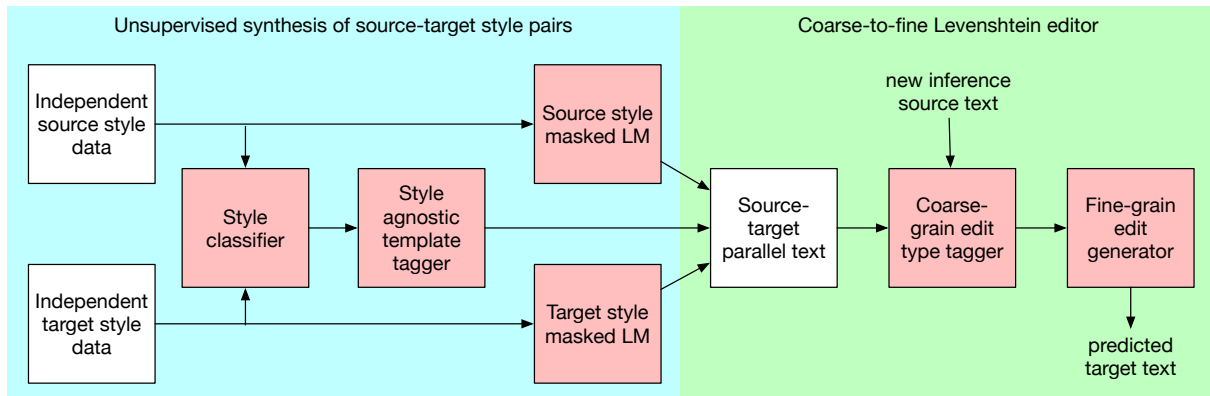Next, we train a fine-grain edit generator to produce the target text. Unlike the coarse-grain edit type

Figure 2: LEWIS consists of two components. Given source-target style text pairs, a coarse-to-fine Levenshtein editor (yellow) first identifies coarse-grain Levenshtein edit types to perform for each token in the source text (e.g. insert, replace, delete), then fills in the final edits with a fine-grain generator to produce the target text. In most applications, supervised source-target style text pairs rarely exist. To resolve this lack of annotated data, we perform unsupervised synthesis of source-target style pairs (blue) by first learning to produce style-agnostic templates given arbitrary style text. Next, we fill in slots in the template by sampling from style-specific masked language-models. In this figure, source and intermediate data are shown in white while model components are shown in red.

generator which only observes the source text, the fine-grain edit generator observes both the original source text and the source text with the coarse-grain edit types applied x_c. We use the edit types produced by the Levenshtein algorithm during training and the edit types predicted by the RoBERTa-tagger during inference.

$$y = \text{BART}_y(x, x_c; \Phi_{\text{fn}}) \quad (2)$$

Our generator is a BART-based (Lewis et al., 2020) masked sequence-to-sequence model. The input to BART is the concatenation of the original source text x and the source text with the coarse-grain edit types applied x_c. The generator is trained to fill in phrases for coarse-grain edit types `<ins>` and `<repl>`. In the example, BART_y is given the input text `the worst ribs I've ever had! SEP <MASK> the <MASK> ribs ever !` and respectively fills in the two `<MASK>`s with `probably` and `best`.

2.2 Unsupervised synthesis of source-target style pairs

Training an editor requires large quantities of source-target text pairs. While there exists an abundant amount of style-specific data, parallel source-target pairs are difficult to collect and annotate. How do we train editing style transfer methods when no such data exists? We hypothesize that pre-trained masked language-models, when carefully constrained to generate only style-specific content,

can provide high-quality source-target pairs for style transfer.

Our synthesis procedure, shown in Figure 3, is two-fold. First, given a text s from either style, we identify a style-agnostic template t, in which style-specific content are replaced with slots. For instance, for the style text `I had a great time at the theatre`, the style-agnostic template is `I SLOT at the theatre`. To identify style-specific content, we train a RoBERTa-based style classifier that differentiates between text from each style. Vaswani et al. (2017) and Hoover et al. (2020) show that heavily attended-tokens correlate strongly with tokens that are indicative of the target class. We observe similar results when inspecting the attention matrices computed by the 12-layer Transfomer for the sentiment classification task. Namely, the penultimate layer's attention weights correlate strongly with words humans identify as strongly indicative of positive vs. negative sentiment. Hence, we define style-specific content as tokens that have higher-than-average attention weights in the classifier.

Consider the multi-head attention matrix A in the penultimate Transformer layer, where A_{ij} represents the attention weight of the jth attention head on the i's token, normalized across all tokens. First, we max-pool A_i over all attention heads to form a_i. Conceptually, a_i represents the maximum extent to which the ith word was attended to by any
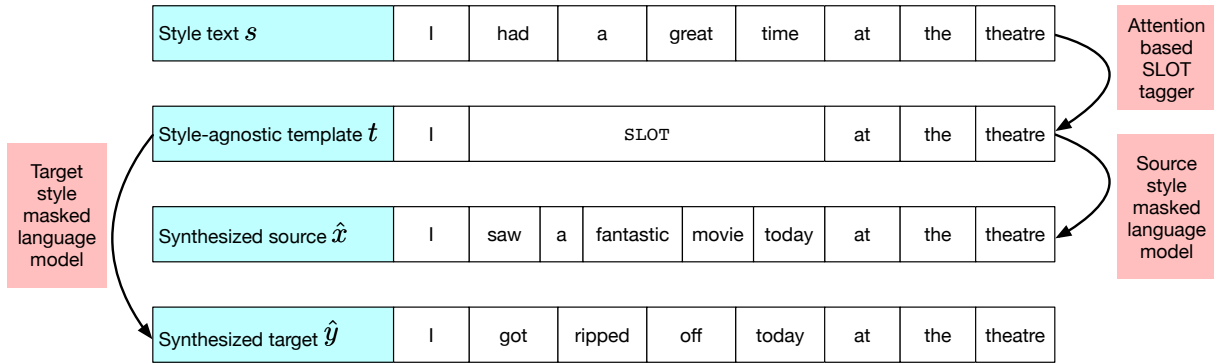
Figure 3: Unsupervised synthesis of source-target style pairs. We first train an attentive style classifier, whose attention weights we use to identify style-specific content. Next, we remove style-specific content with slots to form a style-agnostic template. This template is finally filled using style-specific masked language-models for each style to synthesize parallel style text pairs.

attention head.

$$a_i = \max_j A_{ij} \qquad (3)$$

Let N denote the sequence length. We compute the average attention weight $\tilde{a}$ as

$$\tilde{a} = \frac{\sum_i^N a_i}{N} \qquad (4)$$

To modify the style text s into the style-agnostic template t, we keep tokens that have above-average attention weight.

$$t_i = \begin{cases} \text{SLOT}, & \text{if } a_i \geq \tilde{a} \\ s_i, & \text{if } a_i < \tilde{a} \end{cases} \qquad (5)$$

We merge consecutive SLOT tokens in t. In the running example, for the style text I had a great time at the theatre, the tagger generates I SLOT SLOT SLOT SLOT at the theatre, which after merging becomes I SLOT at the theatre.

We then fine-tune style-specific masked language-models BART_x and BART_y to fill in slots in the template and recover the style-specific text. During training, phrases in the input sentence are randomly discarded and the model is trained to fill the phrases back in (Lewis et al., 2020). Having trained style-specific masked language-models for both the source and target styles, we use both models to generate source and target filled-in text given style-agnostic templates.

$$\hat{x} = \text{BART}_x(t; \Theta_x) \qquad (6)$$
$$\hat{y} = \text{BART}_y(t; \Theta_y) \qquad (7)$$

In our running example, sampling with the positive language model yields the sentence I saw a fantastic movie today at the theatre, while sampling with the negative language model yields the sentence I got ripped off today at the theatre.

The last step we perform is a filtering step using the classifier. For synthesized examples in style k, we keep examples for which the style classifier predicts k. In other words, we keep only examples where the language models and the classifier agree. We find that this improves data quality and editor performance. We use the collection of synthesized source and target text pairs $\hat{x}, \hat{y}$ to train the editor.

3 Experimental Setup

We focus on two types of text style transfer: (1) **Sentiment transfer**, in which we transform a positive sentiment sentence to a corresponding negative sentiment sentence or vice-versa without changing the core content (i.e. attributes of the sentence not concerned with sentiment) (2) **Politeness transfer**, in which we transform the tone of a sentence from impolite to polite.

DATASET	Attributes	Train	Valid	Test
YELP	Positive	270K	2000	500
	Negative	180K	2000	500
AMAZON	Positive	277K	985	500
	Negative	278K	1015	500
POLITE	Polite	219K	26K	800
	Impolite	198K	24K	—

Table 1: Dataset statistics for style transfer tasks. The politeness corpus does not have parallel evaluation data and only evaluates on transfer from impolite to polite.

We make use of three datasets: YELP (Shen et al., 2017) consists of 450K sentences from busi-

Model	Acc	SBLEU	BLEU	SBERT	BERT
Baselines					
Input Copy	1.5	100.0	24.8	100.0	53.74
Reference	81.6	25.3	100.0	53.7	100.0
Generation methods					
Delete and Retrieve (Li et al., 2018)	88.6	36.8	12.2	48.5	33.3
Tag and Generate (Madaan et al., 2020)	86.2	47.1	19.8	57.9	37.2
DeepLatentSeq (He et al., 2020b)	83.8	48.4	18.7	57.9	36.0
Editing methods					
Masker (Malmi et al., 2020)	40.9†	—	14.5	—	—
LaserTagger (Malmi et al., 2019) + Masker data	49.6†	—	15.3	—	—
LaserTagger + our data	59.8	**71.8**	24.8	**81.3**	**51.6**
LEWIS	**93.1**	58.5	24.0	72.2	50.0

Table 2: Results on YELP. Results with † are taken from the classifier trained in Malmi et al. (2020) because the outputs for these models are not released.

Model	Accuracy	SBLEU	BLEU	SBERT	BERT
Input Copy	13.1	100.0	48.7	100.0	63.0
Delete and Retrieve (Li et al., 2018)	51.2	57.1	29.9	66.9	46.2
Tag and Generate (Madaan et al., 2020)	60.8	**68.7**	**34.8**	69.5	48.2
LEWIS	**74.3**	65.6	32.9	**75.2**	**52.2**

Table 3: Results on AMAZON

ness reviews for training and 1000 sentences released by Li et al. (2018) for testing, AMAZON (He and McAuley, 2016) consists of 540K sentences from product reviews for training and 1000 sentences for testing, and POLITE (Madaan et al., 2020), produced by filtering through the Enron Email corpus, consisting of 420K sentences for training and 800 sentences for testing. We list dataset statistics in Table 1.

3.1 Training Setup

We implement our models using `fairseq`[2] (Ott et al., 2019) and HuggingFace[3] (Wolf et al., 2020) — both based on the PyTorch library (Paszke et al., 2019). For BART-based generation models, we initialize with BART-base (Lewis et al., 2020), and train using a batch size of 65K tokens for 30000 iterations. We use a linear warmup schedule, reaching the peak of 3×10^{-5} at 5000 iterations, and then proceed to decay the learning rate with a poly-

nomial decay schedule. For regularization, we use a dropout value of 0.3 and a weight decay value of 0.1. We optimize using Adam, with hyperparameters $\beta_1 = 0.9, \beta_2 = 0.98$ and cross entropy loss. For RoBERTa-based taggers and classifiers, we initialize with RoBERTa-base (Liu et al., 2019), and train using a batch size of 256 for 5000 iterations. We optimize using Adam, warm up the learning rate to 1×10^{-6} and then decay with a cosine schedule. We train all models using mixed precision (Micikevicius et al., 2018) for faster training. Similar to prior work (Wu et al., 2019; Malmi et al., 2020), we decode using a beam width of 5 and rerank outputs produced by beam search using the likelihood of the classifier trained in Section 2.2.

3.2 Comparison with existing methods

We compare LEWIS to five prior methods: Delete, Retrieve, Generate (Li et al., 2018), a retrieval method that finds text from the target domain corpus whose style-agnostic form is similar to that of the source text; Tag and Generate (Madaan et al.,

---

[2]https://github.com/pytorch/fairseq
[3]https://github.com/huggingface/transformers

Model	Accuracy	SBLEU	SBERT
Delete, Retrieve, Generate (Li et al., 2018)	—	11.8	—
Tag and Generate (Madaan et al., 2020)	84.8	70.4	71.6
LEWIS	**87.4**	**75.3**	**81.4**

Table 4: Results on POLITE

Model	Acc	SBLEU	BLEU	SBERT	BERT
LM fill	90.3	42.9	17.4	58.9	41.6
Seq2Seq	87.3	50.0	19.3	68.5	**50.0**
LEWIS w/o filtering	91.2	50.1	23.3	69.8	48.0
LEWIS	**93.1**	**58.5**	**24.0**	**72.2**	**50.0**

Table 5: Ablation on YELP. "LM fill" is the ablation experiment in which we convert the source style text to a style-agnostic template and directly use the target style language model to synthesize a target style text (e.g. the editor is not used). "Seq2Seq" is a pretrained BART model that is fine-tuned on the synthesized data (e.g. a from-scratch generation model trained on the same data as the editor).

2020), a generation method that conditionally generates target text from style-agnostic source text; DeepLatentSeq (He et al., 2020b), an unsupervised machine translation-based approach where generators in each domain are regularized by a language model-based latent prior. Finally, we compare to previous editing approaches proposed by Malmi et al. (2019, 2020) where a single span in the source text two domain-specific language models disagree on is replaced.

3.3 Evaluation

Automatic Evaluation We use five evaluation metrics: BLEU (Papineni et al., 2002) measured against the reference (denoted as **BLEU**) to evaluate lexical overlap with human annotation; Self-BLEU measured against the source to measure content preservation (denoted as **SBLEU**); BERTScore and Self-BERTScore (Zhang et al., 2020) measured against the reference and the source (denoted as **BERT** and **SBERT** respectively); and accuracy measured against an external classifier (denoted as **Accuracy**) to measure how well the style was transferred.

		BLEU	BERT
Ref	great place , great food !	—	—
Hyp 1	pathetic place , great food !	**76.0**	71.5
Hyp 2	amazing place , awesome food !	0.0	**78.6**

Table 6: Example comparing BERTScore vs BLEU. **Ref** denotes the reference sentence and **Hyp 1** and **Hyp 2** represent two example hypotheses.

While measuring BLEU, Self-BLEU, and accuracy are standard for this task, we propose additionally using BERTScore due to its higher correlation with human judgments (Zhang et al., 2020). Compared to BLEU and Self-BLEU which are n-gram based, BERTScore is measured using token-wise cosine similarity between representations produced by BERT (Devlin et al., 2019).

Given this, the usage of BERTScore addresses the potential issue of accurately transferred sentences being scored poorly due to its low n-gram overlap. Table 6 shows an example of this where the style is accurately transferred but is scored poorly by BLEU as a result of low n-gram overlap.

Furthermore, following Malmi et al. (2020) who use a BERT-based classifier to score their outputs, we train a classifier initialized with RoBERTa-base (Liu et al., 2019). This model correctly classifies 98.2% of the YELP classification test set by Shen et al. (2017). Its accuracy is used to evaluate the output of style transfer models.

Human Evaluation We perform a robust human evaluation on all datasets, asking crowdworkers to rate 300 examples from Yelp (150 positive, 150 negative), 200 examples from Amazon (100 positive,100 negative) and 100 from Politeness. Five annotators rate each pair from 1 (strongly disagree) to 5 (strongly agree) in terms of fluency, content preservation (CP) and style transfer. We compare with our strongest baseline Tag and Generate (Madaan et al., 2020).

Dataset	Model	Fluency	CP	Style
YELP	TG	3.84±1.01	3.63±0.93	3.67±1.02
	LEWIS	**3.94**±0.99	**3.76**±0.88	**3.72**±0.98
AMAZON	TG	3.60±1.01	3.48±0.93	3.37±1.02
	LEWIS	**3.65**±0.88	**3.50**±0.88	3.37±0.90
POLITE	TG	3.83±0.84	3.76±0.90	3.48±1.04
	LEWIS	**3.93**±0.78	**3.87**±0.83	**3.63**±0.98

Table 7: Human evaluation results comparing LEWIS and Tag and Generate (TG)

	Mean	Std
# merged edit ops	1.57	0.78
# source toks	10.74	2.73
# style-agnostic template toks	10.42	2.86
# edit output toks	11.24	3.16

Table 8: Coarse-to-fine editor statistics on YELP, after merging consecutive edit operations of the same type, so that the number of operations denote spans as opposed to tokens (e.g. delete, replace).

4   Results

Performance of LEWIS compared to other methods on YELP, AMAZON, and POLITE are respectively shown in Tables 2, 3, and 4, with human evaluation shown in Table 7. LEWIS outperforms prior methods on all datasets in terms of accuracy, BLEU, and BERTScore: LEWIS achieves more successful transfers (2.6-13.5% accuracy depending on task), has higher overlap with human annotations (4-14.4 BERTScore), and retains more source content (5.7-14.3 Self-BERTScore). Human evaluation ($p < 0.01$ for YELP and POLITE using pairwise bootstrap sampling (Koehn, 2004)) shows that LEWIS outperforms Tag and Generate on fluency, content preservation and style across datasets. These results indicate that LEWIS is an effective method for style transfer. On the AMAZON dataset — which is noisier than the YELP dataset — LEWIS underperforms Tag and Generate when evaluating using BLEU, however when evaluating using BERTScore LEWIS outperforms the latter. When we inspect the output of LEWIS, we find that it generates more diverse output as shown in Figure 4.

One reason that LEWIS generates more diverse output is that unlike previous and concurrent editing work that use single-span replacement (Malmi et al., 2019, 2020), our method concurrently edits multiple spans with a larger set of operations. This

is inherently supported by the editor (Figure 5) as well as encouraged during unsupervised data synthesis (Figure 4). Table 8 shows that a large number of examples do require multiple edits, and that the coarse-to-fine editor indeed performs multiple edit operations on average.

In addition to comparing end-to-end systems, we also compare LEWIS to concurrent editing and synthesis methods by Malmi et al. (2019, 2020). Table 2 shows that training the same model (LaserTagger) on our data improves and BLEU by 9.5 (the accuracy difference is not directly comparable since Malmi et al. (2020) used a BERT classifier and did not release model output). This suggests that our data synthesis procedure produces higher quality data than Malmi et al. (2020). Furthermore, because LaserTagger only performs single-span edits (when performing insertion/replacement), it often fails to transfer the style of the text. This also accounts for its high BLEU and BERTScore but low accuracy, as we show that a model that simply copies the input also achieves high BLEU and BERTScore but low accuracy. Replacing LaserTagger with our coarse-to-fine Levenshtein editor results in a sizable 33% gain in accuracy. In Table 1 of the Appendix, we show example outputs of these models for comparison. Finally, we ablate LEWIS to investigate how the different components of LEWIS affect performance.

Editing outperforms pure generation   We replace the coarse-to-fine editor with a sequence-to-sequence BART model, which we also train with synthesized data. This is a strong baseline that outperforms prior pure generation work on style transfer, as shown in Table 3. Nevertheless, Table 5 shows that LEWIS outperforms this baseline on all metrics. This confirms our hypothesis that editing is a more effective means of style transfer compared to pure generation.

Training on synthesized data improves performance.   Instead of training an editor using synthesized data, given a source text during inference, we convert it to a style-agnostic template and immediately fill it using the target language model. Table 5 shows that the resulting model underperforms both the sequence-to-sequence BART and the coarse-to-fine editor on all metrics. This result may be surprising, in that one expects the performance of a model trained on data synthesized by language models to be at-most on par with the per-

Orig	it certainly delivered .	i love the fresh , right out of the oven bread too .	moreover , they found ways to help save on the expense .
Template	`it SLOT delivered .`	`i SLOT the fresh, SLOT out of the oven SLOT.`	`SLOTover, SLOT ways to SLOT save on the expense.`
Synthesized source	it was well delivered. it was absolutely delivered.	i love the fresh , crisp bread out of the oven as well. i love the fresh , hot bread out of the oven as well.	overall , great place with ways to save on the expense. overall , good ways to help save on the expense.
Synthesized target	it was not delivered. it was never delivered.	i ordered the fresh , it came out of the oven cold . i ordered the fresh , it came out of the oven and was cold.	overall , there are better ways to try and save on the expense. overall , there are better ways to save on the expense .

Figure 4: Examples of synthesized parallel text on the YELP dataset.

Source	we will not be coming back .	the food 's ok , the service is among the worst i have encountered .	i said it was disgusting to even serve this to diners .
Coarse edit type tagger output	K K R K K R	K K D R R K K K K K R K K K K	K R K K R K K K K K K K
Fine generator input	we will <mask> be coming back <mask>	the food <mask> , the service is among the <mask> i have encountered .	i <mask> it was <mask> to even serve this to diners .
Fine generator output	we will definitely be coming back again !	the food is great , the service is among the best i have encountered.	i thought it was very nice to even serve this to diners.

Figure 5: Examples of coarse-to-fine editor output on the YELP dataset. We abbreviate the edit operation with K for `<keep>`, D for `<del>`, and R for `<repl>`. Unlike previous and concurrent edit methods, we concurrently edit multiple spans in the text.

formance of the language model. In this case, we observe that training on the synthesized data actually improves over just using the language models. We hypothesize that this gain is due to the editor learning correlations between the source language model and the target language model, namely how to precisely transform the output of the source language model to the output of the target language model. The gains we observe here may be related to gains from training on back-translated or pseudo-parallel data (Sennrich et al., 2016; Edunov et al., 2018; He et al., 2020a). More research is needed to investigate the problem conditions under which such gains occur.

Filtering improves performance. Here, we forgo the filtering step, which removes ≈ 20% of the synthesized data on YELP. Table 5 shows that filtering improves the quality of the synthesized data and leads to consistent gains.

5 Related Work

Text style transfer Previous work on style transfer can largely be divided into two categories: (1) learning a latent space with controllable attributes

such as those found in Shen et al. (2017); John et al. (2019) or (2) using unsupervised generative approaches from retrieval (Li et al., 2018), tagging using style phrases (Madaan et al., 2020), to back-translation and unsupervised machine translation techniques (Prabhumoye et al., 2018; Lample et al., 2019; He et al., 2020b).

Editing for style transfer Our work is closest to Madaan et al. (2020) and Malmi et al. (2020). Madaan et al. (2020) use a tagger to mark style phrases in the source text, then generates the target text conditioned on the tagged source text. In contrast, we do not fully generate target text and only perform small, precise edits. In concurrent work to ours, Malmi et al. (2020) train a BERT language model on each style and edits a span where the models' likelihoods disagree the most. In contrast, instead of performing single-span replacement, our editor concurrently edits multiple spans in the text, and supports a wider set of operations than replacement. We showed that this results in more effective and more diverse style transfer. This coarse-to-fine transformations of text, in which the input context is progressively refined, has also led to improve-

ments in syntactic parsing (Charniak and Johnson, 2005), semantic parsing (Dong and Lapata, 2018), and NER (Choi et al., 2018).

Unsupervised data synthesis for style transfer Malmi et al. (2020) also generate synthetic data with which to train an editing model from Malmi et al. (2019). Our synthesis differs from Malmi et al. (2020) in how slots for generation are chosen. In their work, the highest disagreeing span is chosen for rewriting. In our work, multiple spans with words whose attention weights that exceed the average are chosen for rewriting, which allows for more flexible and diverse samples. In turn, training on our synthesized data improves the performance and diversity of the style transfer model.

6   Conclusion

We proposed LEWIS, a coarse-to-fine editor for style transfer that transforms text using Levenshtein edit operations. Unlike prior edit methods, our methods concurrently performs multi-span edits. To train this editor, we proposed an unsupervised data synthesis procedure that converts text to style-agnostic templates using style classifier attention, then fills in slots in these templates using fine-tuned pretrained language models. LEWIS outperformed existing generation and editing style transfer methods on sentiment and politeness transfer. In addition, the proposed data-synthesis procedure increased transfer performance. Given the same synthesized data, our editor outperformed prior pure generation and editing methods. In future work, we will study the application of LEWIS to general sequence to sequence problems.

Ethical Considerations

This work has impact in the field of controlled text generation, and as with much of language technology has the potential to be both used for good and used maliciously. Our work learns to generate synthetic data in an unsupervised way, and is based on a pre-trained model (BART), which is likely to capture and amplify biases found in the data. Furthermore, as with all text-style transfer models, our model is amenable to malicious use, including impersonation and mass generation of faked opposing opinion, for example, negative and positive product reviews or political statements. We refer readers to Section 8 of Hu et al. (2020) for more on this topic. Given this, further study is needed to see how text-style transfer models are affected by the bias contained in the corpora they are trained on and other aspects to avoid undesired behaviors such as those listed in the cases above.

References

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 173–180, Ann Arbor, Michigan. Association for Computational Linguistics.

Eunsol Choi, Omer Levy, Yejin Choi, and Luke Zettlemoyer. 2018. Ultra-fine entity typing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 87–96, Melbourne, Australia. 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, Minneapolis, Minnesota. Association for Computational Linguistics.

Li Dong and Mirella Lapata. 2018. Coarse-to-fine decoding for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 731–742, Melbourne, Australia. Association for Computational Linguistics.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 489–500, Brussels, Belgium. Association for Computational Linguistics.

Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. 2018. Style transfer in text: Exploration and evaluation. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence,*

(AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018, pages 663–670. AAAI Press.

Junxian He, Jiatao Gu, Jiajun Shen, and Marc'Aurelio Ranzato. 2020a. Revisiting self-training for neural sequence generation. In 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net.

Junxian He, Xinyi Wang, Graham Neubig, and Taylor Berg-Kirkpatrick. 2020b. A probabilistic formulation of unsupervised text style transfer. In 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net.

Ruining He and Julian J. McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11 - 15, 2016, pages 507–517. ACM.

Benjamin Hoover, Hendrik Strobelt, and Sebastian Gehrmann. 2020. exBERT: A Visual Analysis Tool to Explore Learned Representations in Transformer Models. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pages 187–196, Online. Association for Computational Linguistics.

Zhiqiang Hu, Roy Ka-Wei Lee, Charu C. Aggarwal, and Aston Zhang. 2020. Text style transfer: A review and experimental evaluation. arXiv preprint arXiv:2010.12742.

Vineet John, Lili Mou, Hareesh Bahuleyan, and Olga Vechtomova. 2019. Disentangled representation learning for non-parallel text style transfer. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 424–434, Florence, Italy. Association for Computational Linguistics.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, pages 388–395, Barcelona, Spain. Association for Computational Linguistics.

Kalpesh Krishna, John Wieting, and Mohit Iyyer. 2020. Reformulating unsupervised style transfer as paraphrase generation. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 737–762, Online. Association for Computational Linguistics.

Guillaume Lample, Sandeep Subramanian, Eric Smith, Ludovic Denoyer, Marc'Aurelio Ranzato, and Y-Lan Boureau. 2019. Multiple-attribute text rewriting. In International Conference on Learning Representations.

V. I. Levenshtein. 1966. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. Soviet Physics Doklady, 10:707.

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, Online. Association for Computational Linguistics.

Juncen Li, Robin Jia, He He, and Percy Liang. 2018. Delete, retrieve, generate: a simple approach to sentiment and style transfer. 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 1865–1874, New Orleans, Louisiana. 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. arXiv preprint arXiv:1907.11692.

Aman Madaan, Amrith Setlur, Tanmay Parekh, Barnabas Poczos, Graham Neubig, Yiming Yang, Ruslan Salakhutdinov, Alan W Black, and Shrimai Prabhumoye. 2020. Politeness transfer: A tag and generate approach. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 1869–1881, Online. Association for Computational Linguistics.

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, Hong Kong, China. Association for Computational Linguistics.

Eric Malmi, Aliaksei Severyn, and Sascha Rothe. 2020. Unsupervised text style transfer with padded masked language models. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 8671–8680, Online. Association for Computational Linguistics.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory F. Diamos, Erich Elsen, David García, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. 2018. Mixed precision training. In 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings. OpenReview.net.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and

Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.

Shrimai Prabhumoye, Yulia Tsvetkov, Ruslan Salakhutdinov, and Alan W Black. 2018. Style transfer through back-translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 866–876, Melbourne, Australia. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi S. Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6830–6841.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Xing Wu, Tao Zhang, Liangjun Zang, Jizhong Han, and Songlin Hu. 2019. Mask and infill: Applying masked language model for sentiment transfer. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 5271–5277. ijcai.org.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with BERT. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

3942

A Evaluation

Automatic Evaluation When evaluating using BLEU we used detokenized SacreBLEU (Post, 2018). For BERTScore (Zhang et al., 2020), we use the `rescale_with_baseline` option with the following hash: `roberta-large_L17_no-idf_version=0.3.5(hug_trans=4.1.0.dev0)-rescaled`.

Human Evaluation When using Amazon Mechanical Turk, we screen our annotators for English proficiency, and require all to have a greater than 95% approval rate. We hire workers with at least 1000 HITs and pay workers 5 cents per example, amounting to USD$10-15 per hour.

B Training Infrastructure

For training models we use between 1 and 8 NVIDIA V100 16GB GPUs on a DGX-1 machine running Ubuntu 16.04 on a Dual 20-Core Intel Xeon E5-2698 v4 2.2 GHz.

C Source Code & Synthetic Data

We release source code with this work, with preprocessing scripts, training scripts for both conditional lanaguage models, editors and coarse-grain taggers, edit operations extraction scripts, and synthetic data generation scripts at `https://github.com/machelreid/lewis`.

D Synthetic Data

For synthetic data generation, we generate approximately 2.2M pairs for Yelp, 2.0M pairs for Amazon, and 1M pairs for Polite. Note that when generating synthetic data on Polite, given the longer sequence length, we threshold the amount of `SLOT` tokens to be the minimum of one-third of the total sequence length and 6.

We release our synthetic data to help facilitate further development in approaches using synthetic data for this task.

E Qualitative Analysis

We analyzed 100 examples from YELP produced by LEWIS. 83% transfers were correct,6% incorrect,and 11% ambiguous (the resulting sentence expressed both styles). This is in line with automatic metrics and shows LEWIS is effective in successfully transferring style. For diversity of edits, in 59% of cases, LEWIS inverted key phrases

(*and enjoying this → and avoiding this*, *friendly folks, delicious authentic bagels → sorry folks,not authentic bagels*), in 26%, LEWIS rewrote part of the sentence in a way that is not inverting key adjectives/nouns (*and he loved it → and he said it was OK*). In 10%, LEWIS performed purely syntactic editing (*definitely not enough room → enough room*). In contrast to other editors that rely on primarily single-phrase inversion (e.g. LaserTagger), demonstrating that LEWIS provides diverse edits.

F Further Automatic Evaluation

We further evaluate our model on semantic similarity and fluency using the classifiers released by Krishna et al. (2020). Results are shown in Table 9 and 10. LEWIS improves fluency by a significant margin on all, and outperforms other methods on 2/3 datasets on semantic similarity.

Dataset	Model	Fluency
YELP	Delete and Retrieve	38.7
	TG	53.1
	DeepLatentSeq	68.1
	LEWIS	84.5
	Gold	89.4
AMAZON	Delete and Retrieve	49.2
	TG	54.6
	LEWIS	85.6
	Gold	84.5
POLITE	TG	67.5
	LEWIS	93.0
	Gold (src)	92.3

Table 9: Fluency classification from Krishna et al. (2020)

G Further automatic evaluation

We also conduct further human We also evaluate Yelp using the same 300 examples, asking five annotators which system (LEWIS, Tag and Generate, or Neither) they prefer. Results can be seen in Table 11. These results show general preference for LEWIS is consistent across fluency, content preservation, and style transfer.

Dataset	Model	Target sim.	Source sim.
YELP	Delete and Retrieve	49.2	61.0
	Deep Latent Seq	48.8	60.5
	TG	54.0	67.3
	LEWIS	61.2	77.4
	Gold	—	65.7
AMAZON	Delete and Retrieve	64.9	81.0
	TG	59.2	76.3
	LEWIS	59.6	78.8
	Gold	—	75.7
POLITE	TG	—	84.6
	LEWIS	—	87.4

Table 10: Semantic similarity classification from Krishna et al. (2020)

Attribute	LEWIS	Tag and Generate	Neither
Fluency	**50.2**	39.2	10.6
Content Preservation	**53.3**	38.3	8.4
Style	**45.2**	41.0	13.7

Table 11: Human evaluation with annotators indicating their preference when comparing aligned pairs from both LEWIS and Tag and Generate

H   Example Outputs

Source	the wine was very average and the food was even less .
LEWIS	the wine was very good and the food was even better .
LaserTagger	the wine was very good and the food was even less .
Reference	the wine was above average and the food was even better
Source	owner : a very rude man .
LEWIS	owner : a very nice man .
LaserTagger	owner : a very man .
Reference	The owner was such a friendly person.
Source	i love the food ... however service here is horrible .
LEWIS	i love the food and the service here is great .
LaserTagger	i love the food ... however service here is great .
Reference	i love the food ... and service here is awesome .

Table 12: Three examples from the Yelp test set comparing the LaserTagger trained on our synthetic data and LEWIS