Abstract

Training a model for grammatical error correction (GEC) requires a set of labeled ungrammatical/grammatical sentence pairs, but manually annotating such pairs can be expensive. Recently, the Break-It-Fix-It (BIFI) framework has demonstrated strong results on learning to repair a broken program without any labeled examples, but this relies on a perfect critic (e.g., a compiler) that returns whether an example is valid or not, which does not exist for the GEC task. In this work, we show how to leverage a pretrained language model (LM) in defining an LM-Critic, which judges a sentence to be grammatical if the LM assigns it a higher probability than its local perturbations. We apply this LM-Critic and BIFI along with a large set of unlabeled sentences to bootstrap realistic ungrammatical/grammatical pairs for training a corrector. We evaluate our approach on GEC datasets across multiple domains (CoNLL-2014, BEA-2019, GMEG-wiki and GMEG-yahoo) and show that it outperforms existing methods in both the unsupervised setting (+7.7 F0.5) and the supervised setting (+0.5 F0.5).

1 Introduction

Grammatical error correction (GEC) is the task of fixing grammatical errors in text, such as typos, tense and article mistakes. Recent works cast GEC as a translation problem, using encoder-decoder models to map bad (ungrammatical) sentences into good (grammatical) sentences (Yuan and Briscoe, 2016; Xie et al., 2016; Ji et al., 2017; Chollampatt and Ng, 2018; Junczys-Dowmunt et al., 2018). These methods rely on a combination of human-labeled data (i.e., (bad, good) pairs) (Nicholls, 2003; Yannakoudakis et al., 2011; Bryant et al., 2019) and synthetic data, which are generated by corrupting good sentences into (synthetic bad, good) pairs (Awasthi et al., 2019; Kiyono et al., 2019). Human-labeled pairs are representative of real human errors but are expensive to obtain, while synthetic pairs are cheap but are unrealistic, deviating from the distribution of grammatical errors humans make (Grundkiewicz et al., 2019). How to obtain inexpensive yet realistic paired data to improve GEC remains a key challenge, especially in domains or languages with no labeled GEC data (Napoles et al., 2019; Náplava and Straka, 2019).

Break-It-Fix-It (BIFI; Yasunaga and Liang (2021)) is a recent method to obtain realistic paired data from unlabeled data, which has shown promise in the task of source code repair. The idea of BIFI is that using an initial fixer (e.g., trained on synthetic data) and a critic that tells if an input is bad or good (e.g., compiler, which checks if code has an error), BIFI iteratively trains the fixer and a breaker to generate better paired data. Specifically, BIFI (1) applies the fixer to bad examples and keeps outputs accepted by the critic, (2) trains a breaker on the re-
Local optimum criterion of grammaticality

We achieve 65.8 / 72.9 F0.5 words. We hence compare probabilities in local work in practice, (Ng et al., 2014), BEA-2019 (Bryant et al., 2019), (1) and (2). This way, BIFI adapts the fixer to more model learning.

In this work, we propose LM-Critic, a simple approximate critic for assessing grammaticality (§3), and apply it with BIFI to learn GEC from unlabeled data (§4). Specifically, motivated by recent progress in large language models (LMs) (e.g., GPT2, GPT3; Radford et al. (2019); Brown et al. (2020)) and an intuition that a good LM assigns a higher probability to grammatical sentences than ungrammatical counterparts, we use an LM’s probability to define a critic for grammaticality. A naive approach is to deem a sentence as grammatical if its probability exceeds an absolute threshold, but this does not work in practice. In Figure 1, for instance, “Alice likes cats” (4th sentence) is grammatical but has a lower probability (according to GPT2) than “Better that it” (2nd sentence), which is ungrammatical. This is because the two sentences have different meanings and are not directly comparable. We also empirically find that this critic based on absolute threshold does not work well (§3.3.3).

3 LM-Critic

The core of our approach to GEC is a critic, which returns whether a sentence is good (grammatical) or bad (ungrammatical). Motivated by recent progress in large pretrained LMs (e.g., GPT2, GPT3; Radford et al. (2019); Brown et al. (2020)), we aim to use an LM’s probability score to define a critic for grammaticality. Specifically, we propose a criterion that deems a sentence to be good if it has the highest probability within its local neighborhood (local optimum criterion). Using this LM-Critic, we apply BIFI to the GEC task. Notably, our approach, both the LM-Critic and GEC learning, does not require labeled data.

We evaluate our proposed approach on GEC benchmarks across multiple domains, CoNLL-2014 (Ng et al., 2014), BEA-2019 (Bryant et al., 2019), GMG-yahoo, and GMG-wiki (Napoles et al., 2019). We achieve strong performance in the unsupervised setting (i.e., no labeled data), outperforming the baseline fixer trained on synthetic data by 7.7 F0.5 on average. We also evaluate in the supervised setting, where we take the state-of-the-art model GECToR (Omelianchuk et al., 2020) as the baseline fixer, and further fine-tune it by applying our approach using unlabeled data. We achieve 65.8 / 72.9 F0.5 on CoNLL-2014 / BEA-2019, outperforming GECToR by 0.5 F0.5. Our results also suggest that while existing BIFI assumed access to an oracle critic (i.e., compiler), an approximate critic (i.e., LM-Critic) can also help to improve model learning.

2 Problem setup

The task of grammatical error correction (GEC) is to map an ungrammatical sentence $x_{\text{bad}}$ into a grammatical version of it, $x_{\text{good}}$ (one that has the same intended meaning). A GEC model (fixer) $f$ aims to learn this mapping, typically using a paired dataset $D_{\text{pair}} = \{(x_{\text{bad}}^{(i)}, x_{\text{good}}^{(i)})\}$. In particular, we call it labeled if the pairs are human-annotated. In contrast, we call unlabeled data a set of raw sentences $D_{\text{unlabel}} = \{x^{(i)}\}$. For simplicity, we use “good”/“bad” to mean grammatical/ungrammatical interchangeably. Unlike a fixer, which maps $x_{\text{bad}}$ to $x_{\text{good}}$, a critic $c$ merely assesses whether an input is good or bad: for a sentence $x$,

$$c(x) = \begin{cases} 1 & \text{if } x \text{ is good} \\ 0 & \text{if } x \text{ is bad} \end{cases}$$

Given unlabeled data $x$’s (some of which are good, some of which are bad), and a language model (LM), which returns a probability distribution $p(x)$ over sentences $x$, we aim to define the critic (§3; LM-Critic) and use that to obtain the fixer (§4; BIFI).

3.1 Local optimum criterion of grammaticality

Our starting point is the idea that a good LM assigns a higher probability to grammatical sentences than ungrammatical ones. With this idea, a naive way to judge grammaticality might be to find a threshold ($\delta$) for the absolute probability, and let the critic be:

$$\text{AbsThr-Critic}(x) = \begin{cases} 1 & \text{if } p(x) > \delta \\ 0 & \text{otherwise} \end{cases}$$

However, this does not work in practice. In Figure 1, for instance, “Alice likes cats” (4th sentence) is grammatical but has a lower probability (according to GPT2) than “Better that it” (2nd sentence), which is ungrammatical. This is because the two sentences have different meanings and are not directly comparable. We also empirically find that this critic based on absolute threshold does not work well (§3.3.3).

This observation motivates us to compare sentences with the same intended meaning, and leads to the following two refined intuitions.
We implement LM-Critic by approximating the
where the idea is to compare sentences within the
Assuming the above two intuitions, we obtain the
Assume for simplicity that every sentence has
The justification is as follows. If
version). There are three decisions for implementing
LM-Critic: choice of a pretrained LM, perturbation
function, and sampling method of perturbations.

**Pretrained LM.** We experiment with various
sizes of GPT2 models (Radford et al., 2019)—
GPT2 (117M parameters), GPT2-medium (345M),
GPT2-large (774M), GPT2-xl (1.6B). These LMs
were trained on a large set of web text (40GB).

**Perturbation function.** We study three variants:
• ED1. Given a sentence, we generate edit-distance
one (ED1) perturbations in the character space.
Following prior works in typo generation (Pruthi
et al., 2019; Jones et al., 2020), we randomly in-
sert a lowercase letter, delete a character, replace
a character, or swap two adjacent characters.
• ED1 + Word-level heuristics (all). ED1 can
cover most of the character-level typos but may
not cover word-level grammatical errors, such as
missing an article. Besides ED1, here we include
heuristics for word-level perturbations used in
Awasthi et al. (2019), which randomly inserts,
deletes, or replaces a word based on its dictionary.
Please refer to Awasthi et al. for more details.
• ED1 + Word-level heuristics. We noticed
that the above word-level heuristics include
perturbations that may alter the meaning of the
original sentence (e.g., deleting/inserting “not”).
Therefore, we remove such heuristics here.

**Sampling perturbations.** As the output space
of the perturbation function is large, we obtain
samples from \( b(x) \) to be \( \hat{B}(x) \). We experiment with
random sampling with sizes of 100, 200 and 400,
motivated by the finding that with the GPT2 models,
a batch size of 100 sentences can fit into a single
GPU of 11GB memory. Other (potentially more
efficient) sampling methods include gradient-based
sampling which picks perturbation sentences in
a direction that increases the sentence probability
(analogous to adversarial perturbations; Szegedy
et al., 2013; Wallace et al., 2019), but we focus on
random sampling in this work.

The advantage of LM-Critic is that as LMs can
be trained on a wide range of unlabeled corpora, it is
unsupervised and usable in various domains of text.

### Intuition 1 (Correlation of grammaticality and
probability)
For a grammatical sentence, \( x_{\text{good}} \),
and an ungrammatical version of it (with the same
intended meaning), \( x_{\text{bad}} \), we have
\[
p(x_{\text{bad}}) < p(x_{\text{good}}). \tag{3}
\]

**Intuition 2 (Local neighborhood of sentences).**
Assume for simplicity that every sentence has
exactly one grammatical version of it (i.e., if the
sentence is grammatical, itself; if not, its corrected
version).1 For each sentence \( x \), there is a set of
sentences, \( B(x) \) (local neighborhood), that
consists of the grammatical version and all other
ungrammatical versions of \( x \).

Assuming the above two intuitions, we obtain the
following criterion for judging grammaticality,
where the idea is to compare sentences within the
meaning-preserving local neighborhood.

**Local optimum criterion of grammaticality.**
For each sentence \( x \), we let \( \hat{B}(x) \) be its local
neighborhood as defined in Intuition 2. We then have
\[
x \text{ is grammatical iff } x = \arg\max_{x' \in \hat{B}(x)} p(x'). \tag{4}
\]
The justification is as follows. If \( x \) is grammatical,
then by Intuition 1, \( x \) has a higher probability than
any other sentences in \( B(x) \), as they are ungram-
matical; hence, we have the RHS of iff. On the other
hand, if \( x \) is ungrammatical, then by Intuition 1, the
grammatical version of \( x \) has a higher probability
than \( x \), which contradicts with the RHS of iff.

The idea is to deem a sentence to be grammatical
if it has the highest probability within its meaning-
preserving local neighborhood (Figure 1). We will
next describe how to implement this criterion in
practice.

### 3.2 Implementation of LM-Critic
We implement LM-Critic by approximating the
local optimum criterion. First, for the sentence prob-
ability \( p(x) \), we use a pretrained LM’s probability
score. As obtaining the ground-truth local neighbor-
hood \( B(x) \) is difficult, we aim to get an approximate,
\( \hat{B}(x) \): we implement a sentence perturbation func-
tion \( b \), and let \( \hat{B}(x) \) be samples from \( b(x) \). To check
the grammaticality of a sentence, we apply the local

\[
\text{LM-Critic}(x) = \begin{cases} 
1 & \text{if } x = \arg\max_{x' \in \hat{B}(x)} p(x') \\
0 & \text{otherwise.}
\end{cases} \tag{5}
\]

We noticed that the above word-level heuristics include
perturbations that may alter the meaning of the
original sentence (e.g., deleting/inserting “not”).
Therefore, we remove such heuristics here.

We acknowledge that this assumption may not hold in
some cases, e.g., an ungrammatical sentence may have no
correction (“asdghgdflsa”—just a random typo?) or multiple
corrections (“The cat sleep.”—change “sleep” to the present
tense or past?). We accept this assumption considering that
it is often sufficient in common GEC datasets, and leave the
relaxation of the assumption for future work.
We study how well our LM-Critic works in practice. The way LM-Critic will be used in BIFI (§4), where the critic is run on unlabeled sentences; our study here adds commas or quotations to \( x_{\text{bad}} \) and is set to premiere on February 22. 

### Table 1: How well sentence probability returned by pretrained LMs correlates with grammaticality empirically.

<table>
<thead>
<tr>
<th>Pretrained LM</th>
<th>How often ( p(x_{\text{bad}}) &lt; p(x_{\text{good}}) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPT2</td>
<td>94.7%</td>
</tr>
<tr>
<td>GPT2-medium</td>
<td>95.0%</td>
</tr>
<tr>
<td>GPT2-large</td>
<td>95.9%</td>
</tr>
<tr>
<td>GPT2-xl</td>
<td>96.0%</td>
</tr>
</tbody>
</table>

Figure 2: Probability of grammatical (green) and ungrammatical (red) sentences computed by a pretrained LM (GPT2).

#### 3.3 Empirical analysis

We study how well our LM-Critic works in practice. We prepare an evaluation data for judging grammaticality in §3.3.1. We first perform a simple check to make sure that LMs’ probability score correlates with grammaticality (§3.3.2). We then study the performance of LM-Critic judging grammaticality (§3.3.3). The analysis we conduct in this section is just an intrinsic evaluation of LM-Critic. Our main goal is to use LM-Critic with BIFI for learning GEC, which we describe and evaluate in §4.

### 3.3.1 Evaluation data

To gain insights into how well LM-Critic judges grammaticality, we prepare a simple evaluation data consisting of \( (x_{\text{bad}}, x_{\text{good}}) \) sentence pairs. As experimenting with multiple datasets is desired in GEC (Mita et al., 2019), we construct a combined evaluation set from the dev sets of multiple GEC benchmarks, GMEG-wiki (Napoles et al., 2019), GMEG-yahoo, and BEA-2019 (Bryant et al., 2019), which span the domains of Wikipedia, Yahoo! Answers, and essay/learner English. Specifically, we sample ~600 labeled pairs of \( (x_{\text{bad}}, x_{\text{good}}) \) in total from the three benchmarks. We filter out examples where \( x_{\text{bad}} = x_{\text{good}} \) in this process. We acknowledge that while we use annotated \( (x_{\text{bad}}, x_{\text{good}}) \) pairs for the evaluation here, this does not fully match the way LM-Critic will be used in BIFI (§4), where the critic is run on unlabeled sentences; our study here is just to gain intrinsic insights into LM-Critic.

### 3.3.2 Analysis of LM probability

Using the evaluation data, we first make sure that pretrained LMs’ probability correlates with grammaticality. Figure 2 shows a histogram for the probability \( \log p(x) \) of grammatical (green) and ungrammatical (red) sentences computed by GPT2. In Table 1, we study how often pretrained LMs actually assign a higher probability to \( x_{\text{good}} \) than \( x_{\text{bad}} \) on the evaluation pairs \( (x_{\text{bad}}, x_{\text{good}}) \). We find that the LMs satisfy \( p(x_{\text{bad}}) < p(x_{\text{good}}) \) about 94% of the time, with a slight increase when using a larger model (from GPT2 to GPT2-xl). We find that the remaining pairs with \( p(x_{\text{bad}}) > p(x_{\text{good}}) \) consist mostly of cases where \( x_{\text{good}} \) adds commas or quotations to \( x_{\text{bad}} \) (see Table 3 top for examples).

### Table 2: Performance of LM-Critic, when using different choices of a perturbation function, sample size, and pretrained LM described in §3.2.

<table>
<thead>
<tr>
<th>Perturbation</th>
<th>Recognize “Good”</th>
<th>Recognize “Bad”</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( P )</td>
<td>( R )</td>
</tr>
<tr>
<td>ED1</td>
<td>58.7</td>
<td>90.1</td>
</tr>
<tr>
<td>ED1 + word</td>
<td>69.7</td>
<td>10.2</td>
</tr>
<tr>
<td>ED1 + word (all)</td>
<td>68.4</td>
<td>75.5</td>
</tr>
</tbody>
</table>

#### 3.3.3 Performance of LM-Critic

In §3.3.2 we simply made sure that pretrained LMs’ probability correlates with grammaticality. Here we study LM-Critic’s performance of judging bad/good...
basically if it finds a perturbed sentence with P
LM-Critic predicts a false “good” (labeled “bad"
sampled perturbations. If the input has the highest probability among the
higher probability, and predicts “good” correctly
pretrained LM. Recall that LM-Critic predicts
of a perturbation function, sample size, and
of our proposed LM-Critic, using different choices
of LM-Critic achieving 54.3 F
probability of all good and bad sentences in the
evaluation data. This method achieves 54.3 F
in Eq 2. We set the threshold
δ
the critic based on absolute threshold, described
in grammatical error detection/correction literature.

Baseline critic. First, as a baseline, we evaluate
the critic based on absolute threshold, described
in Eq 2. We set the threshold δ as the average probability of all good and bad sentences in the
evaluation data. This method achieves 54.3 F
and 56.0 F
using GPT2.

Proposed LM-Critic. Table 2 shows the results
of our proposed LM-Critic, using different choices of a perturbation function, sample size, and
pretrained LM. Recall that LM-Critic predicts “bad” correctly if it finds a perturbed sentence with higher probability, and predicts “good” correctly
if the input has the highest probability among the
sampled perturbations.

- Perturbation function \( b \) (top table). We set the
pretrained LM to be GPT2 and the perturbation
sample size to be 100, and vary the perturbation
function. We find that when the perturbation space is small (“ED1”), LM-Critic may make false predictions of “good”, leading to low \( P^{(\text{good})} \)
and low \( R^{(\text{bad})} \). When the perturbation space is large (“ED1 + word(all)”), LM-Critic may make false predictions of “bad”, leading to low \( R^{(\text{good})} \)
and low \( P^{(\text{bad})} \). “ED1 + word” is the most balanced and achieves the best F
overall F
henceforth, we use this perturbation method for all our
experiments. Overall, our LM-Critic outperforms
the baseline critic by substantial margins.

- Sample size of perturbations (middle table). We set the LM to be GPT2 and vary the perturbation
sample size. Increasing the sample size tends to improve \( P^{(\text{good})} \) and \( R^{(\text{bad})} \), and improve the
overall F
henceforth, we use this perturbation method for all our
experiments. Overall, our LM-Critic outperforms
the baseline critic by substantial margins.

- Pretrained LM (bottom table). We vary the
LM. Increasing the LM size makes slight or no improvement in F
henceforth, we use this perturbation method for all our
experiments. Overall, our LM-Critic outperforms
the baseline critic by substantial margins.

We also analyze when LM-Critic fails. When
LM-Critic predicts a false “good” (labeled “bad”
but predicted “good”), it is commonly because of \( P(x_{\text{bad}}) > P(x_{\text{good}}) \) (as described in §3.3.2; Table
3 top), or perturbation sampling not hitting a better
version of the input \( x_{\text{bad}} \). When LM-Critic predicts a false “bad” (labeled “good” but predicted “bad”),
it is because some perturbation \( x' \in \hat{B}(x_{\text{good}}) \) yields \( P(x') > P(x_{\text{good}}) \). Common examples are
the change of tense or singular/plural (see Table
3 bottom for examples). This indicates that even if we use a conservative edit-distance like ED1,
there may be unnecessary perturbations (tense,
singular/plural) that pretrained LMs prefer, which
is a limitation of our current LM-Critic.

The analysis done in this section is an intrinsic
evaluation of LM-Critic. Our main goal is to use
LM-Critic with BIFI for learning GEC, which we
describe in §4. While LM-Critic is not perfect in itself as we have seen in this section (it is an
approximate critic), we will show that it is helpful
for obtaining realistic paired data to improve the
downstream GEC performance. Henceforth, we use
the “ED1 + word” perturbation, a sample size
of 100, and GPT2 for our LM-Critic.

4 Learning GEC with LM-Critic

Break-It-Fix-It (BIFI; Yasunaga and Liang (2021)) is an existing method that uses a critic to obtain
realistic paired data from unlabeled data. BIFI was
originally studied in the source code repair task
where an oracle critic (e.g., compiler) exists, but
there is no oracle critic in GEC. Here, we propose to
apply BIFI to the GEC task by using LM-Critic as
the critic (§4.1), and evaluate this approach on GEC
benchmarks (§4.2). The difference from the original
BIFI is that our task is GEC rather than code repair,
and we use an approximate critic (i.e., LM-Critic)
instead of an oracle critic (i.e., compiler).

4.1 Approach

Our goal is to learn a fixer \( f \) that maps an ungrammatical sentence \( x_{\text{bad}} \) into the grammatical version
\( x_{\text{good}} \). A common method to obtain paired data for
GEC from unlabeled text is to heuristically corrupt
good sentences (synthetic data) (Awasthi et al.,
2019; Kiyono et al., 2019). However, such synthetic
errors do not match the distributions of real grammatical errors humans make, which may result
in accuracy drops (Daume III and Marcu, 2006). To
mitigate this mismatch, BIFI aims to obtain more
realistic paired data and train the fixer on it.

Specifically, BIFI takes as inputs:

- **Critic** \( c \), for which we use LM-Critic
- **Unlabeled data** \( D_{\text{unlabel}} \). Using the critic \( c \),
examples in \( D_{\text{unlabel}} \) can be split into bad ones
We evaluate on four GEC benchmarks, CoNLL-2014 test (Ng et al., 2014), BEA-2019 dev / test (Bryant et al., 2019), GMEG-wiki and GMEG-yahoo tests (Napoles et al., 2019), which span domains of essay/learner English, Wikipedia, and Yahoo! Answers. For CoNLL-2014, we use the official M^2 scorer (Dahlmeier and Ng, 2012), and for others we use the ERRANT metric (Bryant et al., 2017). We describe the training data separately for unsupervised (§4.2.2) and supervised (§4.2.3) settings.

### 4.2.2 Unsupervised setting

#### Setup and data.
We consider the setup with no labeled training data. Existing GEC works (e.g., Awasthi et al. (2019); Omelianchuk et al. (2020)) prepare synthetic paired data by heuristically corrupting sentences from the One-billion-word corpus (Chelba et al., 2013). We follow the same procedure, and train an encoder-decoder Transformer (Vaswani et al., 2017) on this synthetic data to be our baseline fixer. The size of the synthetic data is 9M pairs.

We then apply the BIFI training on top of the baseline fixer. As our unlabeled data to be used for BIFI, we want text that is likely to contain both ungrammatical and grammatical sentences. Hence, we take 10M sentences in total from the Yahoo! Answers corpus (Zhang et al., 2015) and the Wikipedia histories data (Grundkiewicz and Junczys-Dowmunt, 2014) for which we take sentences prior to revisions. This unlabeled data is in the domains of two of our benchmarks (GMEG-wiki and GMEG-yahoo) but not of CoNLL-2014 and BEA-2019.

#### Implementation details.
The encoder-decoder Transformer architecture has 12 layers, 16 attention heads and hidden state size of 768. The model parameters are initialized with the BART-base parameters and hidden state size of 768. The model incorporates realistic grammatical errors into our data (as opposed to the synthetic data), and (ii) we can verify if the “bad”-side and “good”-side of the generated pairs are actually “bad” and “good” (Eq 8, 10; red font), which improves the correctness of generated training data compared to vanilla backtranslation (Sennrich et al., 2016; Lample et al., 2018). We refer readers to Yasunaga and Liang (2021) for more details.

### 4.2 Experiments

We study our proposed approach (BIFI with LM-Critic) on GEC benchmarks, in both unsupervised and supervised settings.

#### 4.2.1 Evaluation data

We evaluate on four GEC benchmarks, CoNLL-2014 test (Ng et al., 2014), BEA-2019 dev / test (Bryant et al., 2019), GMEG-wiki and GMEG-yahoo tests (Napoles et al., 2019), which span domains of essay/learner English, Wikipedia, and Yahoo! Answers. For CoNLL-2014, we use the official M^2 scorer (Dahlmeier and Ng, 2012), and for others we use the ERRANT metric (Bryant et al., 2017). We describe the training data separately for unsupervised (§4.2.2) and supervised (§4.2.3) settings.

\[ D_{\text{bad}} = \{ x \mid x \in D_{\text{unlabel}}, c(x) = 0 \} \]
\[ D_{\text{good}} = \{ y \mid y \in D_{\text{unlabel}}, c(y) = 1 \} \]

- **Initial fixer** \( f_0 \), which could be trained on synthetic data (unsupervised setting; §4.2.2) or labeled data (supervised setting; §4.2.3) and improves the fixer by performing a cycle of data generation and training: (1) we apply the fixer \( f \) to the bad examples \( D_{\text{bad}} \), which consists of real grammatical errors made by humans, and use the critic to assess if the fixer’s output is good—if good, we keep the pair; (2) we train a breaker \( b \) on the resulting paired data—consequently, the breaker can generate more realistic errors than the initial synthetic data; (3) we apply the breaker to the good examples \( D_{\text{good}} \); (4) we finally train the fixer on the newly-generated paired data in (1) and (3). This cycle can be iterated to improve the fixer and the breaker simultaneously. Formally, BIFI does the following in each round \( k = 1, 2, \ldots, K \):

\[
P_{k}^{(f)} = \{ (x, f_{k-1}(x)) \mid x \in D_{\text{bad}}, c(f_{k-1}(x)) = 1 \} \tag{8}
\]
\[
b_k = \text{TRAIN}^{\text{good} \rightarrow \text{bad}}(P_{k}^{(f)}) \tag{9}
\]
\[
P_{k}^{(b)} = \{ (b_k(y), y) \mid y \in D_{\text{good}}, c(b_k(y)) = 0 \} \tag{10}
\]
\[
f_k = \text{TRAIN}^{\text{bad} \rightarrow \text{good}}(P_{k}^{(f)} \cup P_{k}^{(b)}) \tag{11}
\]

where each equation corresponds to the steps (1)–(4) in the description above. \( \text{TRAIN}^{\text{good} \rightarrow \text{bad}}(\mathcal{P}) \) trains an encoder-decoder model that maps “good”-side examples to “bad”-side examples in paired data \( \mathcal{P} \), and \( \text{TRAIN}^{\text{bad} \rightarrow \text{good}}(\mathcal{P}) \) does the reverse. **Red font** indicates the use of critic. The key intuition of BIFI is that thanks to the critic, (i) we can extract \( D_{\text{bad}} \) from the unlabeled data \( D_{\text{unlabel}} \) and incorporate realistic grammatical errors into our data (as opposed to the synthetic data), and (ii) we can verify if the “bad”-side and “good”-side of the generated pairs are actually “bad” and “good” (Eq 8, 10; red font), which improves the correctness of generated training data compared to vanilla backtranslation (Sennrich et al., 2016; Lample et al., 2018). We refer readers to Yasunaga and Liang (2021) for more details.
“+BIFI” but they are not verified by LM-Critic. This system (“+BIFI with no critic”) did not improve on the baseline much. These results indicate that the paired data generated by BIFI with LM-Critic is indeed more realistic and helpful than the initial synthetic data or pairs generated without LM-Critic.

The improved results in this unsupervised setting suggest that our approach is especially useful in domains with no labeled GEC data for training (e.g., GMEG-wiki and yahoo; CoNLL-2014 and BEA-2019 have labeled data, which we use in §4.2.3).

Our results also suggest that while existing BIFI assumed access to an oracle critic (i.e., compiler), an approximate critic (i.e., LM-Critic) can also help to improve model learning. Our conjecture is that as long as the LM-Critic is better than random guessing (e.g., 70 F0.5 as shown in §3.3.3), it is useful for improving the quality of GEC training data generated in BIFI (Eq 8, 10), which in turns improves GEC performance. An interesting future direction is to use the breaker learned in BIFI (Eq 9 for the perturbation function in LM-Critic (§3.2) to further improve the critic, which may in turn help BIFI as well as GEC performance, creating a positive loop of learning.

### 4.2.3 Supervised setting

#### Setup and data

We also consider the common leaderboard setup that uses labeled training data and evaluates on CoNLL-2014 and BEA-2019. We take the state-of-the-art model, GECToR (Omelianchuk et al., 2020), as our baseline fixer. Following Omelianchuk et al. (2020), GECToR is first trained on the synthetic paired data described in §4.2.2, and is then trained on the labeled data available for the BEA-2019 task, which is the combination of:

- NUS Corpus of Learner English (NUCLE) (Dahlmeier et al., 2013)
- Lang-8 Corpus of Learner English (Lang-8) (Mizumoto et al., 2011; Tajiri et al., 2012)
- FCE dataset (Yannakoudakis et al., 2011)
- Write & Improve + LOCNESS Corpus (W&I + LOCNESS) (Bryant et al., 2019)

They are all in the domain of CoNLL-2014 and BEA-2019 (learner/essay English). The total size of the labeled data is 1M pairs.

We then apply the BIFI training on top of GECToR. As our unlabeled data to be used for BIFI, we use 10M sentences taken from Yahoo! Answers and Wikipedia histories (same as §4.2.2).

#### Implementation details

We use the same hyper-parameters and training procedures for GECToR as in Omelianchuk et al. (2020). We run the BIFI algorithm for $K = 1$ round. The total training time takes 4 days, on a single GTX Titan X GPU.

#### Results

Table 5 shows our results on CoNLL-2014 test and BEA-2019 test, along with existing systems on the leaderboard. Our approach (“+BIFI”) provides an additional boost over our base model (“GECToR”). This suggests that BIFI with LM-Critic is helpful not only in the unsupervised setting but also when a substantial amount of labeled data (1M pairs) is available.

### 4.2.4 Analysis

#### Varying the amount of labeled data

We have studied GEC results when we have no labeled data (§4.2.2) and when we use all the labeled data (1M
We observe that (a) tends to deviate from the type where \( x_{\text{good}} \) is broken (e.g., the first pair) or \( x_{\text{bad}} \) is already grammatical, as pairs are not verified by a critic. (c) is the most realistic.

Table 6: Examples of paired data generated by (a) synthetic corruption, (b) BIFI without critic, and (c) BIFI with LM-Critic. (a) tends to deviate from the type of grammatical errors humans make. (b) tends to have pairs where \( x_{\text{good}} \) is broken (e.g., the first pair) or \( x_{\text{bad}} \) is already grammatical, as pairs are not verified by a critic. (c) is the most realistic.

Table 7: Examples where the baseline fixer trained with synthetic data fails but BIFI succeeds. The baseline tends to make unnecessary edits (e.g., changing verb inflection or articles, due to heuristics used when generating synthetic data).

5 Related work and discussion

Grammatical error correction (GEC). GEC models are commonly trained from human-labeled data (Nicholls, 2003; Dahlmeier et al., 2013; Yannakoudakis et al., 2011; Bryant et al., 2019), or synthetic data generated by heuristically corrupting unlabeled sentences (Awasthi et al., 2019; Zhao et al., 2019; Grundkiewicz et al., 2019; Katsumata and Komachi, 2019; Omelianchuk et al., 2020). Several works aim to improve the methods for generating paired data, such as learning a breaker from existing labeled data (Lichtarge et al., 2019), applying backtranslation (Sennrich et al., 2016) to GEC (Xie et al., 2018; Kiyono et al., 2019), and synthesizing extra paired data by comparing model predictions and references (Ge et al., 2018). Different from the above works, our method (i) does not require labeled data (works for both unsupervised and supervised settings), and (ii) uses LM-Critic to filter the “bad”-side and “good”-side of generated pairs.

Automatic text evaluation. Popular metrics used to assess the quality of text in GEC include GLEU (Napoles et al., 2015, 2017), M\(^2\) (Dahlmeier and Ng, 2012), ERRANT (Bryant et al., 2017) and I-measure (Felice and Briscoe, 2015). While these methods require reference text to compare to, LM-Critic does not. Several prior works also study reference-less methods to assess grammaticality of text: Wan et al. (2005); Mutton et al. (2007); Vadlapudi and Katragadda (2010) use part-of-speech (POS) tagger or parser predictions to score grammaticality; Naples et al. (2016); Warstadt et al. (2018); Katinskaia et al. (2019); Niu and Penn (2020) train grammatical error detection (GED) or acceptability judgment systems. However, these works require POS taggers, parsers or GED systems trained on labeled data, which may not scale or generalize well beyond the domain of training data. In contrast, LM-Critic only requires an LM, which is unsupervised and can be pretrained on various domains of unlabeled corpora.

Pretrained LM for text evaluation. Several works use pretrained LMs for text evaluation. For reference-based metrics, Zhang et al. (2020) use an LM’s embeddings to measure the similarity between input text and reference text. For reference-less
metrics, several works (Kann et al., 2018; Stahlberg et al., 2019) use an LM’s probability as a fluency score of text. While this provides a continuous score for fluency, it in itself cannot classify grammatical / ungrammatical sentences. Our LM-Critic goes a step further to consider the local optimum criterion for classifying grammaticality. The reason we want a classifier (critic) is that we work on unsupervised learning of GEC. In the unsupervised setting, there is a distributional shift problem—the synthetically-generated paired data does not match the distribution of grammatical errors humans make. BIFI is a solution for obtaining realistic paired data in an unsupervised way, but it requires a critic. This led us to design a critic for GEC in this work. We note that LM-Critic is not meant to replace existing evaluation metrics for GEC, but rather is an approximate critic to assess grammaticality and help the learning of GEC.

Separately, several works (Tenney et al., 2019; Hewitt and Manning, 2019; Yasunaga and Lafferty, 2019; Cao et al., 2020) induce grammar or syntactic structures from LMs, suggesting that LMs can learn about grammaticality in an unsupervised way. As this capacity is likely to grow with the size of LMs (Radford et al., 2019; Brown et al., 2020; Kaplan et al., 2020), we think that how to leverage pretrained LMs for GEC will become an increasingly important research problem.

6 Conclusion

We presented LM-Critic, a method that uses a pretrained language model (LM) as a critic for assessing sentence grammaticality. Using LM-Critic and the BIFI algorithm, we learn grammatical error correction (GEC) by generating realistic training data from unlabeled text. Notably, our approach does not require labeled data, and can also be viewed as an unsupervised method to turn a (GPT2-scale) pretrained LM into an actual GEC system. Using multiple GEC datasets, we showed that our approach achieves strong performance on unsupervised GEC, suggesting the promise of our method for domains and languages with no labeled GEC data. We hope this work opens up research avenues in LM-based critics and unsupervised GEC.

Acknowledgments

We thank Pang Wei Koh, Tianyi Zhang, Rodrigo Castellon, members of the Stanford P-Lambda, SNAP and NLP groups, as well as our anonymous reviewers for valuable feedback. This work was supported in part by a Funai Foundation Scholarship and NSF CAREER Award IIS-1552635.

Reproducibility

Code and data are available at https://github.com/michiyasunaga/LM-Critic.

Experiments are available at https://worksheets.codalab.org/worksheets/0x94456a63e1ee4ccfaabd7f6a3566c82.

References


Christopher Bryant, Mariano Felice, and Edward Briscoe. 2017. Automatic annotation and evaluation of error types for grammatical error correction. In Association for Computational Linguistics (ACL).


Katharina Kann, Sascha Rothe, and Katja Filippova. 2018. Sentence-level fluency evaluation: References help, but can be spared! In Conference on Computational Natural Language Learning (CoNLL).


Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel Tetreault. 2015. Ground truth for grammatical error correction metrics. *In Association of Computational Linguistics (ACL)*.

Courtney Napoles, Keisuke Sakaguchi, and Joel Tetreault. 2016. There’s no comparison: Reference-less evaluation metrics in grammatical error correction. *In Empirical Methods in Natural Language Processing (EMNLP)*.

Courtney Napoles, Keisuke Sakaguchi, and Joel Tetreault. 2017. Jfleg: A fluency corpus and benchmark for grammatical error correction. *In European Chapter of the Association for Computational Linguistics (EACL)*.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The conll-2014 shared task on grammatical error correction. *In Conference on Computational Natural Language Learning (CoNLL)*.


Danish Pruthi, Bhuwan Dhingra, and Zachary C Lipton. 2019. Combating adversarial misspellings with robust word recognition. *In Association for Computational Linguistics (ACL)*.


Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. *In Association for Computational Linguistics (ACL)*.

Felix Stahlberg, Christopher Bryant, and Bill Byrne. 2019. Neural grammatical error correction with finite state transducers. *In North American Association for Computational Linguistics (NAACL)*.


Toshikazu Tajiri, Mamoru Komachi, and Yuji Matsumoto. 2012. Tense and aspect error correction for esl learners using global context. *In Association for Computational Linguistics (ACL)*.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. Bert rediscovers the classical nlp pipeline. *In Association for Computational Linguistics (ACL)*.


Ziang Xie, Guillaume Genthial, Stanley Xie, Andrew Y Ng, and Dan Jurafsky. 2018. Noising and denoising natural language: Diverse backtranslation for grammar correction. *In North American Chapter of the Association for Computational Linguistics (NAACL)*.


