# Reflective Decoding: Beyond Unidirectional Generation with Off-the-Shelf Language Models

**Peter West**[1,2] **Ximing Lu**[1,2] **Ari Holtzman**[1]
**Chandra Bhagavatula**[2] **Jena Hwang**[2] **Yejin Choi**[1,2]
[1]Paul G. Allen School of Computer Science & Engineering, University of Washington
[2]Allen Institute for Artificial Intelligence
{pawest, ahai, yejin}@cs.washington.edu
{ximinglu, chandrab, jenah}@allenai.org

## Abstract

Publicly available, large pretrained Language Models (LMs) generate text with remarkable quality, but only sequentially from left to right. As a result, they are not immediately applicable to generation tasks that break the unidirectional assumption, such as paraphrasing or text-infilling, necessitating task-specific supervision.

In this paper, we present REFLECTIVE DECODING, a novel unsupervised algorithm that allows for direct application of unidirectional LMs to non-sequential tasks. Our 2-step approach requires no supervision or even parallel corpora, only two off-the-shelf pretrained LMs in opposite directions: *forward* and *backward*. First, in the *contextualization* step, we use LMs to generate ensembles of past and future contexts which collectively capture the input (e.g. the source sentence for paraphrasing). Second, in the *reflection* step, we condition on these "context ensembles", generating outputs that are compatible with them. Comprehensive empirical results[1] demonstrate that REFLECTIVE DECODING outperforms strong unsupervised baselines on both paraphrasing and abductive text infilling, significantly narrowing the gap between unsupervised and supervised methods. REFLECTIVE DECODING surpasses multiple supervised baselines on various metrics including human evaluation.

## 1 Introduction

Language Models (LMs) like GPT-2 (Radford et al., 2019), trained over vast unstructured data, can leverage enhanced generation methods (Holtzman et al., 2020; Martins et al., 2020; Welleck et al., 2019) to give fluent and coherent continuations to given input text—e.g. news articles or stories.

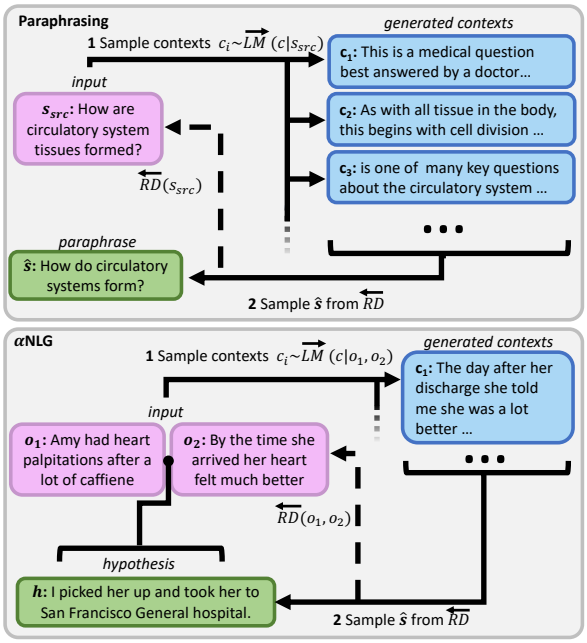


Figure 1: Illustration of REFLECTIVE DECODING applied to paraphrasing and abductive infilling ($\alpha$NLG Bhagavatula et al., 2020). **Only** the right-context is shown, although **both** right- and left-contexts are used in practice. First the contextualization step (1) captures aspects of an input by generating many representative contexts for it. Then in the reflection step (2) we sample generations that can replace the input and fit these representative contexts $\overleftarrow{\text{RD}}$.

GPT-3 (Brown et al., 2020) takes this a step further: given a small number of examples and a well-constructed prompt, it shows remarkable performance on tasks where vast quantities of supervised data and finetuning were thought to be necessary. While this demonstrates the potential for LM-decoding in few-shot or even zero-shot out-of-the-box settings, limited access to GPT-3 and immense computational cost keep this from being a widely usable or efficient solution.

Yet recent work shows that GPT-2 may hold similar capabilities when it is primed correctly. Li and

[1]Further results and resource are available at https://homes.cs.washington.edu/~pawest/ReflectiveDecoding.html



*Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pages 1435–1450
August 1–6, 2021. 

Liang (2021) achieve supervised-level performance in a few-shot setting using smaller, accessible models like GPT-2. They learn a small number of task-specific vectors as a prefix to the input, without tuning the model itself. Off-the-shelf GPT-2 is capable of few-shot learning given the right setup; our work aims to push this concept further, by showing that out-of-the-box LMs can solve complex generation problems simply by using the right decoding algorithm.

We introduce REFLECTIVE DECODING—a novel decoding method that allows LMs to be applied to generation tasks that break the "text continuation" paradigm, such as paraphrasing and text-infilling. REFLECTIVE DECODING requires *no* supervision, only two complementary off-the-shelf LMs—one forward ($\overrightarrow{\text{LM}}$) and one backward ($\overleftarrow{\text{LM}}$). That means no per-task finetuning, even on unstructured text in the target domain.

Inspired by the distributional hypothesis (Firth, 1957), REFLECTIVE DECODING works by generating text that might occupy the *same contexts* as an input. We use two LMs ($\overrightarrow{\text{LM}}$ and $\overleftarrow{\text{LM}}$) to *generate* contexts for a given input, which implicitly capture aspects of its meaning (the contextualization step). Then, in the reflection step, we condition on this ensemble of contexts, decoding *over* the input with generations that are distributionally related to—or replace—the input.

Paraphrasing is a natural application: a good paraphrase should intuitively be compatible with the same contexts as the original text. REFLECTIVE DECODING shows strong unsupervised paraphrasing performance: On the Quora question pair dataset, we find one variant of our model ($\text{RD}_{30}$) outperforms unsupervised baselines on all but one metric, and supervised baselines on both the SARI metric and human evaluation. We see the same trends on the Twitter URL corpus (Lan et al., 2017).

REFLECTIVE DECODING can also be applied to tasks that only replace part of the input, or generate within it, like **infilling**; on $\alpha$NLG (Bhagavatula et al., 2020), we outperform the best unsupervised baseline on overall quality, effectively halving the gap with supervised methods. In both applications, REFLECTIVE DECODING directly applies off-the-shelf pretrained models, without finetuning on the task or target domain. This provides evidence that off-the-shelf Language Models can excel at surprising applications, when paired with decoding algorithms designed to elicit specific kinds of information.

## 2 Method

### 2.1 Notation

Arrows indicate the order in which sampling functions condition on and generate tokens: $\rightarrow$ indicates generating from the left-most token to the right (left-to-right), while $\leftarrow$ indicates going right-to-left. For Language Models (LMs), this means $\overrightarrow{\text{LM}}$ is what is often called a "forward LM", while $\overleftarrow{\text{LM}}$ is a "backward LM". For our sampling function (RD), this also indicates which generated context is being conditioned on, e.g. $\overrightarrow{\text{RD}}$ conditions on left context, extending it to the right to generate output.

### 2.2 Overview

Currently, LM-decoding is limited to a *text continuation* paradigm. Given an input text $s_{input}$, $\text{LM}(c|s_{input})$ generates contexts $c$ that might come after (forward, i.e. $\overrightarrow{\text{LM}}$) or before (backward, i.e. $\overleftarrow{\text{LM}}$) the input. LM-decoding generates *outside* of the input by continuing it, but many tasks require us to generate *over* or *within* the input: paraphrasing requires reformulating the input, while infilling requires inserting text in the middle of it.

Reflective Decoding approaches this shortcoming by turning conventional LM-decoding around. While $\text{LM}(c|s_{input})$ generates the kinds of contexts $c$ the input might appear in, RD generates $s$ that might replace $s_{input}$ in these same contexts. The distributional hypothesis (Firth, 1957) suggests semantically similar texts appear in similar contexts, meaning RD is also likely to sample in the semantic neighborhood of $s_{input}$.

Concretely, REFLECTIVE DECODING samples $s$ that fits the same contexts as $s_{input}$ in 2 simple steps. We first sample many representative contexts $c_i$ that could neighbor the input, e.g. using $\overrightarrow{\text{LM}}$ in Figure 1. This is the **contextualization step**. Second, in the **reflection step**, we generate text in the opposite direction (using $\overleftarrow{\text{LM}}$ in Figure 1), which fits these contexts as well as $s_{input}$ fits them. To consider all $c_i$'s while decoding, we ensemble the different distributions imposed by conditioning on each $c_i$:

$$\overleftarrow{\text{RD}}\,(\text{s}) = \frac{\prod_i \overleftarrow{\text{LM}}(s|c_i)^{w_i}}{Z(s,c,w)} \qquad (1)$$

where $Z$ normalizes the fraction to a proper probability distribution (see Equation 2). In essence, this

**Algorithm 1:** Learn REFLECTIVE DECODER $\overleftarrow{\text{RD}}$

**Input:** Forward language model $\overrightarrow{\text{LM}}$
Backward language model $\overleftarrow{\text{LM}}$
Source text $s_{input}$

1: Sample contexts, $c_1 ... c_{n_c} \sim \overrightarrow{\text{LM}}(c|s_{input})$
2: Initialize parameters $\mathbf{w} = w_1 ... w_{n_c}$ s.t. $\sum w_i = 1, w_i \geq 0$
3: learn $\mathbf{w} = \arg\max_{\mathbf{w}} \overleftarrow{\text{RD}}(s_{input})$ s.t. $\sum w_i = 1, w_i \geq 0$

**Output:** $\overleftarrow{\text{RD}}$

ensemble $\overleftarrow{\text{RD}}$ restricts generations to fit all contexts $c_i$. Reversed $\overrightarrow{\text{RD}}$ is the same, except it uses $\overrightarrow{\text{LM}}$ with *left* contexts $c_i$ generated by $\overleftarrow{\text{LM}}$.

By ensembling the contexts in a Product of Experts (Hinton, 2002) framework, we can generate a hypothesis $s$ that fits the full contextual fingerprint. Yet, some contexts are more informative than others: probable but generic contexts like "See the appendix for details." are not descriptive of neighboring text. We learn weights $w_i$ to prioritize contexts $c_i$ in the ensemble that are most informative for $s_{input}$, by maximizing the probability of $s_{input}$ under Equation 1 (described in Algorithm 1). In effect, we are learning an on-the-fly autoencoder at inference time, using weighted ensembles of contexts as a representation (see §2.7, §A.1).

To motivate how this method functions, consider the paraphrasing example from Figure 1 with input $s_{input}$ = *How are circulatory system tissues formed?* Generated contexts reflect different aspects of $s_{input}$: $c_1$ situates $s_{input}$ as a question (*This is a medical question...*), while $c_2$ and $c_3$ explore central concepts (*as with all tissue...*; *about the circulatory system*). Even though each context could follow many sentences, together they form a fingerprint for $s_{input}$. A sentence that could be followed by all of $c_1, c_2, c_3$ will likely be a question ($c_1$), about tissue formation ($c_2$), and the circulatory system ($c_3$), and generally occupy the same semantic neighborhood as $s_{input}$, e.g. *How do circulatory systems form?*

In the case of paraphrasing, our task is to replace all of $s_{input}$ with something that might appear in the same contexts. Other tasks, however, might require us to replace only part of a sentence (e.g. in-context paraphrasing) or even insert text at a given position (e.g. infilling). REFLECTIVE DECODING makes this easy: simply hold part of $s_{input}$ static when we generate from RD.

### 2.3 REFLECTIVE DECODING

Here we dive into the details of REFLECTIVE DECODING, by considering the right-hand context ensemble ($\overleftarrow{\text{RD}}$), keeping in mind that the process is repeated on the left-hand as well ($\overrightarrow{\text{RD}}$).

First, in the **contextualization step** (line 1 of Algorithm 1), we sample many right-hand contexts $c_i$ for $s_{input}$, using $\overrightarrow{\text{LM}}$. These will be used as a representative sample of the contexts $s_{input}$ appears in. Second, in the **reflection step** (lines 2 & 3) our goal is to construct a sampling function $\overleftarrow{\text{RD}}$ that will yield texts similar to $s_{input}$. We define $\overleftarrow{\text{RD}}$ as:

$$\overleftarrow{\text{RD}}(s) = \frac{\prod_i \overleftarrow{\text{LM}}(s|c_i)^{w_i}}{\prod_{j=0}^{|s|} \sum_{t \in V} \prod_i \overleftarrow{\text{LM}}(t|s_{j+1:|s|} + c_i)^{w_i}} \quad (2)$$

This is equivalent to Equation 1, but giving the exact normalization factor in the denominator.

Equation 2 is a token-wise Product of Experts model, that captures the semantic neighborhood of $s_{input}$ via the combination of contexts $c_i$ and their weights $w_i$ (§2.7). We learn $w_i$ that maximize $\overleftarrow{\text{RD}}(s_{input})$ (probability of generating $s_{input}$ under $\overleftarrow{\text{RD}}$), thereby up-weighting contexts specific to $s_{input}$. We initialize these weights (line 2), then train them (line 3) using the Adam optimizer (Kingma and Ba, 2014). We normalize weights into a proper probability distribution at every step.

Reverse-direction $\overrightarrow{\text{RD}}$ is learned symmetrically, flipping the roles of $\overrightarrow{\text{LM}}$ and $\overleftarrow{\text{LM}}$ and sampling left-hand context instead (see §B.1 for details). Finally, we generate from $\overleftarrow{\text{RD}}$ (and $\overrightarrow{\text{RD}}$), sampling outputs that would appear in the same contexts as $s_{input}$. Depending on the application, we rank and select a final output in different ways, always using $\overrightarrow{\text{LM}}$ and $\overleftarrow{\text{LM}}$ together to capture bidirectional fit.

### 2.4 Implementation

**Weight Learning and Pruning** Context weights $w_i$ are learned using the Adam optimizer (Kingma and Ba, 2014). In practice this takes under 100 steps (negligible time compared to LM decoding). While we sample tens of contexts (line 1 of Algorithm 1), many end up with negligible weight under the learned distribution (Equation 2). To efficiently sample from $\overleftarrow{\text{RD}}$ and $\overrightarrow{\text{RD}}$, we drop all but the top $k_c$ contexts and renormalize weights: $k_c < n_c$ contexts are used during the reflection step.

**Parameters** We sample $n_c$ contexts to describe the source $s_{input}$. We use nucleus sampling (Holtz-

| Task: Paraphrasing | |
|---|---|
| what is it like to have a midlife crisis? | |
| $RD_{30}$ | what does it mean to have a midlife crisis? |
| $RD_{45}$ | what do you do when you have a midlife crisis? |
| is it possible to make money as a film critic? | |
| $RD_{30}$ | is there a way to make money as a film critic? |
| $RD_{45}$ | is it possible to make a living as a movie critic? |

| Task: $\alpha$NLG | |
|---|---|
| $o_1$: Ray hung a tire on a rope to make his daughter a swing. ___? $o_2$: Ray ran to his daughter to make sure she was okay. | |
| RD | He put her on the swing, and while she was on the swing, she fell off and was lying on the ground. |
| $o_1$: Tom and his family were camping in a yurt. ___? $o_2$: He chased it around until it left the yurt. | |
| RD | He went to the yurt and found a bear that was in the yurt |

Figure 2: Example generations of REFLECTIVE DECODING on paraphrasing and abductive text infilling ($\alpha$NLG). $RD_{45}$ encourages more difference from the input than $RD_{30}$ (§3.1).

man et al., 2020) with parameter $p_c$, and a maximum length of $len_c$. Once $\overrightarrow{\text{RD}}$ and $\overleftarrow{\text{RD}}$ are learned, we sample $n_s$ generations from each, of length $len_s$. We again use nucleus sampling, but choose $p_s$ per-example to account for vastly different entropy in RD (§B.3). Values for all hyperparameters are available in §B.4.

**Language Models** We train large forward ($\overrightarrow{\text{LM}}$) and backward ($\overleftarrow{\text{LM}}$) Language Models based on GPT-2 (Radford et al., 2019) using the OpenWebText training corpus (Gokaslan and Cohen, 2019)[2]. Our implementation details follow those of past work retraining GPT-2 (Zellers et al., 2019).

## 2.5 Application: Paraphrasing

To paraphrase, we begin by generating candidate outputs. Following §2.3 the REFLECTIVE DECODING sampling function is learned in each direction ($\overrightarrow{\text{RD}}$, $\overleftarrow{\text{RD}}$) using the source sentence $s_{input}$. Then, $n_s$ generations are sampled from both $\overrightarrow{\text{RD}}$ and $\overleftarrow{\text{RD}}$:

$$s_1, ..., s_{n_s} \sim \overrightarrow{\text{RD}}, s_{n_s+1}, ..., s_{2*n_s} \sim \overleftarrow{\text{RD}}$$

This gives a robust set of candidates that are compatible with the same left and right contexts as $s_{input}$. Many of these will be semantically related to $s_{input}$, but must be scored and ranked in order to select true paraphrases. REFLECTIVE DECODING is based on the notion that good "fit" with the same contexts is a robust measurement of similarity, yielding a natural "contextual scoring function" (Equation 7 and §2.7). We measure how likely candidate $s$ is to generate the same contexts that $s_{input}$ did when constructing $\overrightarrow{\text{RD}}$ and $\overleftarrow{\text{RD}}$:

$$score(s) = \frac{1}{n_c}\sum_{c_{rh}} \overrightarrow{\text{LM}}(c_{rh}|s) + \frac{1}{n_c}\sum_{c_{lh}} \overleftarrow{\text{LM}}(c_{lh}|s) \quad (3)$$

where $c_{rh}$ are the generated contexts used in $\overleftarrow{\text{RD}}$, and $c_{lh}$ for $\overrightarrow{\text{RD}}$. This explicitly estimates how similar the contexts of $s$ and $s_{input}$ are on both sides, the underlying objective of REFLECTIVE DECODING.

## 2.6 Application: Abductive Reasoning

Abductive natural language generation ($\alpha$NLG from Bhagavatula et al. 2020) is the task of filling in the blank between 2 observations $o1$ and $o2$, with a hypothesis $h$ that abductively explains them. The challenge for LM-decoding is making use of context from both sides ($o_1$ on the left and $o_2$ on the right). This is particularly challenging for unsupervised decoding methods because unidirectional LMs cannot naturally condition on both sides when generating $h$.

REFLECTIVE DECODING simplifies this problem by capturing information about both $o_1$ and $o_2$ in a single decoding function ($\overleftarrow{\text{RD}}$ or $\overrightarrow{\text{RD}}$), then holding $o_1$ and $o_2$ static at generation time (i.e. teacher forcing). Concretely, we use concatenated $o_1 + o_2$ as $s_{input}$ in Algorithm 1, and construct sampling functions $\overrightarrow{\text{RD}}$, $\overleftarrow{\text{RD}}$ informed by both observations. We are interested in sampling in between $o_1$ and $o_2$, so when sampling hypotheses $h$ from $\overleftarrow{\text{RD}}$ we condition on the right-side observation $o_2$ (and vice-versa for $\overrightarrow{\text{RD}}$ and $o_1$). This is equivalent to appending the given observation to sampled contexts:

$$\begin{aligned} h_1, ..., h_{n_{\hat{s}}} &\sim \overleftarrow{\text{RD}}(h|o_2) \\ h_{n_{\hat{s}}+1}, ..., h_{2*n_{\hat{s}}} &\sim \overrightarrow{\text{RD}}(h|o_1) \end{aligned} \quad (4)$$

Note that both $\overrightarrow{\text{RD}}$ and $\overleftarrow{\text{RD}}$ contain information about **both** $o_1$ and $o_2$, effectively turning a 2-sided contextual constraint into a 1-sided one.

We also use a task-specific scoring function to rank sampled hypotheses. We would like a hypothesis $h$ that best explains both observations, and so use Language Models to measure this:

$$score(h) = \overleftarrow{\text{LM}}(o_1|h+o_2) + \overrightarrow{\text{LM}}(o_2|o_1+h) \quad (5)$$

[2]https://github.com/yet-another-account/openwebtext

Adding $h$ should help to "explain" each observation given the other, i.e. that $o_2$ follows from $o_1 + h$ and $o_1$ from $h + o_2$. To filter hypotheses that only explain one of the two observations, we remove any that make either observation less probable than the empty hypothesis, imposing:

$$\overleftarrow{\text{LM}}(o_1|h + o_2) > \overleftarrow{\text{LM}}(o_1|o_2)$$
$$\overrightarrow{\text{LM}}(o_2|o_1 + h) > \overrightarrow{\text{LM}}(o_2|o_1)$$

### 2.7 Intuitions and Theory

Here we discuss the theoretical intuition for REFLECTIVE DECODING, as a way to sample generations that share contextual "fit" with a source text, deriving the sampling function of Equation 2.

We start by considering how to relate the meaning of two texts, generation $s$ and input $s_{input}$. We follow a distributional intuition (Firth, 1957), that meaning can be understood through the contexts in which text appears. Many distributional approaches learn contentful neural representations by predicting context given input text (Mikolov et al., 2013; Kiros et al., 2015), then compare these representations to establish semantic similarity. We can, instead, compare contexts directly—judging the difference in meaning between texts $s_{input}$ and $s$ by their divergence:

$$D_{KL}(\overrightarrow{\text{LM}}(c|s_{input}), \overrightarrow{\text{LM}}(c|\hat{s})) \quad (6)$$

We use $\overrightarrow{\text{LM}}$ to interchangeably denote the theoretical left-to-right distribution of text, and the LM estimating it. Thus, $\overrightarrow{\text{LM}}(c|s)$ is the distribution over right contexts $c$ given sentence s, and Equation 6 can be understood as the "contextual information difference" we expect $s$ to have from $s_{input}$. Note, we could similarly use left-hand context and $\overleftarrow{\text{LM}}$—and do so in practice.

We use finite-sample cross entropy as an effective empirical proxy for $D_{KL}$:

$$\hat{H}(\overrightarrow{\text{LM}}(c|s_{input}), \overrightarrow{\text{LM}}(c|s)) = \frac{1}{N} \sum_{c_i \sim \overrightarrow{\text{LM}}(c|s_{input})} -\log \overrightarrow{\text{LM}}(c_i|s) \quad (7)$$

Where $c_i \sim \overrightarrow{\text{LM}}(c|s_{input})$ indicates sampling contexts for $s_{input}$ from $\overrightarrow{\text{LM}}$. Intuitively, we want to minimize this score when generating $s$: an optimal output has a similar meaning to $s_{input}$ and so fills approximately the same contextual hole, minimizing the value of this "contextual distance".

In this form, $\hat{H}$ compares 2 complete texts–$s$ and $s_{input}$–but we are trying to **generate** $s$ for which the divergence from $s_{input}$ is low. We flip the role of "text" and "context"[3] to define a function from which we can sample $s$:

$$\overleftarrow{\text{RD}}(s_j|, s_{j+1:n}) = \frac{\prod_i \overleftarrow{\text{LM}}(s_j|s_{j+1:n} + c_i)^{w_i}}{\sum_{t \in V} \prod_i \overleftarrow{\text{LM}}(t|s_{j+1:n} + c_i)^{w_i}} \quad (8)$$

(equivalent to Equation 2, derived in §A.1) $s_j$ is the $j^{th}$ token in $s$ (sampled right-to-left from n to 0), and $V$ is the vocabulary. Weights $w_i$ are learned by maximizing the probability of $s_{input}$.

Equation 8, estimates the probability of *predicting* $s_{input}$ and $s$ from a finite set of contexts $c_i$ generated from $s_{input}$. This approximately minimizes Equation 6, as being generated by the same weighted ensemble of contexts strongly correlates with generating the same contexts in the same proportions, i.e. low divergence, due to the sparsity of language. We can sample $s$ with low contextual distance from $s_{input}$ using $\overleftarrow{\text{RD}}$. Further, we can use left context to construct $\overrightarrow{\text{RD}}$ by simply reversing the directions of the LMs used.

## 3 Experiments

### 3.1 Task: Paraphrase Generation

**Task:** Following past work, we test our paraphrasing method (§2.5) on the Quora question pair dataset. We hold out 1000 examples for testing, with the rest for training and validation (used by supervised baselines), disallowing overlap with the test set. We test a subset of models (compatible unsupervised models, MT) on the Twitter URL corpus (Lan et al., 2017), using 1000 examples from the canonical test split.

**Metrics:** Following past work, we include automatic metrics BLEU (Papineni et al., 2002), METEOR (Denkowski and Lavie, 2014), and $\text{TER}_p$ (Snover et al., 2009). These measure agreement with references, but high reference/input overlap means copying is rewarded (Mao and Lee, 2019); indeed, *copying source sentences as-is wins on these metrics (Table 1), meaning both BLEU and METEOR can be easily gamed.*

[3] Context is a symmetric relation: a given text serves as the one-sided context of its own context.

| | Method | SARI↑ | BLEU↑ | METEOR↑ | $\text{TER}_P$↓ | Human↑ | $Novelty$ ↑ |
|---|---|---|---|---|---|---|---|
| *Human* | Source | 17.8 | 56.0 | 37.6 | 48.0 | - | 0.0 |
| | Reference | 91.9 | 100.0 | 100.0 | 0.0 | 71.7 | 43.9 |
| *Supervised* | PG-IL | 32.8 | **49.1** | 33.8 | **49.0*** | 29.4 | 24.4 |
| | DiPS | **38.8** | 41.0 | 27.9 | 56.0 | 36.6 | **48.5*** |
| | BART | 36.1 | 44.7 | **34.7*** | 66.0 | **46.1** | 35.2 |
| *Supervised (Bilingual)* | MT | 35.6 | 48.1 | 33.5 | 52.0 | 59.3 | 26.8 |
| *Unsupervised* | R-VQVAE | 27.2 | 43.6 | 32.3 | 60.0 | 33.5 | 26.2 |
| | $\text{CGMH}_{Top}$ | 32.3 | 42.0 | 28.2 | 59.0 | 27.0 | 27.6 |
| | $\text{CGMH}_{30}$ | 33.9 | 40.9 | 27.5 | 60.0 | 31.5 | 29.7 |
| | $\text{CGMH}_{45}$ | 32.6 | 33.8 | 23.4 | 65.0 | 15.8 | 44.5 |
| | UPSA | 34.0 | 36.6 | 26.7 | 70.0 | 37.8 | 44.4 |
| | $\text{RD}_{Top}$ (Us) | 29.0 | **49.9*** | **33.9** | **52.0** | 27.5 | 20.8 |
| | $\text{RD}_{30}$ (Us) | **40.0*** | 46.8 | 32.2 | 57.0 | **63.2*** | 30.0 |
| | $\text{RD}_{45}$ (Us) | 38.6 | 39.9 | 28.9 | 65.0 | 61.1 | **45.0** |

Table 1: Model performance on the Quora test split. **Bold** indicates best for model-type, * indicates best overall (excluding human), underline indicates second-best for unsupervised. The first 5 columns are measures of quality, while the last measures novelty (Equation 9) or difference from the input. We rerun evaluations from past work.

Past work has emphasized the important challenge of generating novel paraphrases (Liu et al., 2010; Chen and Dolan, 2011) We address this in 3 ways. First, we explicitly quantify a simple notion of novelty:

$$Novelty(\hat{s}) = 100 - BLEU(\hat{s}, s_{input}) \quad (9)$$

to quantify the novelty-quality trade-off. Second, we include the SARI metric (Xu et al., 2016) which explicitly balances novelty from input with reference overlap. Third, we quantify an overall human quality metric accounting for this.

We have humans evaluate fluency, consistency, and novelty on Amazon Mechanical Turk. The overall score ("Human" in Table 1) is the rate examples meet thresholds for all 3: fluent enough to understand, with at most minor differences in meaning and at *least* minor differences in wording. On quora, we test 200 examples, with agreement (Fleiss' $\kappa$ Fleiss, 1971) of 0.40 (fluency), 0.54 (consistency), 0.77 (novelty) and 0.48 (overall) i.e. moderate to substantial agreement (Landis and Koch, 1977). On the Twitter corpus, we use 100 examples with agreement of 0.39, 0.42, 0.54, and 0.36, indicating fair to moderate agreement. On both we have 3 raters per example. See §C.2 for more.

**Baselines:** Parameters for REFLECTIVE DECODING are given in §B.4. We mainly compare against 3 unsupervised baselines: Controlled Sentence Generation by Metropolis Hastings (CGMH from Miao et al. 2019), Simulated Annealing (UPSA from Liu et al. 2019) and the residual VQ-VAE of Roy and Grangier (2019a) (R-VQVAE). This is a cross-section of recent approaches (VAE, editing).

We also compare against a machine-translation approach (see Sec 6), pivoting through German using Transformer (Vaswani et al., 2017) models trained on WMT19 data (Barrault et al., 2019). MT is included in a separate section in our results as it uses supervised bilingual data (Table 1).

We include supervised baselines: the pointer generator trained by imitation learning (PG-IL) as in Du and Ji (2019), the diversity-promoting DiPS model (Kumar et al., 2019), and a finetuned BART model (Lewis et al., 2019), which uses a more complex pretraining method than our LMs. Note that DiPS generates multiple diverse paraphrases so we pick one at random.

CGMH and REFLECTIVE DECODING both return multiple sampled, ranked paraphrases. We can easily control for $Novelty$ by taking the highest-ranked output that meets a $Novelty$ threshold. For both, we have a version with no threshold ($Top$), and with thresholds such that average $Novelty$ is 30 and 45. $Novelty$ cutoffs do not depend on the reference, only the source, and are equivalent to selecting with *BLEU-ori* ($Novelty$ is 100 − BLEU-ori) by Miao et al. (2019) or Bao et al. (2019).

### 3.2 Task: Abductive NLG

**Task:** The Abductive natural language generation task ($\alpha$NLG) presented in Bhagavatula et al. (2020) requires generating a hypothesis that fits

| Method | SARI↑ | Human ↑ | $Novelty$ ↑ |
|---|---|---|---|
| Source | 13.6 | - | 0.0 |
| Reference | 90.7 | 51.3 | 63.3 |
| MT | 36.1 | 70.9 | 30.4 |
| R-VQVAE | 31.1 | 32.3 | 40.4 |
| $\text{CGMH}_{Top}$ | 32.7 | 27.8 | 25.5 |
| $\text{CGMH}_{30}$ | 33.2 | 25.1 | 30.1 |
| $\text{CGMH}_{45}$ | 31.8 | 13.5 | 45.2 |
| $\text{RD}_{Top/30}$ (Us) | 31.4 | 46.5 | 37.0 |
| $\text{RD}_{45}$ (Us) | **36.4** | **56.9** | **45.3** |

Table 2: Model performance on the Twitter URL test split. **Bold** indicates best for model-type. We show only metrics accounting for novelty (more in §C.3)

between observations $o_1$ and $o_2$, and explains them. We apply REFLECTIVE DECODING to this problem as outlined in §2.6, using the given data splits.

**Metrics:** For human evaluation, over 200 examples we ask 3 raters on Amazon Mechanical Turk about coherence between h and $o_1$, $o_2$, $o_1 + o_2$, and overall quality on 4-value likert scales. We found Fleiss' kappa (Fleiss, 1971) of 0.32, 0.40, 0.41, and 0.41 respectively, indicating fair to moderate agreement (Landis and Koch, 1977).

**Baselines:** Parameters for REFLECTIVE DECODING are given in §B.4. We include baselines from the original work: different supervised variants of GPT-2 large with access to the observations, and optionally commonsense embeddings or generations from COMET (Bosselut et al., 2019). We include unsupervised baselines of GPT-2 conditioned on $o_1$ + $o_2$ directly, the gradient-based DeLorean model of Qin et al. (2020), and ILM infilling model of Donahue et al. (2020), representing recent unsupervised methods.

## 4 Results and Analysis

**Paraphrasing** First, the Quora dataset: On automatic metrics from past works (BLEU, METEOR, $\text{TER}_P$) our lowest-$Novelty$ model setting ($\text{RD}_{Top}$) achieves the highest unsupervised scores, and highest overall on BLEU. Other high scoring rows (Source, PG-IL) are similarly low-$Novelty$. The SARI metric explicitly balances $Novelty$ with similarity to reference. On SARI we see such low-$Novelty$ models perform worse. The best overall model on SARI is our medium-$Novelty$ setting ($\text{RD}_{30}$) which outperforms MT and supervised models.

Our human evaluation measures what fraction of outputs are found to be fluent, consistent, and novel. As with SARI, both our mid and high-$Novelty$ models perform quite well, again with the medium-$Novelty$ setting outperforming all baselines. As further validation for SARI as a proxy for human, they share the same top-5 models.

Results on the Twitter URL corpus largely support those on Quora. REFLECTIVE DECODING achieves the best unsupervised scores on novelty-aware metrics (Table 2), with the best overall SARI, even outperforming reference on the human metric, although MT achieves the highest overall.

In sum, REFLECTIVE DECODING is able to compete on previously used quality metrics favoring low-$Novelty$, but can produce more varied outputs preferred by humans. $\text{RD}_{45}$ is among the best models by SARI and Human on Quora despite exceeding the novelty of even the reference.

**$\alpha$NLG** Results on $\alpha$NLG (Table 3) present a strong case that REFLECTIVE DECODING can effectively use bidirectional context. Strong hypotheses use information from both initial the observation $o_1$ and the future observation $o_2$. Humans ranked the ability of REFLECTIVE DECODING to capture this 42.4, about 17 points above the next-best unsupervised baseline and only 15 points below the best supervised method tested. We see similar results for overall evaluation. A likely factor in this is the (comparatively) high degree of coherence between $h$ and $o_2$ by REFLECTIVE DECODING. Where other methods seem to pay more attention to observation $o_1$ (the $o_2$ column generally has much lower values), REFLECTIVE DECODING has comparably high coherence with left-hand ($o_1$) and right-hand ($o_2$) contexts.

We also include example generations in Figure 2 to demonstrate the ability of REFLECTIVE DECODING to combine $o_1$ and $o_2$. For example, $h =$ *He put her on the swing, and while she was on the swing, she fell off and was lying on the ground.* incorporates information from both observations. Specifically, it takes into account the swing that Ray is building for his daughter which is only mentioned in $o_1$, and hypothesizes about a potential injury due to Ray checking on his daughter in $o_2$. See appendix for more generations.

Overall, the strong performance of REFLECTIVE DECODING on $\alpha$NLG shows that unsupervised generation with context ensemble applies to infilling in addition to paraphrasing.

## 5 Discussion

**REFLECTIVE DECODING Out-of-the-Box** A major advantage to applying REFLECTIVE DECODING is ease-of-use: armed with our pretrained language models, practitioners can immediately begin generating. With general pretrained models and underlying principles that are domain-agnostic, REFLECTIVE DECODING works across a broad range of text style–no finetuning required–making exploration and adaptation simple. Multiple rounds of generation mean REFLECTIVE DECODING may run slower than other methods at inference time[4], but it avoids training time. There are clearly settings that favor supervised learning (narrow, known domain with abundant training data), but REFLECTIVE DECODING is a good option to begin generating and exploring immediately with high quality generation.

A useful abstraction for understanding REFLECTIVE DECODING for current applications is "prompting", i.e., writing a prefix to implicitly or explicitly describe a task for a pretrained model. REFLECTIVE DECODING generates natural contexts that the desired generation would appear in. This breaks from other methods of automatic prompting, which often forego "natural" prompts (Shin et al., 2020; Reynolds and McDonell, 2021), even making them continuous (Li and Liang, 2021; Hambardzumyan et al., 2021; Lester et al., 2021; Qin and Eisner, 2021). REFLECTIVE DECODING also notably creates a set of prompts (contexts) for each example, where other methods attempt to learn an overall task prompt. Still, all of these are connected by the popular intuition that useful behavior in pretrained models can be induced through contextual input.

**Future Applications** REFLECTIVE DECODING can extend beyond our experiments here, however. A simple example is in-context paraphrasing, i.e. writing a paraphrase that fits the true context that the original sentence appears in. Most existing paraphrasing methods consider only out-of-context sentences, and would require significant changes to consider context as a constraint; for REFLECTIVE DECODING we can simply combine true and generated contexts without with the same algorithm.

Driving REFLECTIVE DECODING is a notion of context as a representation, with clear potential for future work. Pretrained LMs capture rich information about text spans, but accessing it without fine-tuning is nontrivial; within the model it is an uninterpretable mass of parameters and activation weights. Our work observes that unidirectional LMs are only capturing this information to predict adjacent context–this is the sole learning signal–so all of this information is expressed in the model's context prediction. Thus, we capture some of this rich information to represent spans, by capturing a finite-sample version of this full predictive distribution in generated contexts. In REFLECTIVE DECODING specifically, we use this form of representation to generate back into the source span–paraphrasing or infilling–but the notion can be applied much more generally. In translation for instance, we might first generate contexts for the source sentence that represent its meaning, noisily translate these contexts, then impose that any translations for the source fit the same contexts under a translation-language LM. Constraining translations in this way can add robustness to existing systems by anchoring translations to informative contexts. Beyond explicit generation even, we might use a very large LM like GPT-3 to define a strong scoring function or metric as in Equation 7, first generating contexts for some target sentence, then scoring candidates by how well they generate these same contexts. As in our work, such a score could indicate how well the option fills the same contextual role as the target, harnessing the strong reasoning of whatever model is used.

## 6 Related Work

**Distributional Intuitions** A key aspect of REFLECTIVE DECODING is using a distributional intuition to represent the meaning of a text through many contexts. Kiros et al. (2015); Miao et al. (2019) quantify semantic relationships and Lin and Pantel (2001) identify paraphrastic relationships under similar intuitions. A major point of difference between past work and ours is that we sample explicit contexts, allowing unsupervised generation back from these contexts, while past work typically learns a neural representation based on contexts and conditions on this vector-encoded representation.

**Unsupervised Paraphrasing** Some approaches train neural variational auto-encoders unsupervised to represent source sentences, then decodes from these representations to paraphrase (Roy

[4]Depending on parameters we found most baselines took multiple seconds per example vs. 10s of seconds for REFLECTIVE DECODING on a multi-gpu machine.

| | $o_1$ | $o_2$ | $o_1 + o_2$ | all |
|---|---|---|---|---|
| Human | 86.3 | 89.1 | 85.1 | 84.4 |
| *Supervised* | | | | |
| COMeT$_{\text{Emb}}$+GPT2 | **69.3** | **60.1** | **56.4** | **56.3** |
| COMeT$_{\text{Txt}}$+GPT2 | 68.9 | 54.8 | 51.9 | 50.6 |
| $O_1$-$O_2$-Only | 69.2 | 57.7 | 54.3 | 53.8 |
| *Unsupervised* | | | | |
| GPT2-Fixed | 20.6 | 13.9 | 10.8 | 10.3 |
| DeLorean | 48.7 | 24.6 | 23.6 | 22.5 |
| ILM | 45.9 | 27.3 | 25.3 | 25.0 |
| Reflective Decoding | **53.4** | **51.7** | **42.4** | **41.9** |

Table 3: Model performance on $\alpha$NLG. The first 3 scores query agreement between hypothesis and given observation(s), "all" indicates overall judgement. REFLECTIVE DECODING significantly outperforms all unsupervised baselines.

and Grangier, 2019b; Bao et al., 2019). This requires training specialized representations, whereas REFLECTIVE DECODING applies general-purpose LMs. We compare to Roy and Grangier (2019b).

Paraphrasing by editing the input (Miao et al., 2019; Liu et al., 2019) has shown promise. Like REFLECTIVE DECODING, these approaches can be applied without training specialized models, but are necessarily limited by edit-paths and local minima, as edits are often restricted to single-word replacement, insertion, and deletion. Generated paraphrases must follow a continuous local edit path, while REFLECTIVE DECODING can generate new sentences from scratch.

REFLECTIVE DECODING and MT-based paraphrasing both pivot through an alternative textual form to paraphrase (context and translation, respectively). But MT paraphrasing systems cycle-translate through a pivot language (Federmann et al., 2019; Wieting and Gimpel, 2018), which requires supervised bilingual translation data, with an implicit notion of interlingual paraphrasing.

**Novelty in Paraphrasing** Mao and Lee (2019) observe that paraphrases close to the source often win on automatic quality metrics. However, dissimilarity from the source correlates with human notions of paraphrasing (Liu et al., 2010). Kumar et al. (2019) increase novelty through their diversity-promoting sampling method. Alternative metrics that consider novelty alongside quality have been proposed (Sun and Zhou, 2012; Federmann et al., 2019). The SARI metric (Xu et al., 2016), included here, combines these notions.

**Abductive Text Infilling** $\alpha$NLG (Bhagavatula et al., 2020) is a text infilling task that specifically measures the ability of models to *explain* bidirectional context (observations $o1, o2$) with a hypothesis that fits between them. This naturally fits REFLECTIVE DECODING, which fills in contextual gaps. Recent work has directly addressed this task (Qin et al., 2020) while the infilling literature is also quite applicable (Donahue et al., 2020). We compare to both of these methods on abductive infilling, showing superior results.

## 7 Conclusions

We present REFLECTIVE DECODING, a novel unsupervised text generation method for tasks that do not fit the text continuation paradigm. It uses just two pretrained Language Models to generate contexts that capture aspects of input text, generating back into the input from there. It significantly outperforms unsupervised baselines in quality and novelty for paraphrasing. Further, in abductive natural language generation it outperforms unsupervised baselines by a significant margin and halves the gap with supervised models. REFLECTIVE DECODING uses the concept of representing meaning with generated contexts, offering new possibilities for unsupervised conditional text generation.

## Acknowledgements

We thank anonymous reviewers for many helpful comments. This research is supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC) (funding reference number 401233309), DARPA CwC through ARO (W911NF15-1-0543), DARPA MCS program through NIWC Pacific (N66001-19-2-4031), the Allen Institute for AI, and a gift from Intel Labs Cognitive Computing Research.

## Ethical Considerations

In order to complete our human evaluation we used Amazon Mechanical Turk. We estimated the range of times we expected our task to take, and made sure that at minimum workers would be paid a wage of $15.00 per hour if they were solely completing our task.

As part of this effort, we plan to release our code and model. Our forward and backward language models are the same size as the publicly available GPT-2 (Radford et al., 2019). Training time/energy was likely significantly smaller than

the original release; existing code and hyperparameters were available, and we use a smaller dataset. Further, there is no publicly available backward GPT-2 model that we are aware of, so releasing a pair of forward and backward models that were trained on the same data allows for proper comparisons about left-to-right vs. right-to-left processing of English text.

We estimate that the potential dangers of releasing this from a malicious generation perspective are low. Our forward model is similar to already released GPT-2 models. While the backward model adds new generation potential and scientific novelty, it is unlikely to compare to GPT-3 (Brown et al., 2020) which many hobbyists and private companies now have access to. We believe that releasing a pair of forward and backward models will be more useful to researchers who wish to study the symmetries and asymmetries of the linguistic distribution.

## References


Yu Bao, Hao Zhou, Shujian Huang, Lei Li, Lili Mou, Olga Vechtomova, Xinyu Dai, and Jiajun Chen. 2019. Generating sentences from disentangled syntactic and semantic spaces. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6008–6019.

Loïc Barrault, Ondřej Bojar, Marta R Costa-Jussà, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Philipp Koehn, Shervin Malmasi, et al. 2019. Findings of the 2019 conference on machine translation (wmt19). In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 1–61.

Chandra Bhagavatula, Ronan Le Bras, Chaitanya Malaviya, Keisuke Sakaguchi, Ari Holtzman, Hannah Rashkin, Doug Downey, Scott Yih, and Yejin Choi. 2020. Abductive commonsense reasoning. *ICLR*.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. 2019. Comet: Commonsense transformers for automatic knowledge graph construction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4762–4779.

T. Brown, B. Mann, Nick Ryder, Melanie Subbiah, J. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, G. Krüger, T. Henighan, R. Child, Aditya Ramesh, D. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, E. Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, J. Clark, Christopher Berner, Sam McCandlish, A. Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *ArXiv*, abs/2005.14165.

David Chen and William Dolan. 2011. Collecting highly parallel data for paraphrase evaluation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 190–200, Portland, Oregon, USA. Association for Computational Linguistics.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the ninth workshop on statistical machine translation*, pages 376–380.

Chris Donahue, Mina Lee, and Percy Liang. 2020. Enabling language models to fill in the blanks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2492–2501, Online. Association for Computational Linguistics.

Wanyu Du and Yangfeng Ji. 2019. An empirical comparison on imitation learning and reinforcement learning for paraphrase generation. In *EMNLP/IJCNLP*.

Christian Federmann, Oussama Elachqar, and Chris Quirk. 2019. Multilingual whispers: Generating paraphrases with translation. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 17–26, Hong Kong, China. Association for Computational Linguistics.

John R Firth. 1957. A synopsis of linguistic theory, 1930-1955. *Studies in linguistic analysis*.

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378.

Aaron Gokaslan and Vanya Cohen. 2019. Openwebtext corpus.

Karen Hambardzumyan, Hrant Khachatrian, and Jonathan May. 2021. Warp: Word-level adversarial reprogramming. *arXiv preprint arXiv:2101.00121*.

Geoffrey E Hinton. 2002. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800.

Ari Holtzman, Jan Buys, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. *ICLR*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ryan Kiros, Yukun Zhu, Russ R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Ashutosh Kumar, Satwik Bhattamishra, Manik Bhandari, and Partha Talukdar. 2019. Submodular optimization-based diverse paraphrasing and its effectiveness in data augmentation. In *NAACL-HLT*.

Wuwei Lan, Siyu Qiu, Hua He, and Wei Xu. 2017. A continuously growing dataset of sentential paraphrases. In *Proceedings of The 2017 Conference on Empirical Methods on Natural Language Processing (EMNLP)*, pages 1235–1245. Association for Computational Linguistics.

J Richard Landis and Gary G Koch. 1977. The measurement of observer agreement for categorical data. *biometrics*, pages 159–174.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Xiang Lisa Li and P. Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *ArXiv*, abs/2101.00190.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Dekang Lin and Patrick Pantel. 2001. Dirt@ sbt@ discovery of inference rules from text. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 323–328.

Chang Liu, Daniel Dahlmeier, and Hwee Tou Ng. 2010. Pem: A paraphrase evaluation metric exploiting parallel texts. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 923–932.

Xianggen Liu, Lili Mou, Fandong Meng, Hao Zhou, Jie Zhou, and Sen Song. 2019. Unsupervised paraphrasing by simulated annealing. *arXiv preprint arXiv:1909.03588*.

Hongren Mao and Hungyi Lee. 2019. Polly want a cracker: Analyzing performance of parroting on paraphrase generation datasets. In *EMNLP/IJCNLP*.

Pedro Henrique Martins, Zita Marinho, and André F. T. Martins. 2020. Sparse text generation.

Ning Miao, Hao Zhou, Lili Mou, Rui Yan, and Lei Li. 2019. Cgmh: Constrained sentence generation by metropolis-hastings sampling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6834–6842.

Tomas Mikolov, Kai Chen, Greg S. Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space.

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 on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Guanghui Qin and Jason Eisner. 2021. Learning how to ask: Querying lms with mixtures of soft prompts. *arXiv preprint arXiv:2104.06599*.

Lianhui Qin, Vered Shwartz, Peter West, Chandra Bhagavatula, Jena D. Hwang, Ronan Le Bras, Antoine Bosselut, and Yejin Choi. 2020. Back to the future: Unsupervised backprop-based decoding for counterfactual and abductive commonsense reasoning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 794–805, Online. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. Unpublished manuscript.

Laria Reynolds and Kyle McDonell. 2021. Prompt programming for large language models: Beyond the few-shot paradigm. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–7.

Aurko Roy and David Grangier. 2019a. Unsupervised paraphrasing without translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6033–6039.

Aurko Roy and David Grangier. 2019b. Unsupervised paraphrasing without translation. In *ACL*.

Hubert JA Schouten. 1986. Nominal scale agreement among observers. *Psychometrika*, 51(3):453–466.

Thibault Sellam, Dipanjan Das, and Ankur P. Parikh. 2020. Bleurt: Learning robust metrics for text generation. In *ACL*.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235, Online. Association for Computational Linguistics.

Matthew G Snover, Nitin Madnani, Bonnie Dorr, and Richard Schwartz. 2009. Ter-plus: paraphrase, semantic, and alignment enhancements to translation edit rate. *Machine Translation*, 23(2-3):117–127.

Hong Sun and Ming Zhou. 2012. Joint learning of a dual SMT system for paraphrase generation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 38–42, Jeju Island, Korea. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. 2019. Neural text generation with unlikelihood training. *arXiv preprint arXiv:1908.04319*.

John Wieting and Kevin Gimpel. 2018. Paranmt-50m: Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 451–462.

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.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. Defending against neural fake news. In *Advances in Neural Information Processing Systems*, pages 9051–9062.

Tianyi Zhang, V. Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. *ArXiv*, abs/1904.09675.

# A Appendix

## A.1 Derivation of Sampling Function

Here we derive the sampling function used for REFLECTIVE DECODING, which allows generation using contextual similarity. This supplements §2.7. $P_{c|s}$ denotes the distribution of contexts $c$ for sentence $s$. This will be 1-sided context, for instance right-hand context $c_{rh}$ (i.e. $P_{c|s}$ would be estimated by left-to-right $\overrightarrow{\text{LM}}$ conditioned on s $\overrightarrow{\text{LM}}(c|s)$). Reversed $P_{s|c}$ goes back from context *towards* text. With right-hand context, this is estimated by $\overleftarrow{\text{LM}}(s|c)$.

In §2.7, we consider the task of comparing a source sentence $s_{src}$ with another sentence $s$. For instance, we may want to know if $s$ is a paraphrase of $s_{src}$. Following a distributional intuition (Firth, 1957) we define a simple way to compare meaning:

$$D_{KL}(P_{c|s_{src}}, P_{c|s}) \tag{10}$$

Where $D_{KL}$ is the Kullback–Leibler divergence measuring the difference between distributions $P_{c|s_{src}}$ and $P_{c|s}$. This captures a notion above: we take the amount the contexts of $s_{src}$ and $s$ differ as a proxy for their difference in meaning.

In paraphrase generation, we want to select for contextual closeness, and thus only need to rank options. We will then use cross-entropy:

$$\begin{aligned} &H(\overrightarrow{\text{LM}}(c|s_{src}),\overrightarrow{\text{LM}}(c|s)) \\ &\qquad = \sum_c -\overrightarrow{\text{LM}}(c|s_{src}) log(P_{c|s}(c)) \end{aligned} \tag{11}$$

which is equivalent to $D_{KL}$ up to a constant offset, and is easier to estimate. Here, the sum over $c$ indicates every possible context $c$, but in practice we us finite samples.

From Sec 2.7, this quantifies contextual difference in meaning. For paraphrasing, we want a sentence $s$ that minimizes this, which is equivalent to maximizing the exponent of its negation:

$$\begin{aligned} Score(s) &= e^{\sum_c -P_{c|s} log(P_{c|s}(c))} \\ &= \prod_c \left( \frac{P_{s|c}(s) P(c)}{P(s)} \right)^{P_{c|s}(c)} \\ &= \frac{a_0}{P(s)} \prod_c P_{s|c}(s)^{P_{c|s}(c)} \end{aligned} \tag{12}$$

Constant $a_0$ results from factors of $P(c)$. The result is a Product of Experts (Hinton, 2002). $P(s)^{-1}$ will prioritize more context-specific paraphrases (low probability but likely in context). However, our LMs are not well equipped to handle unlikely text, (expressivity is likely spent on likely text). Second, while less likely text can have higher similarity, this may not be the goal of our system. Rather we want related sentences that are also *fluent* and *reasonable*, so we drop $P(s)^{-1}$, the equivalent of multiplying in $P(s)$, biasing the model towards likely sequences:

$$Score(s) = c_0 \prod_c P_{s|c}(s)^{P_{c|s}(c)} \tag{13}$$

A product of experts of the form:

$$Score(s) = \prod_c P_{s|c}(s)^{w_{c|s}} \tag{14}$$

We must set the weights $w_{c|s}$ in the finite sample setting. To keep in line with this the format, we will enforce that weights constitute a proper distribution. In the limiting case (unlimited samples) $w_{c|s}$ should be set to $P_{c|s}(c)$. However, these are likely not efficient estimation weights. Further, exponentiating by this estimate will magnify errors. Instead, we learn these weights using a heuristic, discussed later.

Next, we move to the finite-sample setting, replacing distributions with LM estimates. Here we will consider right-context (meaning $P_{s|c}$ is estimated by $\overleftarrow{\text{LM}}$) but the left-context case proceeds symmetrically. Substituting in the LM distribution:

$$Score(s) = \prod_c \overleftarrow{\text{LM}}(s|c)^{w_{c|s}} \tag{15}$$

Where now the product over $c$ indicates product over the finite sampled contexts. We convert this to a sampling function, decomposing into tokens of generation $s = s_0 ... s_n$:

$$Score(s_{0:n}) = \prod_j \prod_c \overleftarrow{\text{LM}}(s_j|s_{j+1:n})^{w_{c|s}} \tag{16}$$

This restates equation 15 factorizing LM probability by tokens. Renormalizing and decomposing by token position gives a natural distribution to sample from:

$$\begin{aligned} &P_{sample}(s_j|s_{j+1:n}) = \\ &\qquad \frac{\prod_c \overleftarrow{\text{LM}}(s_j|s_{j+1:n})^{w_{c|s}}}{\sum_{t \in V} \prod_c \overleftarrow{\text{LM}}(t|s_{j+1:n})^{w_{c|s}}} \end{aligned} \tag{17}$$

**Algorithm 2:** Learn REFLECTIVE DECODING sampling function (left-to-right)

**Input:** Left to right language model $\overrightarrow{\text{LM}}$
Right to left language model $\overleftarrow{\text{LM}}$
Source text: $s_{src}$

1: Sample contexts, $c_1 ... c_{n_c} \sim \overleftarrow{\text{LM}}(c|s_{src})$

2: Initialize parameters $\mathbf{w} = w_1 ... w_{n_c}$ s.t. $\sum w_i = 1, w_i \geq 0$

3: learn $\mathbf{w} = \arg\max_{\mathbf{w}} \overrightarrow{\text{RD}}(s_{src})$ under $\sum w_i = 1, w_i \geq 0$

**Output:** $\overrightarrow{\text{RD}}$

normalizing token-wise over the vocabulary $V$ to a proper distribution (sampling right-to-left, index $n$ down, to match convention). This is referred to as $\overleftarrow{\text{RD}}$ in the body of the paper, and stated in equation 8. This samples candidate generations that encourage adherence to the contextual scoring function.

Finally, we learn the weights (a proper distribution): $s_{src}$ should receive the highest score (or similarly, should have the lowest contextual difference with itself, as it is likely in its own contexts).

# B Implementation Details

## B.1 Left-to-Right REFLECTIVE DECODING sampling function

From §2.3, $\overrightarrow{\text{RD}}$ is learned similar to $\overleftarrow{\text{RD}}$, switching the roles of $\overrightarrow{\text{LM}}$ and $\overleftarrow{\text{LM}}$ in algorithm 1. First, the roles of the language models are flipped in the sampling function:

$$\overrightarrow{\text{RD}}\,(\text{s}) = \frac{\prod_i \overrightarrow{\text{LM}}(s|c_i)^{w_i}}{\prod_{j=0}^{|s|} \sum_{t \in V} \prod_i \overrightarrow{\text{LM}}(t|s_{0:j-1} + c_i)^{w_i}} \tag{18}$$

$c_i$ are now generated by right-to-left $\overleftarrow{\text{LM}}$ (i.e. left-contexts). see Algorithm 2.

## B.2 Post-processing Generations

Without learning stop-tokens, REFLECTIVE DECODING samples fixed number ($len_s$) of tokens. Candidates are extracted from raw generations using sentence tokenization.

## B.3 Entropy Calibration

Entropy calibration is used when sampling candidate generations (§2.4). When sampling output generations, generation parameters (truncation parameter $p_s$ from nucleus sampling, in paraphrasing) control how "greedy" or stochastic sampling is. However, the effect of $p_s$ depends on many dynamic (example-wise) factors. Setting $p_s$ too low may sample only the most likely option, too high gives off-topic candidates. The "correct" value of $p_s$ is highly example-dependent.

We define **entropy calibration** to control how much "randomness" is used in sampling in a robust way. Rather than directly setting a $p_s$ for all examples, this specifies the approximate entropy $\hat{h}$ to sample with for each example. In the greedy case for instance, the desired entropy $\hat{h}$ is set to 0 (i.e. picking from a set of 1 possible option).

We search for $p_s$ in each case that is expected to give the correct entropy for the full generation, although $p_s$ is a token-level parameter. To estimate this, we take sampling entropy over the source text $s_0 ... s_n$ under the nucleus-sampling truncated distribution $\text{P}_p$:

$$\hat{h} = \tag{19}$$

$$\sum_i \sum_{w \in V_p} -P_p(w|s_0 ... s_{i-1}) \log P_p(w|s_0 ... s_{i-1}) \tag{20}$$

$V_p$ is the truncated vocabulary with parameter $p_s$. We select $p_s$ that gives a desired entropy, setthing this to 4 or 6 which we found effective (App. B.4).

| | $len_s$ | $len_c$ | $n_s$ | $n_c$ | $h$ | $p_c$ | $k_c$ |
|---|---|---|---|---|---|---|---|
| Pphrase | $inp + 5$ | 50 | 30 | 80 | 4. | 0.7 | 6 |
| $\alpha$NLG | 20 | 50 | 20 | 50 | 6. | 0.9 | 6 |

Table 4: Most parameters are explained in §2.4. $h$ is entropy for calibration in §B.3

## B.4 Parameters

Here, we give model settings for our 2 experimental settings, paraphrasing and $\alpha$NLG. See Table 4. $\alpha$NLG requires higher variety (higher $h_{sample}$, $p_c$), and fewer generated contexts ($n_c$). We experimented with different reasonable values on the dev set of each model, evaluating manually. We use transformer language models (Mega size) trained on TPU pods (TensorFlow) of size 512. These will be made publicly available. For generation we used 2 NVIDIA Titan Xp GPUs.

| | What is your creative process? | What are some tips for keeping ice cream from melting? |
|---|---|---|
| ***Unsupervised*** | | |
| **$RD_{Top}$** (us) | What is your creative process? | What's the best way to keep ice cream from melting? |
| **$RD_{30}$** (us) | What's your creative process? | What's the best way to keep ice cream from melting? |
| **$RD_{45}$** (us) | What's your creative process like? | What's the best way to keep ice cream from melting? |
| **R-VQVAE** | What is your creative process? | What tips are for keeping some ice cream from melting? |
| **UPSA** | What is in your career choice process? | what are some good tips for making ice cream with hair loss treatment? |
| **$CGMH_{Top}$** | What is your dream key? | What are some arguments for keeping crude cream from? |
| **$CGMH_{30}$** | What is your dream key? | What are some arguments for keeping crude cream from? |
| **$CGMH_{45}$** | What is your dream key? | What are some arguments for keeping crude cream from? |
| ***Supervised*** | | |
| **PG-IL** | What is your creative process? | What are some tips for ice cream from melting? |
| **DiPS** | What is your creative strategy? | How do I cure ice cream from melting? |
| **BART** | What is your creative process? | What are some ways to keep ice cream from melting? |
| ***Bilingual*** | | |
| **MT** | What is your creative process? | What tips are there to prevent ice from melting? |

Figure 3: Example generations of baselines on Quora paraphrasing dataset (§3).

| | **$o_1$: I once knew a girl named Sammy**<br>**$o_2$:She got help and everything was alright** | **$o_1$: Ray hung a tire on a rope to make his daughter a swing**<br>**$o_2$: Ray ran to his daughter to make sure she was okay.** |
|---|---|---|
| *Unsupervised* | | |
| **RD** (us) | She had problems and needed help. | He put her on the swing, and while she was on the swing, she fell off and was lying on the ground. |
| **GPT-2-fixed** | I didn't think to her, this was a normal situation | of course, that's what he does, right? |
| **DeLorean** | Sammy was a very sweet girl | She hit the rope and the tire fell on top of her. |
| **ILM** | She wanted my daughter to have a new boyfriend | His daughter was flying on the rope. |
| *Supervised* | | |
| **COMeT-Emb** | Sammy was in a car accident | Ray's daughter fell off the swing. |
| **COMeT-Txt** | Sammy got into a bad accident and her car broke down | Ray's daughter fell and fell off the swing. |
| **$O_1 + O_2$** | Sammy got hit by a drunk driver | Ray's daughter fell off the swing. |

Figure 4: Example generations of baselines on $\alpha$NLG dataset (§3). Models attempt to fill in the blank between $o_1, o_2$ to explain them both.

# C Evaluation

## C.1 Automatic Metrics

Links to the automatic metrics: ROUGE, BLEU, METEOR, $TER_P$, SARI, BERTScore,BLEURT. We include extra further metrics tested for Quora in table 5: ROUGE (Lin, 2004), BLEURT (Sellam et al., 2020), BERTScore (Zhang et al., 2020). For BLEURT, and BERTScore we use default settings. We also include iBLEU Sun and Zhou (2012) with $\alpha = 0.9$.

## C.2 Human Evaluation

Human evaluation for Quora and Twitter are largely described in §3. We reiterate that thresholds are used for each measure, and "overall" is the rate that all thresholds are met. Agreement is calculated on these binary combined threshold categories (following Schouten 1986). Full human results for paraphrasing are in Table 6. Human eval for $\alpha$NLG is described in §3.

## C.3 Twitter Dataset

We include here the full results for paraphrasing on the Twitter URL corpus (Lan et al., 2017), a set of paraphrase pairs created by linking tweets with matching shared URLs. We test unsupervised models CGMH, R-VQVAE (UPSA Twitter model is not available), and the backtranslation MT model. We include supplementary results to the main paper in Table 7.

# D Further Generations

See Figures 3,4 for outputs of REFLECTIVE DECODING and baselines.

| | Method | R-1↑ | R-2↑ | BLEURT↑ | BERTScore↑ | iBLEU↑ | $Novelty$ ↑ |
|---|---|---|---|---|---|---|---|
| *Human* | Source | 70.1 | 47.0 | 19.9 | 95.2 | 40.4 | 0.0 |
| | Reference | 100.0 | 100.0 | 99.3 | 100.0 | 84.4 | 43.9 |
| *Supervised* | PG-IL | 66.6 | 44.0 | 11.1 | 94.7 | 36.7 | 24.4 |
| | DiPS | 56.7 | 33.7 | -29.5 | 92.7 | 31.8 | 48.5 |
| | BART | 63.6 | 41.6 | 9.6 | 94.4 | 33.8 | 35.2 |
| *Supervised (Bilingual)* | MT | 64.7 | 39.8 | 16.7 | 94.8 | 35.9 | 26.8 |
| *Unsupervised* | R-VQVAE | 68.2 | 32.0 | -7.6 | 93.2 | 31.9 | 26.2 |
| | $\text{CGMH}_{Top}$ | 55.6 | 29.6 | -53.6 | 92.1 | 30.6 | 27.6 |
| | $\text{CGMH}_{30}$ | 54.5 | 28.3 | -58.9 | 91.9 | 29.8 | 29.7 |
| | $\text{CGMH}_{45}$ | 48.5 | 22.1 | -80.9 | 90.7 | 24.9 | 44.5 |
| | UPSA | 56.2 | 30.4 | -44.5 | 90.7 | 27.3 | 44.4 |
| | $\text{RD}_{Top}$ (Us) | 65.8 | 42.3 | 15.3 | 94.8 | 37.0 | 20.8 |
| | $\text{RD}_{30}$ (Us) | 62.1 | 38.0 | 7.7 | 94.2 | 35.1 | 30.0 |
| | $\text{RD}_{45}$ (Us) | 56.8 | 31.1 | -1.9 | 93.5 | 30.4 | 45.0 |

Table 5: Model performance on the Quora test split. Included here are extra metrics beyond what is in the main paper. R-1 and R-2 refer to ROUGE-1 and ROUGE-2.

| | Quora | | | | Twitter | | | |
|---|---|---|---|---|---|---|---|---|
| Method | Fluency | Consistency | Novelty | Overall (%) | Fluency | Consistency | Novelty | Overall (%) |
| Reference | 98.7 | 78.3 | 94.0 | 71.7 | 91.7 | 58.7 | 95.3 | 51.3 |
| PG-IL | 95.9 | 79.9 | 51.0 | 29.4 | - | - | - | - |
| DiPS | 85.6 | 45.1 | 93.3 | 36.6 | - | - | - | - |
| BART | 97.2 | 77.6 | 68.8 | 46.1 | - | - | - | - |
| MT | 98.7 | 88.7 | 71.2 | 59.3 | 99.0 | 90.0 | 80.9 | 70.9 |
| R-VQVAE | 84.2 | 76.3 | 60.3 | 33.5 | 65.3 | 44.0 | 94.3 | 32.3 |
| $\text{CGMH}_{Top}$ | 79.4 | 43.1 | 85.6 | 27.0 | 71.9 | 48.8 | 82.6 | 27.8 |
| $\text{CGMH}_{30}$ | 78.8 | 37.9 | 96.4 | 31.5 | 67.2 | 35.8 | 92.0 | 25.1 |
| $\text{CGMH}_{45}$ | 71.6 | 19.9 | 98.5 | 15.8 | 51.5 | 20.9 | 96.3 | 13.5 |
| UPSA | 84.4 | 46.7 | 91.6 | 37.8 | - | - | - | - |
| $\text{RD}_{Top}$ (Us) | 98.0 | 84.6 | 43.5 | 27.5 | 98.7 | 70.9 | 76.3 | 46.5 |
| $\text{RD}_{30}$ (Us) | 98.7 | 75.3 | 88.2 | 63.2 | 98.7 | 70.9 | 76.3 | 46.5 |
| $\text{RD}_{45}$ (Us) | 97.5 | 67.3 | 95.3 | 62.1 | 98.0 | 64.5 | 92.6 | 56.9 |

Table 6: Human evaluation results on both datasets for tested models. See §C.2.

| | Method | SARI↑ | BLEU↑ | METEOR↑ | $\text{TER}_P$↓ | $Novelty$ ↑ |
|---|---|---|---|---|---|---|
| *Human* | Source | 13.6 | 36.7 | 25.0 | 75.0 | 0.0 |
| | Reference | 90.7 | 100.0 | 100.0 | 0.0 | 63.3 |
| *Supervised (Bilingual)* | MT | 36.1 | 29.4 | 22.1 | 80.0 | 30.4 |
| *Unsupervised* | R-VQVAE | 31.1 | 25.2 | 21.0 | 90.0 | 40.4 |
| | $\text{CGMH}_{Top}$ | 32.7 | 28.2 | 19.8 | 77.0 | 25.5 |
| | $\text{CGMH}_{30}$ | 33.2 | 26.3 | 18.7 | 78.0 | 30.1 |
| | $\text{CGMH}_{45}$ | 31.8 | 20.5 | 15.4 | 82.0 | 45.7 |
| | $\text{RD}_{Top/30}$ (Us) | 31.4 | 27.2 | 19.9 | 86.0 | 37.0 |
| | $\text{RD}_{45}$ (Us) | 36.1 | 25.6 | 19.0 | 88.0 | 45.3 |

Table 7: Model performance on the Twitter URL test split. Note: Diversity of $\text{RD}_{Top}$ is over 30 and so this model is equivalent to $\text{RD}_{30}$ here.