Data-to-text Generation by Splicing Together Nearest Neighbors

Sam Wiseman[*]
Duke University
swiseman@cs.duke.edu

Arturs Backurs
Microsoft Research
arturs.backurs@microsoft.com

Karl Stratos
Rutgers University
karl.stratos@rutgers.edu

Abstract

We propose to tackle data-to-text generation tasks by directly splicing together retrieved segments of text from "neighbor" source-target pairs. Unlike recent work that conditions on retrieved neighbors but generates text token-by-token, left-to-right, we learn a policy that directly manipulates segments of neighbor text, by inserting or replacing them in partially constructed generations. Standard techniques for training such a policy require an oracle derivation for each generation, and we prove that finding the shortest such derivation can be reduced to parsing under a particular weighted context-free grammar. We find that policies learned in this way perform on par with strong baselines in terms of automatic and human evaluation, but allow for more interpretable and controllable generation.

1 Introduction

There has been recent interest in text generation systems that make use of retrieved "neighbors" — examples of good text retrieved from a database, perhaps paired with the source information on which these example texts condition — on the hope that these neighbors might make a generation task easier, or the system more interpretable or controllable (Song et al., 2016; Weston et al., 2018; Guu et al., 2018; Zhang et al., 2018; Peng et al., 2019, *inter alia*).

Whereas most work along these lines has adopted a conventional encoder-decoder approach, conditioning on the retrieved neighbors and then autoregressively generating text token-by-token from left to right, we instead propose to generate text by directly splicing together segments of text from retrieved neighbors. Generating in this way aligns with the intuition that in settings such as data-to-text generation it ought to be sufficient to retrieve sentences similar to the one that must be generated, and then merely change some details, such as names or dates.

There are notable advantages to a generation-by-splicing approach. First, generation becomes more interpretable: it is always clear from which neighbor a particular piece of generated text derives, and it is also clear how these pieces have come together to form the generated text. Generation-by-splicing may also increase our control over the generated text, and we suspect that approaches that make clear the provenance of each piece of generated text (as ours does) will be useful in preventing text generation systems from emitting harmful or biased text (Sheng et al., 2019; Wallace et al., 2019; Gehman et al., 2020, *inter alia*). That is, we might imagine preventing systems from emitting harmful or biased text by only allowing generation from approved neighbor examples.

Methodologically, we implement this generation-by-splicing approach by training a policy to directly insert or replace spans of neighbor text at arbitrary positions within a partially constructed generation, and we define a generalized insert function capable of such manipulations in Section 3. We train this policy with "teacher forcing" (Williams and Zipser, 1989), which requires, for each training example, an oracle sequence of insert actions that derive it. Accordingly, we define a shortest sequence of actions deriving a training generation from its neighbors to be an oracle one, and we prove that, given some neighbors, an oracle sequence of actions can be obtained by parsing under a particular weighted context-free grammar, introduced in Section 3.1.1.

Empirically, we find our proposed approach yields text of comparable quality to strong baselines under automatic metrics and human evaluation on the E2E dataset (Novikova et al., 2017) and Wikibio datasets (Lebret et al., 2016), but with added interpretability and controllability (see Section 5.2). Our reduction of minimum-

[*]This work was done primarily while the first two authors were at TTI-Chicago.

insertion generation to WCFG parsing may also be of independent interest. Our code is available at https://github.com/swiseman/neighbor-splicing.

2 Background and Notation

Conditional text generation tasks involve generating a sequence of tokens $\hat{y}_1, \ldots, \hat{y}_T = \hat{y}_{1:T}$ conditioned on some $x \in \mathcal{X}$, where each generated token $\hat{y}_t$ is from a vocabulary $\mathcal{V}$. We will consider in particular the task of table-to-text generation, where x is some tabular data and $\hat{y}_{1:T}$ is a natural language description of it.

For supervision, we will assume we have access to a dataset, which pairs an input x with a *true* corresponding reference text $y_{1:T_x} \in \mathcal{V}^{T_x}$ consisting of T_x tokens. Since we are interested in nearest neighbor-based generation, we will also assume that along with each input x we have a set $\mathcal{N} = \{\nu^{(n)}_{1:T_n}\}_{n=1}^{N}$ of N neighbor sequences, with each $\nu^{(n)}_t \in \mathcal{V}$. We will be interested in learning to form $y_{1:T_x}$ from its corresponding x and neighbor set $\mathcal{N}$ in a way that will be made more precise below. We note that finding an appropriate set of neighbor sequences to allow for successful generation with respect to an input x is an interesting and challenging problem (see, e.g., Hashimoto et al. (2018)), but for the purposes of our exposition we will assume these neighbor sequences are easy to obtain given only x (and without knowledge of y). We give the details of our simple retrieval approach in Section 5.

2.1 Imitation Learning for Text Generation

Much recent work views conditional text generation as implementing a policy $\pi : \mathcal{X} \times \mathcal{V}^* \rightarrow \mathcal{A} \cup \{\langle\text{stop}\rangle\}$ (Bengio et al., 2015; Ranzato et al., 2016); see Welleck et al. (2019) for a recent review. That is, we view a generation algorithm as implementing a policy that consumes an input $x \in \mathcal{X}$ as well as a partially constructed output in the Kleene closure of $\mathcal{V}$, which we will refer to as a "canvas" (Stern et al., 2019), and which outputs either an action $a \in \mathcal{A}$ or a decision to stop. Taking action a leads (deterministically in the case of text generation) to a new canvas, and so generation is accomplished by following π from some distinguished start canvas until a $\langle\text{stop}\rangle$ decision is made, and returning the resulting canvas. For example, sequence-to-sequence style generation (Sutskever et al., 2014; Cho et al., 2014) implements a policy π that consumes x and a canvas $\hat{y}_{1:M} \in \mathcal{V}^M$ representing a prefix, and produces either an action $a \in \mathcal{A} = \mathcal{V}$, or else a $\langle\text{stop}\rangle$ action and generation terminates. When an action $a \in \mathcal{V}$ is chosen, this leads to the formation of a new prefix canvas $\hat{y}_{1:M} \cdot a$, where $\cdot$ is the concatenation operator.

Imitation learning of text generation policies conventionally proceeds by "rolling in" to a canvas $\hat{y}_{1:M}$ using a roll-in policy, and then training a parameterized policy $\pi_{\boldsymbol{\theta}}$ to mimic the actions of an oracle policy π^* run from $\hat{y}_{1:M}$. The most common form of such training in the context of neural text generation is known as "teacher forcing" (Williams and Zipser, 1989), which simply amounts to using π^* to roll-in to a canvas $y_{1:M}$, viewing $\pi_{\boldsymbol{\theta}}$ as a probabilistic classifier, and training $\pi_{\boldsymbol{\theta}}$ using the action $\pi^*(x, y_{1:M})$ as its target.

We will adopt this policy-based perspective on conditional text generation, and we will attempt to learn a policy that generates text by splicing together text from retrieved neighbors. In order to do so, we will make use of a more general form of policy than that implemented by standard sequence-to-sequence models. Our policies will allow for both inserting an arbitrary span of neighbor text (rather than a single token) anywhere in a canvas, as well as for *replacing* an arbitrary span of text in a canvas with a span of neighbor text, as we make more precise in the next section. Before doing so, we note that there has been much recent interest in generalizing the forms of policy used in generating text; see Section 6 for references and a discussion.

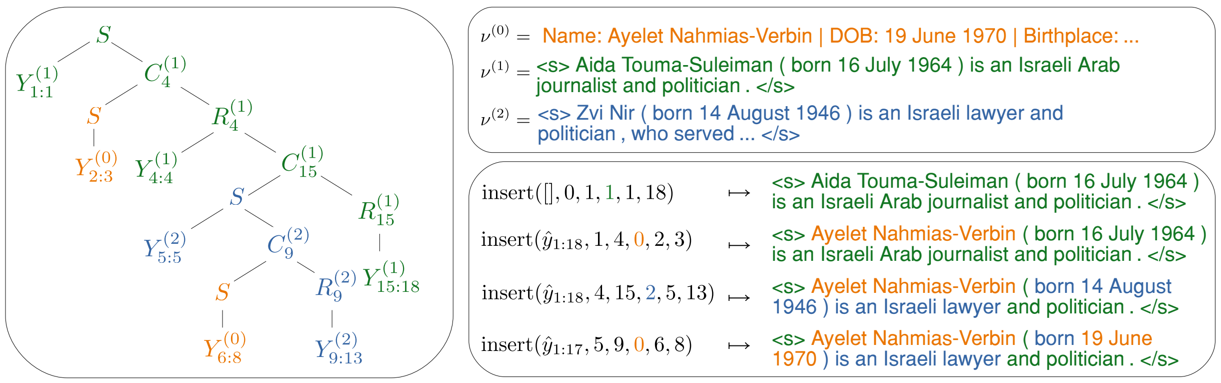
3 Splicing Nearest Neighbors

Given a canvas $\hat{y}_{1:M} \in \mathcal{V}^M$ and a set of neighbor sequences $\mathcal{N} = \{\nu^{(n)}_{1:T_n}\}_{n=1}^{N}$, we define a generalized insertion function, which forms a new canvas from $\hat{y}_{1:M}$. This generalized insertion function implements the following mapping

$$\text{insert}(\hat{y}_{1:M}, i, j, n, k, l) \mapsto \hat{y}_{1:i} \cdot \nu^{(n)}_{k:l} \cdot \hat{y}_{j:M}, \tag{1}$$

where $\cdot$ is again the concatenation operator, the slice indexing is inclusive, $0 \leq i < j \leq M + 1$,[1] and $1 \leq k \leq l \leq T_n$. Note that this generalized insertion function allows both for inserting a span into any position in the canvas (when $j = i + 1$), as well as for *replacing* a span anywhere in the canvas with another span (when $j > i + 1$), which results in the removal of tokens from the canvas.

[1]We take $\hat{y}_{1:0}$ and $\hat{y}_{M+1:M}$ to be empty sequences.



$\nu^{(0)} =$ Name: Ayelet Nahmias-Verbin | DOB: 19 June 1970 | Birthplace: ...

$\nu^{(1)} =$ <s> Aida Touma-Suleiman (born 16 July 1964) is an Israeli Arab journalist and politician . </s>

$\nu^{(2)} =$ <s> Zvi Nir (born 14 August 1946) is an Israeli lawyer and politician , who served ... </s>

$\text{insert}([], 0, 1, 1, 1, 18)$	$\mapsto$	<s> Aida Touma-Suleiman (born 16 July 1964) is an Israeli Arab journalist and politician . </s>
$\text{insert}(\hat{y}_{1:18}, 1, 4, 0, 2, 3)$	$\mapsto$	<s> Ayelet Nahmias-Verbin (born 16 July 1964) is an Israeli Arab journalist and politician . </s>
$\text{insert}(\hat{y}_{1:18}, 4, 15, 2, 5, 13)$	$\mapsto$	<s> Ayelet Nahmias-Verbin (born 14 August 1946) is an Israeli lawyer and politician . </s>
$\text{insert}(\hat{y}_{1:17}, 5, 9, 0, 6, 8)$	$\mapsto$	<s> Ayelet Nahmias-Verbin (born 19 June 1970) is an Israeli lawyer and politician . </s>

Figure 1: Deriving a sentence from the WikiBio dataset, “Ayelet Nahmias-Verbin (born 19 June 1970) is an Israeli lawyer and politician.” Top right: neighbor sequences $\nu^{(0)}, \nu^{(1)}, \nu^{(2)}$; $\nu^{(0)}$ is from the corresponding table. Bottom right: a sequence of insert operations (see Equation (1)) deriving the sentence from the neighbors above. Left: the parse of the target sentence under the grammar in Section 3.1.1 corresponding to the derivation on the bottom right.

Intuitively, this generalized insertion function attempts to capture a generation scheme where text is generated by making only minor insertions or replacements in some existing text. For example, we might imagine generating a new sentence by copying a neighbor sentence to our canvas, and then simply replacing the names or dates in this neighbor sentence to form a new sentence; see Figure 1 for an example.

Having defined this insertion function, we can generate text with a policy that consumes an input x, a set of neighbors, and a canvas,[2] and outputs the arguments of the insert function, or else the $\langle\text{stop}\rangle$ action. Thus, for a given canvas and neighbor set, we take our policy to produce actions in

$$\mathcal{A}_{\text{ins}}(\hat{y}_{1:M}) = \{(i,j,n,k,l) \in \mathbb{N}^5 \mid 0 \leq i < j \leq M{+}1, \\ 1 \leq n \leq N, 1 \leq k \leq l \leq T_n\}$$

or else the $\langle\text{stop}\rangle$ action. We show examples of generating with such a policy in Figure 1 and Figure 3.

3.1 An Oracle Policy

As described in Section 2.1, we are interested in learning a parameterized text generation policy $\pi_{\boldsymbol{\theta}}$. Since we would like to generate using the generalized insert function in Equation (1), we will attempt to learn a parameterized distribution $\pi_{\boldsymbol{\theta}}(\cdot \mid x, \hat{y}_{1:M}, \mathcal{N})$ over the arguments to this insert function given input x, canvas $\hat{y}_{1:M}$, and neighbor set $\mathcal{N}$, by training it with the one-hot action distribution $\pi^*(x, y_{1:M}, \mathcal{N})$ as its target. In order to do so, however, we must first obtain an oracle policy π^*. That is, for each true output $y_{1:T_x}$ in our dataset, we require an oracle sequence of canvases paired with corresponding oracle actions in $\mathcal{A}_{\text{ins}}$, which derive $y_{1:T_x}$ from x and $\mathcal{N}$. In this section we suggest an approach to obtaining these.

In what follows, we will assume that each word type represented in $y_{1:T_x}$ is also represented among the neighbor sequences; in practice, we can always ensure this is the case by using the following expanded neighbor set: $\mathcal{N}' = \{\nu^{(n)}_{1:T_n}\}_{n=1}^{N} \cup \mathcal{V}$, where the vocabulary $\mathcal{V}$ is viewed as containing spans of length one. Thus, policies will be able to emit any word in our vocabulary. Furthermore, because the source table x itself will often also contain spans of words that might be used in forming $y_{1:T_x}$, going forward we will also assume $\mathcal{N}'$ includes these spans from x.

In arriving at an oracle policy, we first note that given $\mathcal{N}'$ there will often be many sequences of actions in $\mathcal{A}_{\text{ins}}$ that derive the reference text $y_{1:T_x}$ from an empty canvas. For instance, we can simulate standard left-to-right, token-by-token generation with a sequence of actions $((i, i+1, n_i, k_i, k_i))_{i=0}^{T-1}$ such that $\nu^{(n_i)}_{k_i} = y_i$. However, other derivations, which insert or replace spans at arbitrary canvas locations, will often be available. We posit that derivations with fewer actions will be more interpretable, all else equal, and so we define our oracle policy to be that which derives $y_{1:T_x}$ from $\mathcal{N}'$ (starting from an empty canvas) in as few actions as possible. We show this optimization problem can be reduced to finding the lowest-cost parse of $y_{1:T_x}$ using a particular weighted context free grammar (WCFG) (Salomaa, 1969), which can be done in polynomial time with, for instance, the CKY algorithm (Kasami, 1966; Younger, 1967;

[2]To conform with our generation policy definition above, we can view $\mathcal{X}$ as containing input and neighbor set pairs.

Baker, 1979).

3.1.1 Reduction to WCFG Parsing

Consider the following WCFG in Chomsky Normal Form (Chomsky, 1959):

$$\begin{array}{llll} [1] & S & \to \nu^{(n)}_{k:l} & \forall n, k \leq l \\ [0] & S & \to S\, S & \\ [1] & S & \to \nu^{(n)}_{k:l}\, C^{(n)}_{s} & \forall n, l < s \\ [0] & C^{(n)}_{s} & \to S\, R^{(n)}_{s} & \forall n, s \\ [0] & R^{(n)}_{s} & \to \nu^{(n)}_{s:t} & \forall n, s \leq t \\ [0] & R^{(n)}_{s} & \to \nu^{(n)}_{s:t}\, C^{(n)}_{u} & \forall n, s \leq t < u, \end{array}$$

where S is the start non-terminal and where the bracketed number gives the cost of the rule application. We can see that there is a cost (of 1) for introducing a new neighbor span with an S non-terminal, but no cost for continuing with the remainder of a neighbor already introduced (as represented by the C and R non-terminals).

Claim 1. *Given neighbors $\{\nu^{(n)}_{1:T_n}\}_{n=1}^{N}$, the length of the shortest derivation of a sequence $y_{1:T_x}$ using actions in $\mathcal{A}_{\text{ins}}$ is equal to its lowest cost derivation under the WCFG above.*

We prove the above claim in Appendix A. The proof proceeds by two simulation arguments. First, we show that, given a derivation of $y_{1:T_x}$ with a certain weight under the WCFG, we can simulate a subset of the derivations with a number of insert operations equal to the total weight of the derivations and still obtain $y_{1:T_x}$. This implies that the cost of the *optimal* derivation under the WCFG is at least the cost of the *optimal* number of insert operations. Second, we show that, given a derivation of $y_{1:T_x}$ with a certain number of insert operations, we can simulate a subset of these operations with a cost-1 derivation of the grammar per insert operation. This implies that the optimal number of insert operations is at least the cost of the *optimal* derivation according to the WCFG. Together, the two simulation arguments imply the claim.

Complexity If T is the maximum length of all sequences in $\{y_{1:T_x}\} \cup \mathcal{N}'$ and $|\mathcal{N}'| = N$, parsing under the above WCFG with the CKY algorithm is $O(NT^6)$. The runtime is dominated by matching the $S \to Y^{(n)}_{k:l}\, C^{(n)}_{s}$ rule; there are $O(NT^3)$ sequences that match the right-hand side (all $k \leq l < s$ for all $\nu^{(n)}$), and we must consider this rule for each span in $y_{1:T_x}$ and each split-point.

Obtaining the policy Using Claim 1, we obtain an oracle action sequence deriving $y_{1:T_x}$ from its neighbors $\mathcal{N}'$ by first computing the minimum-cost parse tree. As noted in Section 3.1, $\mathcal{N}'$ is guaranteed to contain any word-type in $y_{1:T_x}$. In practice, we ensure this by only adding word-types to $\mathcal{N}$ that are not already represented in some neighbor, so that computed oracle parses use the neighbor sequences rather than the vocabulary. Given the minimum-cost parse tree, we then obtain a sequence of insert actions by doing a depth-first left-to-right traversal of the tree.[3] In particular, we can obtain all the arguments for an insert operation after seeing all the children of its corresponding S non-terminal. For example, in Figure 1, the insert operations on the bottom right follow the order in which S non-terminals are encountered in a left-to-right, depth-first traversal of the tree on the left; the arguments of the operation that introduces $\nu^{(2)}$, for example, are determined by keeping track of the corresponding S's distance from the left sentence-boundary and the length of the span it yields. We precompute these oracle derivations for each $(x, y_{1:T_x}, \mathcal{N}')$ triplet in our training corpus.

3.2 Additional Oracle Policies

We will refer to policies derived as above as "FULL" policies. While FULL policies minimize the number of insert operations used in deriving $y_{1:T_x}$, there are at least two other reasonable neighbor-based oracle policies that suggest themselves. One is the oracle policy that derives $y_{1:T_x}$ from left to right, one token at a time. This policy is identical to that used in training sequence-to-sequence models, except each token comes from $\mathcal{N}'$. In particular, a generated token is always copied from a neighbor sequence if it can be. We will refer to this policy as "LRT," for "left-to-right, token-level."

Another oracle policy one might consider would allow for inserting spans rather than words left-to-right, but like FULL policies would attempt to minimize the number of span insertion operations. While a greedy algorithm is sufficient for deriving such policies, in preliminary experiments we found them to consistently underperform both FULL and LRT, and so we do not consider them further.

[3]While there is a derivation corresponding to *each* depth-first traversal, the left-to-right traversal performed best.

4 Models, Training, and Generation

To avoid directly parameterizing a distribution over the impractically large number of combinations of arguments to the insert function, we factorize the distribution over its arguments as

$$\pi_{\boldsymbol{\theta}}(i, j, n, k, l) = \pi_{\boldsymbol{\theta}}(j, l \,|\, i, n, k) \times \pi_{\boldsymbol{\theta}}(i, n, k), \tag{2}$$

where we have left out the explicit conditioning on x, $\hat{y}_{1:M}$, and $\mathcal{N}'$ for brevity. Thus, our policy first predicts an insertion of token $\nu_k^{(n)}$ after the canvas token $\hat{y}_i$. Conditioned on this, the policy then predicts the final token $\nu_l^{(n)}$ of the inserted span, and which canvas token $\hat{y}_j$ immediately follows it.

More concretely, we obtain token-level representations $\mathbf{x}_1, \ldots, \mathbf{x}_S$ and $\hat{\mathbf{y}}_0, \ldots, \hat{\mathbf{y}}_{M+1}$, all in $\mathbb{R}^d$, of source sequence $x = x_{1:S}$ and of canvas sequence $\hat{y}_{1:M}$, padded on each side with a special token, by feeding them to an encoder-decoder style transformer (Vaswani et al., 2017) with no causal masking. We obtain neighbor token representations $\boldsymbol{\nu}_k^{(n)}$ by feeding neighbor sequences through the same encoder transformer that consumes x. We provide additional architectural details in Appendix D. Viewing the source sequence x as the 0th neighbor, we then define

$$\pi_{\boldsymbol{\theta}}(i, n, k) \propto \begin{cases} \exp(\hat{\mathbf{y}}_{i-1}^{\top} \mathbf{W}_1 \mathbf{W}_0 \, \mathbf{x}_k) & \text{if } n = 0 \\ \exp(\hat{\mathbf{y}}_{i-1}^{\top} \mathbf{W}_1 \mathbf{W}_2 \, \boldsymbol{\nu}_k^{(n)}) & \text{if } n > 0, \end{cases}$$

where the normalization is over all pairings of a canvas token with either a neighbor or source token, and where $\mathbf{W}_0$, $\mathbf{W}_1$, and $\mathbf{W}_2$, all in $\mathbb{R}^{d \times d}$, are learnable transformations. Similarly, we let

$$\pi_{\boldsymbol{\theta}}(j, l \,|\, i, n, k) \propto \begin{cases} \exp(\hat{\mathbf{y}}_{j+1}^{\top} \mathbf{W}_4 \mathbf{W}_3 \, \mathbf{x}_l) & \text{if } n = 0 \\ \exp(\hat{\mathbf{y}}_{j+1}^{\top} \mathbf{W}_4 \mathbf{W}_5 \, \boldsymbol{\nu}_l^{(n)}) & \text{if } n > 0, \end{cases}$$

where now the normalization only considers pairings of the jth canvas token with the lth neighbor or source token, where $j > i, l \geq k$. $\mathbf{W}_3$, $\mathbf{W}_4$, and $\mathbf{W}_5$ are again learnable and in $\mathbb{R}^{d \times d}$.

While the above holds for FULL policies, LRT policies always insert after the most recently inserted token, and so they require only a $\pi_{\boldsymbol{\theta}}(n, k \,|\, i)$ policy, and only $\mathbf{W}_0$ and $\mathbf{W}_2$ transformations.

4.1 Training

As noted in Section 3.1, we propose to train our policies to imitate the oracle derivations obtained by the CKY parse, using teacher-forcing. Suppose that, for a given (oracle) canvas $y_{1:M}$ and set of neighbors $\mathcal{N}'$, the oracle next-action obtained from the parse is $(i^*, j^*, n^*, k^*, l^*)$. Since there may be multiple spans in $\mathcal{N}'$ that are identical to $\nu_{k^*:l^*}^{(n^*)}$, we train the $\pi_{\boldsymbol{\theta}}(i, n, k)$ policy to minimize

$$-\log \sum_{\{(n,k,l) | \nu_{k:l}^{(n)} = \nu_{k^*:l^*}^{(n^*)}\}} \pi_{\boldsymbol{\theta}}(i^*, n, k). \tag{3}$$

The $\pi_{\boldsymbol{\theta}}(j, l \,|\, i, n, k)$ policy is simply trained to minimize

$$-\log \pi_{\boldsymbol{\theta}}(j^*, l^* \,|\, i^*, n^*, k^*), \tag{4}$$

since there is one correct target given i^*, n^*, k^*.

Training proceeds by sampling a mini-batch of examples and their derivations, and minimizing the sums of the losses (3) and (4) over each action in each derivation, divided by the mini-batch size.

4.2 Generation

For FULL models, we generate with beam search, following the factorization in Equation 2. At each iteration, the beam first contains the top-K partial hypotheses that can be constructed by predicting the i, n, k arguments to the insert function given the current canvas and neighbors. Given these, the remaining j, l arguments are predicted, and the top K of these are kept for the next iteration. We search up to a maximum number of actions, and in computing the final score of a hypothesis, we average the $\pi_{\boldsymbol{\theta}}$ log probabilities over all the actions taken to construct the hypothesis (rather than summing).

For LRT models, we generate with standard left-to-right, token-level beam search. We note that in this setting it is common to marginalize over all occurrences of a word-type (e.g., among neighbors or in the table) in calculating its probability. While this generally improves performance (see below), it also hurts interpretability, since it is no longer clear which precise neighbor or source token gives rise to a predicted token. Below we report results in both the standard marginalization setting, and in a no-marginalization ("no-marg") setting.

5 Experiments

Our experiments are designed to test the quality of the text produced under FULL policies and LRT policies, as well as whether such policies allow for more controllable or interpretable generation.

Datasets We expect our approach to work best for tasks where different generations commonly share surface characteristics. Table-to-text tasks meet this requirement, and are accordingly often used to evaluate generation that makes use of retrieved neighbors (Peng et al., 2019; Lin et al., 2020) or induced templates (Wiseman et al., 2018; Li and Rush, 2020). Following recent work, we evaluate on the E2E (Novikova et al., 2017) and WikiBio (Lebret et al., 2016) datasets.

Preprocessing We whitespace-tokenize the text, and mask spans in neighbor sequences that appear in their corresponding sources, which discourages derivations from copying content words from neighbors. We pad each y and $\nu^{(n)}$ sequence with beginning- and end-of-sequence tokens, which encourages derivations that insert into the middle of sequences, rather than merely concatenating spans.

Obtaining Neighbors We precompute neighbors for each training example, taking the top-scoring 20 neighbors for each example in the training data (excluding itself) under a simple score $s(\cdot, \cdot)$ defined over pairs of inputs in $\mathcal{X}$. For the E2E and WikiBio datasets, we define $s(x, x') = F_1(\text{fields}(x), \text{fields}(x')) + 0.1 F_1(\text{values}(x), \text{values}(x'))$, where fields extracts the field-types (e.g., "name") from the table x, values extracts the unigrams that appear as values in x, and F_1 is the F_1-score.

Baselines We compare FULL policies to LRT policies, to a transformer-based sequence-to-sequence model with a copy mechanism (Gu et al., 2016) that uses no retrieved neighbors (henceforth "S2S+copy"), and to recent models from the literature (see below). The S2S+copy model uses a generation vocabulary limited to the 30k most frequent target words. The neighbor-based policies, on the other hand, are limited to generating (rather than copying) only from a much smaller vocabulary consisting of target words that occur at least 50 times in the training set and which cannot be obtained from the target's corresponding neighbors.

Additional Details All models are implemented using 6-layer transformer encoders and decoders, with model dimension 420, 7 attention heads, and feed-forward dimension 650;[4] all models are trained from scratch. We train with Adam (Kingma and Ba, 2015; Loshchilov and Hutter, 2018), using linear learning-rate warm-up and square-root decay as in Devlin et al. (2019), until validation loss stops decreasing. We generate with beam search (see Section 4.2), and neighbor-based models use 20 neighbors at test time, just as at training time. We discuss hyperparameters and tuning in Appendix D. We include sample generations from all systems in Appendix F, and additional visualizations of some FULL generations in Figure 3.

[4]We use the huggingface (Wolf et al., 2020) implementation of the BART (Lewis et al., 2020b) architecture, but with no pretraining.

E2E	BLEU	NIST	RG	CID	MET
FULL	70.5	9.54	76.0	2.37	49.6
LRT-no-marg	55.8	7.39	63.7	1.68	41.0
LRT	68.1	8.83	70.0	2.38	46.2
S2S+copy	64.7	8.26	69.1	2.22	43.7
Li & Rush	67.1	8.52	68.7	2.24	45.4
KGPT	68.1	-	70.9	-	45.8
WB	BLEU	NIST	RG-4		
FULL	43.5	9.59	41.4		
LRT-no-marg	45.4	10.16	39.6		
LRT	45.7	9.99	44.0		
S2S+copy	45.4	9.72	44.6		
Peng et al.	44.1	-	41.1		
Li & Rush	44.7	9.92	43.3		
KGPT	45.1	-	-		

Table 1: Standard automatic evaluation metrics for the E2E dataset (top) and WikiBio dataset (bottom). Baselines include our own transformer sequence-to-sequence-with-copy model ("S2S+copy"), and the models of Li and Rush (2020), Peng et al. (2019), and Chen et al. (2020, "KGPT").

E2E	Natural	Faithful	Informative
FULL	3.87	3.97	3.89
LRT	3.75	3.94	3.94
S2S+copy	3.87	3.94	3.81
WB			
FULL	3.69	3.52	3.27
LRT	3.83	3.74	3.37
S2S+copy	3.75	3.75	3.40

Table 2: Average rating (on 1-5 Likert scale) of generations' naturalness, faithfulness, and informativeness, according to crowd-workers. No pairwise differences are significant under a Tukey HSD test.

5.1 Quality Evaluation

We first evaluate our models and baselines using the standard automatic metrics associated with each dataset, including BLEU (Papineni et al., 2002), NIST, ROUGE (Lin, 2004), CIDEr (Vedantam et al., 2015) and METEOR (Banerjee and Lavie, 2005), in Table 1. There we also compare with the model of Peng et al. (2019), which uses retrieved neighbors, and of Li and Rush (2020), which produces interpretable segmentations, as well as with the model of Chen et al. (2020) ("KGPT" in tables), which is a fine-tuned, large pretrained model, and which we take to be close to the state of the art.

We first note that our baselines are quite strong, largely outperforming previous work, including large pretrained models. In the case of E2E, we find that the FULL model slightly outperforms these strong baselines and attains, we believe, state-of-the-art performance in the setting where no pretrained models or data augmentation is used. (See Chang et al. (2021) for even better results without these restrictions). On WikiBio, however, FULL slightly underperforms the strongest baselines.

Human Evaluation In Table 2 we show the (average) results of a human evaluation conducted following the methodology described in Reiter (2017). We ask crowd-workers on Amazon Mechanical Turk to score generations in terms of their naturalness, their faithfulness to the source table, and their informativeness, on a 5-point Likert scale. (Note that following Dhingra et al. (2019) we ask about informativeness rather than usefulness). We score a total of 45 random examples from each test dataset, with each generation being rated by 3 crowd-workers, and each crowd-worker seeing a generation from each system. We ran multi-way ANOVAs with system-type (i.e., FULL, LRT, or S2S+copy), example index, and crowd-worker-id as independent variables, and the rating as the dependent variable.

The only significant interaction involving system-type was with respect to "faithfulness" on the WikiBio dataset ($p < 0.018$), though this does *not* reflect a necessary correction accounting for the multiple comparisons implied by crowd-workers rating along 3 dimensions. Furthermore, under Tukey's HSD test no significant pairwise (i.e., between system pairs) interactions were found in any setting. Thus, we find no significant difference between any system pairs according to crowd-workers, although (as with the automatic metrics) FULL performs slightly better on E2E and worse on WikiBio. We give the precise p-values as well as more details about the questions crowd-workers were asked in Appendix E.

We also conduct a manual analysis of the faithfulness errors made by FULL generations in Appendix B; we generally find that FULL generations do not hallucinate more than S2S+Copy generations (the most faithful generations according to crowd-workers), but they do more frequently contradict information in the source table. This generally occurs when a span containing information that contradicts the table is copied to the canvas, and this information is not subsequently replaced; see Appendix B for more details and a discussion.

5.2 Interpretability Evaluation

We first emphasize that, on an intuitive level, we believe FULL policies lead to significantly more interpretable generation than token-level policies. This is because FULL policies give an explicit (and often short) span-based derivation of a generated text in terms of neighbor text. We show visualizations of two randomly chosen generations from the WikiBio validation set, along with their derivations, in Figure 3.

However, it is difficult to precisely quantify the interpretability of a text generation model, and so we now quantify several aspects of our models' predictions that presumably correlate with their interpretability. Table 3 shows the average length of the derivations (i.e., how many insert operations are required to form a generation), the average number of neighbors used in forming a prediction, and the percentage of generated tokens copied from a neighbor or the source x (rather than generated from the model's output vocabulary) for FULL and LRT-no-marg policies over 500 randomly-chosen examples from the E2E and WikiBio validation-sets. All else equal, we expect fewer insert operations, fewer neighbors, and more tokens copied from neighbors (resp.) to correlate with better interpretability. We find that FULL generations require many fewer insert operations and distinct neighbors per generation on average than LRT generations, although they use their output-vocabulary slightly more than LRT-no-marg. Note that we use LRT-no-marg for this comparison because marginalization obscures whether a predicted token is from a neighbor.

The fact that FULL policies use so few distinct neighbors per example motivates asking how well these policies perform at test time when given fewer neighbors than they are trained with (namely, 20 in all experiments). We plot the average validation ROUGE of FULL and LRT for both datasets (using ROUGE-4 for WikiBio and ROUGE-L for E2E, as is conventional) against the number of neigh-

	FULL E2E / WB	LRT-no-marg E2E / WB
# Inserts	6.9 / 7.4	24.7 / 25.8
# Neighbors	1.1 / 1.4	5.5 / 5.6
% Tok. Copied	99.3 / 95.3	99.8 / 96.4

Table 3: Number of inserts, number of neighbors used, and percentage of generation tokens from a neighbor, averaged over 500 random examples from the E2E and Wikibio validation sets.

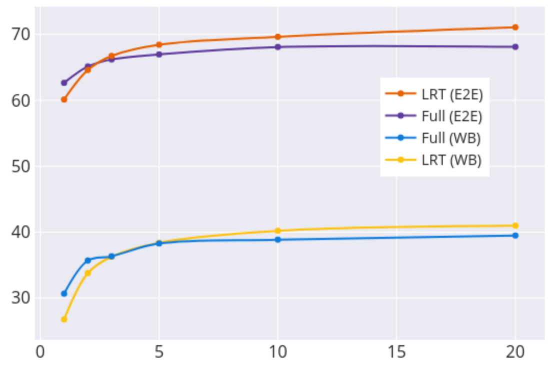


Figure 2: ROUGE validation performance on WikiBio and E2E, by number of neighbors used at test time.

bors used at generation time in Figure 2. We see that while using fewer neighbors hurts both types of policies, FULL outperforms LRT for very few neighbors.

Controllability Another approach to evaluating the interpretablility of a model is to use our understanding of the model's prediction process to control it, and then evaluate controllability. In Appendix C we describe, as a case study, attempting to control the number of sentences used in E2E dataset generations by controlling the neighbors; we find that FULL significantly outperforms LRT policies in ensuring that generations have at least three sentences.

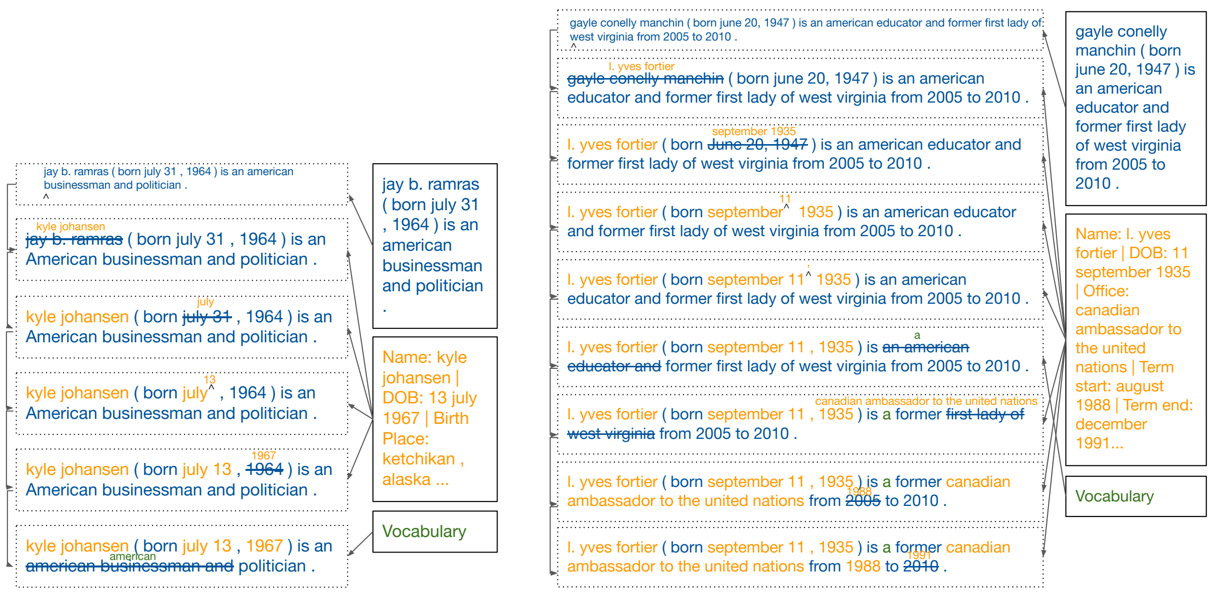
6 Related Work

NLP systems have incorporated neighbors for decades. Early work focused on machine translation (Sumita and Hitoshi, 1991), syntactic disambiguation (Cardie, 1994), and tagging (Daelemans, 1993; Daelemans et al., 1996).

While some more recent work has made use of retrieved neighbors for problems such as sequence labeling (Wiseman and Stratos, 2019), auditing multi-label text classification predictions (Schmaltz and Beam, 2020), and reasoning over knowledge bases (Das et al., 2020, 2021), the majority of recent NLP work involving neighbor-based methods has focused on conditioning neural text generation systems on retrieved neighbors. This conditioning is variously accomplished using a conventional encoder in an encoder-decoder setup (Song et al., 2016; Weston et al., 2018; Gu et al., 2018b; Cao and Xiong, 2018; Bapna and Firat, 2019), by allowing the parameters of the decoder to depend on the retrieved neighbor (Peng et al., 2019), or by viewing the unknown neighbor as a latent variable (Hashimoto et al., 2018; Guu et al., 2018; Chen et al., 2019; He et al., 2020). Recent work (Zhang et al., 2018; Khandelwal et al., 2019, 2020) has also used retrieved neighbors at decoding time to modify the next-token distribution of the decoder. Our work differs from these approaches in that we explicitly parameterize the splicing operations that form a generation from neighbors, rather than conditioning or otherwise modifying a left-to-right token generation model using retrieved neighbors.

Our parameterization is motivated by trying to increase the interpretability and controllability of the generation process, which also motivates recent work making explicit the template or plan being followed by the generation (Iyyer et al., 2018; Wiseman et al., 2018; Puduppully et al., 2019; Chen et al., 2019; Li and Rush, 2020, *inter alia*). This more structural or syntactic flavor of controllability differs slightly from foundational work on controlling content or stylistic attributes of text (Hu et al., 2017; Ficler and Goldberg, 2017; Fan et al., 2018).

Our approach is also related to work in non-left-to-right text generation, including tree-based (Welleck et al., 2019; Akoury et al., 2019), non-autoregressive (Gu et al., 2018a; Lee et al., 2018, *inter alia*), masked language model-based (Ghazvininejad et al., 2019, *inter alia*), and, most closely, insertion-based (Stern et al., 2019; Gu et al., 2019b,a, *inter alia*) approaches. Our work differs from this last category in several important respects: first, we insert and replace (and model) full spans rather than tokens. Our policies are trained to minimize the number of insertion operations rather than to insert (centrally positioned) correct tokens in available slots, as is Insertion Transformer (Stern et al., 2019), or to mimic a Levenshtein distance-based oracle, as is LevT (Gu et al., 2019b). Our policies are also fundamentally sequential, unlike these partially autoregressive alternatives, which can generate tokens in parallel. The sequential nature of our approach makes us-


jay b. ramras (born july 31 , 1964) is an american businessman and politician . ^

kyle johansen ~~jay b. ramras~~ (born july 31 , 1964) is an American businessman and politician .

kyle johansen (born july ~~july 31~~ , 1964) is an American businessman and politician .

kyle johansen (born july ^ 13 , 1964) is an American businessman and politician .

kyle johansen (born july 13 , 1967 ~~1964~~) is an American businessman and politician .

kyle johansen (born july 13 , 1967) is an american ~~american businessman and~~ politician .

jay b. ramras (born july 31 , 1964) is an american businessman and politician .

Name: kyle johansen | DOB: 13 july 1967 | Birth Place: ketchikan , alaska ...

Vocabulary

gayle conelly manchin (born june 20, 1947) is an american educator and former first lady of west virginia from 2005 to 2010 . ^

l. yves fortier ~~gayle conelly manchin~~ (born june 20, 1947) is an american educator and former first lady of west virginia from 2005 to 2010 .

l. yves fortier (born september 1935 ~~June 20, 1947~~) is an american educator and former first lady of west virginia from 2005 to 2010 .

l. yves fortier (born september ^ 11 1935) is an american educator and former first lady of west virginia from 2005 to 2010 .

l. yves fortier (born september 11 ^ , 1935) is an american educator and former first lady of west virginia from 2005 to 2010 .

l. yves fortier (born september 11 , 1935) is a ~~an american educator and~~ former first lady of west virginia from 2005 to 2010 .

l. yves fortier (born september 11 , 1935) is a former canadian ambassador to the united nations ~~first lady of west virginia~~ from 2005 to 2010 .

l. yves fortier (born september 11 , 1935) is a former canadian ambassador to the united nations from 1988 ~~2005~~ to 2010 .

l. yves fortier (born september 11 , 1935) is a former canadian ambassador to the united nations from 1988 to 1991 ~~2010~~ .

gayle conelly manchin (born june 20, 1947) is an american educator and former first lady of west virginia from 2005 to 2010 .

Name: l. yves fortier | DOB: 11 september 1935 | Office: canadian ambassador to the united nations | Term start: august 1988 | Term end: december 1991...

Vocabulary

Figure 3: A visualization of how two randomly selected model generations from the WikiBio validation set — *kyle johansen (born july 13, 1967) is an american politician.* and *l. yves fortier (born september 11, 1935) is a former canadian ambassador the the united nations from 1988 to 1991.* — are derived. In both cases only one non-table neighbor is used, and is depicted in the top-most solid box (in blue). The solid box second from the top (in orange) represents the linearized source table, $\nu^{(0)}$, and the third (in green) represents all word types in the vocabulary. Derivations should be read top-down; the current canvas is represented by a dotted box, carets indicate insertion, struck through text has been replaced, and straight arrows indicate the provenance of a span. Beginning- and end-of-sequence tokens are omitted.

ing beam search straightforward (unlike in token-parallel approaches) and, we think, leads to interpretable, serial derivations. On the other hand, decoding serially with beam search will generally be slower than the iterated parallel decoding of partially autoregressive models.

Our work also relates to recent work on sentence-level transduction tasks, like grammatical error correction (GEC), which allows for directly predicting certain span-level edits (Stahlberg and Kumar, 2020). These edits are different from our insertion operations, requiring token-level operations except when copying from the source sentence, and are obtained, following a long line of work in GEC (Swanson and Yamangil, 2012; Xue and Hwa, 2014; Felice et al., 2016; Bryant et al., 2017), by heuristically merging token-level alignments obtained with a Damerau-Levenshtein-style algorithm (Brill and Moore, 2000).

7 Conclusion

We have presented an approach to data-to-text generation, which directly splices together retrieved neighbors. We believe this line of work holds promise for improved interpretability and controllability of text generation systems.

In future work we hope to tackle more ambitious text generation tasks, which will likely require retrieving many more neighbors, perhaps dynamically, from larger data-stores, and with more sophisticated retrieval techniques, such as those currently being used in retrieval-based pretraining (Lewis et al., 2020a; Guu et al., 2020).

We also hope to consider more sophisticated models, which explicitly capture the history of produced canvases, and more sophisticated training approaches, which search for optimal insertions while training, rather than as a preprocessing step (Daumé III et al., 2009; Ross et al., 2011).

References

Nader Akoury, Kalpesh Krishna, and Mohit Iyyer. 2019. Syntactically supervised transformers for faster neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1269–1281.

James K Baker. 1979. Trainable grammars for speech recognition. *The Journal of the Acoustical Society of America*, 65(S1):S132–S132.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings* *of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Ankur Bapna and Orhan Firat. 2019. Non-parametric adaptation for neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1921–1931.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179.

Eric Brill and Robert C Moore. 2000. An improved error model for noisy channel spelling correction. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 286–293. Association for Computational Linguistics.

Christopher Bryant, Mariano Felice, and Ted Briscoe. 2017. Automatic annotation and evaluation of error types for grammatical error correction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 793–805, Vancouver, Canada. Association for Computational Linguistics.

Qian Cao and Deyi Xiong. 2018. Encoding gated translation memory into neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3042–3047, Brussels, Belgium. Association for Computational Linguistics.

Claire Cardie. 1994. Domain-specific knowledge acquisition for conceptual sentence analysis. *Computer Science Department Faculty Publication Series*, page 60.

Ernie Chang, Xiaoyu Shen, Dawei Zhu, Vera Demberg, and Hui Su. 2021. Neural data-to-text generation with LM-based text augmentation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 758–768, Online. Association for Computational Linguistics.

Mingda Chen, Qingming Tang, Sam Wiseman, and Kevin Gimpel. 2019. Controllable paraphrase generation with a syntactic exemplar. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5972–5984.

Wenhu Chen, Yu Su, Xifeng Yan, and William Yang Wang. 2020. KGPT: Knowledge-grounded pretraining for data-to-text generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8635–8648, Online. Association for Computational Linguistics.

KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*.

Noam Chomsky. 1959. On certain formal properties of grammars. *Information and control*, 2(2):137–167.

Walter Daelemans. 1993. Memory-based lexical acquisition and processing. In *Workshop on Machine Translation and Lexicon*, pages 85–98. Springer.

Walter Daelemans, Jakob Zavrel, Peter Berck, and Steven Gillis. 1996. Mbt: A memory-based part of speech tagger-generator. In *Fourth Workshop on Very Large Corpora*.

Rajarshi Das, Ameya Godbole, Shehzaad Dhuliawala, Manzil Zaheer, and Andrew McCallum. 2020. A simple approach to case-based reasoning in knowledge bases. In *Automated Knowledge Base Construction*.

Rajarshi Das, Manzil Zaheer, Dung Thai, Ameya Godbole, Ethan Perez, Jay-Yoon Lee, Lizhen Tan, Lazaros Polymenakos, and Andrew McCallum. 2021. Case-based reasoning for natural language queries over knowledge bases. *arXiv preprint arXiv:2104.08762*.

Hal Daumé III, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine Learning*, 75(3):297–325.

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

Bhuwan Dhingra, Manaal Faruqui, Ankur Parikh, Ming-Wei Chang, Dipanjan Das, and William Cohen. 2019. Handling divergent reference texts when evaluating table-to-text generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4884–4895.

Angela Fan, David Grangier, and Michael Auli. 2018. Controllable abstractive summarization. *ACL 2018*, page 45.

Mariano Felice, Christopher Bryant, and Ted Briscoe. 2016. Automatic extraction of learner errors in ESL sentences using linguistically enhanced alignments. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 825–835, Osaka, Japan. The COLING 2016 Organizing Committee.

Jessica Ficler and Yoav Goldberg. 2017. Controlling linguistic style aspects in neural language generation. *EMNLP 2017*, page 94.

Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A Smith. 2020. Realtoxicityprompts: Evaluating neural toxic degeneration in language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 3356–3369.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Mask-predict: Parallel decoding of conditional masked language models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6111–6120, Hong Kong, China. Association for Computational Linguistics.

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

Jiatao Gu, Qi Liu, and Kyunghyun Cho. 2019a. Insertion-based decoding with automatically inferred generation order. *Transactions of the Association for Computational Linguistics*, 7:661–676.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *ACL*.

Jiatao Gu, Changhan Wang, and Junbo Zhao. 2019b. Levenshtein transformer. In *Advances in Neural Information Processing Systems*, pages 11181–11191.

Jiatao Gu, Yong Wang, Kyunghyun Cho, and Victor OK Li. 2018b. Search engine guided neural machine translation. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Kelvin Guu, Tatsunori B Hashimoto, Yonatan Oren, and Percy Liang. 2018. Generating sentences by editing prototypes. *Transactions of the Association of Computational Linguistics*, 6:437–450.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: Retrieval-augmented language model pre-training. *arXiv preprint arXiv:2002.08909*.

Tatsunori B Hashimoto, Kelvin Guu, Yonatan Oren, and Percy S Liang. 2018. A retrieve-and-edit framework for predicting structured outputs. In *Advances in Neural Information Processing Systems*, pages 10052–10062.

He He, Jason Eisner, and Hal Daume. 2012. Imitation learning by coaching. *Advances in Neural Information Processing Systems*, 25:3149–3157.

Junxian He, Taylor Berg-Kirkpatrick, and Graham Neubig. 2020. Learning sparse prototypes for text generation. *Advances in Neural Information Processing Systems*, 33.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In *International Conference on Machine Learning*, pages 1587–1596.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1875–1885.

Tadao Kasami. 1966. An efficient recognition and syntax-analysis algorithm for context-free languages. *Coordinated Science Laboratory Report no. R-257*.

Urvashi Khandelwal, Angela Fan, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Nearest neighbor machine translation. *arXiv preprint arXiv:2010.00710*.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2019. Generalization through memorization: Nearest neighbor language models. In *International Conference on Learning Representations*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *EMNLP*, pages 1203–1213.

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

Mike Lewis, Marjan Ghazvininejad, Gargi Ghosh, Armen Aghajanyan, Sida Wang, and Luke Zettlemoyer. 2020a. Pre-training via paraphrasing. *Advances in Neural Information Processing Systems*, 33.

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

Xiang Lisa Li and Alexander Rush. 2020. Posterior control of blackbox generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2731–2743, Online. Association for Computational Linguistics.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Shuai Lin, Wentao Wang, Zichao Yang, Xiaodan Liang, Frank F Xu, Eric Xing, and Zhiting Hu. 2020. Record-to-text generation with style imitation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 1589–1598.

Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Jekaterina Novikova, Ondrej Dušek, and Verena Rieser. 2017. The E2E dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, Saarbrücken, Germany.

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.

Hao Peng, Ankur Parikh, Manaal Faruqui, Bhuwan Dhingra, and Dipanjan Das. 2019. Text generation with exemplar-based adaptive decoding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2555–2565.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2019. Data-to-text generation with content selection and planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6908–6915.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *ICLR*.

Ehud Reiter. 2017. You need to understand your corpora! the weathergov example. https://ehudreiter.com/2017/05/09/weathergov/.

Stéphane Ross, Geoffrey J. Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no -regret online learning. In *Proceedings of the Fourteenth International Conference on Artific ial Intelligence and Statistics*, pages 627–635.

Arto Salomaa. 1969. Probabilistic and weighted grammars. *Information and Control*, 15(6):529–544.

Allen Schmaltz and Andrew Beam. 2020. Exemplar auditing for multi-label biomedical text classification. *arXiv preprint arXiv:2004.03093*.

Emily Sheng, Kai-Wei Chang, Premkumar Natarajan, and Nanyun Peng. 2019. The woman worked as a babysitter: On biases in language generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3407–3412, Hong Kong, China. Association for Computational Linguistics.

Yiping Song, Rui Yan, Xiang Li, Dongyan Zhao, and Ming Zhang. 2016. Two are better than one: An ensemble of retrieval-and generation-based dialog systems. *arXiv preprint arXiv:1610.07149*.

Felix Stahlberg and Shankar Kumar. 2020. Seq2edits: Sequence transduction using span-level edit operations. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5147–5159.

Mitchell Stern, William Chan, Jamie Kiros, and Jakob Uszkoreit. 2019. Insertion transformer: Flexible sequence generation via insertion operations. In *International Conference on Machine Learning*, pages 5976–5985.

Eiichiro Sumita and HDA Hitoshi. 1991. Experiments and prospects of example-based machine translation. In *29th Annual Meeting of the Association for Computational Linguistics*, pages 185–192.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3104–3112.

Ben Swanson and Elif Yamangil. 2012. Correction detection and error type selection as an ESL educational aid. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 357–361.

Zhaopeng Tu, Yang Liu, Lifeng Shang, Xiaohua Liu, and Hang Li. 2017. Neural machine translation with reconstruction. In *AAAI*, pages 3097–3103.

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.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing NLP. In *Proceedings of the 2019 Conference on Empirical Methods* *in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2153–2162, Hong Kong, China. Association for Computational Linguistics.

Sean Welleck, Kianté Brantley, Hal Daumé III, and Kyunghyun Cho. 2019. Non-monotonic sequential text generation. In *International Conference on Machine Learning*, pages 6716–6726.

Jason Weston, Emily Dinan, and Alexander H Miller. 2018. Retrieve and refine: Improved sequence generation models for dialogue. *arXiv preprint arXiv:1808.04776*.

Ronald J Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280.

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

Sam Wiseman and Karl Stratos. 2019. Label-agnostic sequence labeling by copying nearest neighbors. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5363–5369.

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

Huichao Xue and Rebecca Hwa. 2014. Improved correction detection in revised ESL sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 599–604.

Daniel H Younger. 1967. Recognition and parsing of context-free languages in time n^3. *Information and control*, 10(2):189–208.

Jingyi Zhang, Masao Utiyama, Eiichiro Sumita, Graham Neubig, and Satoshi Nakamura. 2018. Guiding neural machine translation with retrieved translation pieces. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1325–1335.

A Proof of Claim 1

Given sequences $\nu^{(1)}, \ldots, \nu^{(N)}$ and a target sequence y, let C be the minimum integer such that there exist sequences $y^{(0)}, \ldots, y^{(C)}$ with $y^{(0)} = \varepsilon$ being the empty sequence, $y^{(C)} = y$, and $y^{(c)} = \text{insert}(y^{(c-1)}, i, j, n, k, l)$ for $c = 1, \ldots, C$, where $M = |y^{(c-1)}|$, $0 \leq i < j \leq M+1$, and $1 \leq k \leq l \leq |\nu^{(n)}|$. Let $\text{cost}_{\text{ins}}(y)$ be this minimum C, equivalent to the length of the shortest derivation of y with actions in $\mathcal{A}_{\text{ins}}$, and let $\text{cost}_{\text{CFG}}(y)$ be the cost of the minimum cost derivation of y with the WCFG in Section 3.1.1. We want to show that $\text{cost}_{\text{ins}}(y) = \text{cost}_{\text{CFG}}(y)$, which we accomplish by showing that $\text{cost}_{\text{ins}}(y) \leq \text{cost}_{\text{CFG}}(y)$ and that $\text{cost}_{\text{ins}}(y) \geq \text{cost}_{\text{CFG}}(y)$.

Proposition 1. *$cost_{ins}(y) \leq cost_{CFG}(y)$.*

Proof. Consider the minimum cost derivation tree of y under the WCFG in Section 3.1.1, and let C be the minimum cost. We show inductively that there exists a sequence of $\leq C$ insert operations that yields y from ε, by considering two cases.

Case 1: y is derived using only the first two grammar rules, and so is a concatenation of sequences $\nu_{k:l}^{(n)}$ derived from the first grammar rule. If there are C such sequences, the WCFG derivation costs C. Also, constructing this sequence using insertions requires at most C insertions.

Case 2: y is derived using at least one application of the third grammar rule and can be written as $y = s^{(1)} \cdot y^{(1)} \cdot s^{(2)} \cdot y^{(2)} \cdot \ldots \cdot y^{(Q)} \cdot s^{(Q+1)}$, where the $s^{(q)}$ sequences are all from the same $\nu^{(n)}$ (using the last three grammar rules), and the $y^{(q)}$ sequences are derived from an S nonterminal. We can derive y using insert operations by inserting a substring of $\nu^{(n)}$ containing all the $s^{(q)}$ (which costs 1 under the grammar), then inserting the remaining $y^{(q)}$ recursively, which, by induction, costs at most $\text{cost}_{\text{CFG}}(y^{(q)})$. The total number of insertions is then at most $\text{cost}_{\text{CFG}}(y)$. □

Proposition 2. *$cost_{ins}(y) \geq cost_{CFG}(y)$.*

Proof. Let $\tilde{y}^{(0)} = \varepsilon, \tilde{y}^{(1)}, \ldots, \tilde{y}^{(C)}$ be sequences of integers defined as follows:

$$\tilde{y}^{(c)} = \tilde{y}_{1:i}^{(c-1)}, \underbrace{c, c \ldots c, c}_{|\nu_{k:l}^{(n)}|}, \tilde{y}_{j:M}^{(c-1)}.$$

That is, instead of inserting sequence $\nu_{k:l}^{(n)}$, we insert a sequence of $|\nu_{k:l}^{(n)}|$ integers c. We call a sequence z of integers non-interleaving if there is no $i < j < k < l$ such that $z_i = z_k$ and $z_j = z_l$.

Observation 1. *The sequences $\tilde{y}^{(0)}, \ldots, \tilde{y}^{(C)}$ are non-interleaving.*

Proof. By induction: if $\tilde{y}^{(c)}$ is non-interleaving, then so is $\tilde{y}^{(c+1)}$ by its construction from $\tilde{y}^{(c)}$. □

Now let $y' = \tilde{y}^{(C)}$ to simplify notation. Let $\text{distinct}(y') \leq C$ be the number of distinct integers in the sequence y'. We show how to derive y from the grammar with derivation cost $\text{distinct}(y')$, which proves the proposition. We call an integer c' contiguous in y' if y' can be written as $y' = y'', c', c', \ldots, c', c', y'''$ such that sequences y'' and y''' do not contain c'. We derive y from the grammar inductively, by considering two cases.

Case 1: all integers in y' are contiguous, so y' consists of contiguous blocks of repeated integers. Let b be the number of blocks. We invoke the second grammar rule $b-1$ times to get S repeated b times and then invoke the first grammar rule for each S to derive the contiguous sequence $\nu_{k:l}^{(n)}$ corresponding to the block. This costs $\text{distinct}(y') = b$ in total as required.

Case 2: there is an integer in y' that is not contiguous. Let c' be the left-most non-contiguous integer in y'. c' splits y' into several shorter sequences $y'^{(1)}, \ldots, y'^{(Q)}$, where each sequence $y'^{(q)}$ does not contain any copy of integer c'. Since y' is non-interleaving (by Observation 1), $y'^{(q)}$ and $y'^{(q')}$ do not share any integers for $q \neq q'$. Therefore, $\text{distinct}(y') = 1 + \text{distinct}(y'^{(1)}) + \ldots + \text{distinct}(y'^{(Q)})$. Furthermore, each sequence $y'^{(q)}$ is non-interleaving. Therefore, we can derive the subsequence of $\tilde{y}^{(C)}$ corresponding to $y'^{(q)}$ from the non-terminal S and it costs $\text{distinct}(y'^{(q)})$ by induction. To finish the proof we need to show that we can combine the resulting sequences into the sequence corresponding to y' by paying an additional cost of only 1. We can do that by using the last three rules of the grammar where the rule of cost 1 is applied only once. In particular, we pay 1 to derive the sequence corresponding to the first block of integers c' in y'. The sequence is derived from $Y_{k:l}^{(n)}$, which comes from the rule $S \rightarrow Y_{k:l}^{(n)}\ C_s^{(n)}$ and the application of this rule costs 1. The rest of the blocks of c' are derived from the last three rules and they cost 0. □

B Manual Analysis of WikiBio Errors

In Table 4 we analyze the faithfulness errors of the FULL policies on 50 random test examples from

	FULL	S2S+Copy
Hallucination	6	5
Explicit contradiction	6	1
Implicit contradiction	3	0

Table 4: Manual categorization of faithfulness errors made by FULL and S2S+Copy models on 50 random examples from the WikiBio test set.

the WikiBio dataset, comparing them to the generations of the S2S+Copy model. We divide the errors into hallucination errors, where the model invents facts neither supported nor contradicted by the table, explicit contradiction errors, where the model explicitly contradicts information in the table, and implicit contradiction errors, where the model contradicts information that is only implicit in the table.

We find that the FULL model hallucinates at approximately the same rate as does S2S+Copy. However, it also generates more explicit contradictions. These tend to occur when a span containing contradictory information is copied to the canvas, but is not subsequently edited. We suspect that incorporating additional losses, such as a round-trip reconstruction loss (Tu et al., 2017) will be helpful here. It is notable that the FULL generations struggle with implicit contradiction more than S2S+Copy. Some examples of implicit contradiction we observed include when a person's gender or nationality are strongly suggested by their name, or their area of study by their thesis title, despite this information not being explicit in the table. We suspect that bigger models, especially if they store state in addition to the canvas (which our FULL models do not), will better address these cases.

C Controllability Case Study

We briefly consider a case-study that exemplifies controlling generation by controlling the neighbors used at test time. We consider in particular a situation where control is much more easily accomplished under FULL policies than token-level policies.

Some examples in the E2E dataset consist of only a single sentence (e.g., "The Golden Curry is a non family friendly Indian restaurant with an average rating located in the riverside area near Cafe Rouge."), while others split the description into multiple sentences. We consider requiring the generated text to consist of at least 3 sentences, which is interesting and challenging for two reasons. First, only about 8% of the training examples have ≥ 3 sentences. Second, while it is sometimes possible to force the generations of token-level models to obey structural constraints by constraining beam search (e.g., by disallowing certain tokens depending on the context), a constrained beam search does not make it easy to guarantee the *presence* of certain structural features. Specifically, while it is easy to constrain beam search so that hypotheses with too many sentences are kept off the beam, it is unclear how to ensure beam search finds only (or even any) hypotheses with enough sentences.

We accordingly restrict both the FULL and LRT models to use only neighbors with ≥ 3 sentences (as determined by a regular expression) when generating on the E2E development set. We find that 87.2% of the resulting FULL generations have ≥ 3 sentences, while only 73.5% of the LRT generations do. Furthermore, the quality of the resulting text remains high, with a ROUGE score of 67.6 for the FULL generations and 71.7 for LRT. (Note this comparison unfairly favors LRT, which generates many fewer of the rare ≥ 3 sentence generations).

When the FULL model fails to respect the constraint it is because it has inserted text that replaces the end of a sentence (or two). We can reach 100% constraint satisfaction by simply constraining the FULL model's beam search to never replace a full sentence in the canvas. As noted above, we *cannot* easily constrain the LRT beam search to reach 100% constraint satisfaction.

D Additional Model and Training Details

Our models are BART (Lewis et al., 2020b)-style encoder-decoder transformers. They consume embeddings of the linearized source tokens x and the current canvas $\hat{y}_{1:M}$ (plus positional embeddings). To allow the model to capture how recently tokens were added to the canvas, we add to each canvas token embedding an embedding of a feature indicating how many time-steps have elapsed since it was added. We also add to each x token embedding the embedding of an indicator feature indicating whether it has been copied to $\hat{y}_{1:M}$. We obtain neighbor embeddings by putting neighbor token embeddings plus positional embeddings plus the embedding of an indicator feature indicating that these are neighbor tokens through the same encoder that consumes x.

All transformer encoder-decoders have 6 lay-

learning rate	{1e-4, 3e-4, 5e-4, 1e-3}
β_1	{0.85, 0.9, 0.95 }
β_2	{0.9, 0.99, 0.999 }
ε	{1e-6, 1e-7, 1e-8 }
weight decay	{0, 1e-3, 1e-2}
beam width	{1, 5, 10, 20}

Table 5: Hyperparameter bounds.

	LR	β_1, β_2	ε	WD	BW
E2E-FULL	1e-3	0.9, 0.999	1e-7	1e-3	5
E2E-LRT	5e-4	0.9, 0.999	1e-7	1e-3	5
E2E-S2S+copy	5e-4	0.9, 0.999	1e-7	1e-3	5
WB-FULL	5e-4	0.9, 0.999	1e-7	1e-3	10
WB-LRT	5e-4	0.9, 0.999	1e-7	1e-3	10
WB-S2S+copy	3e-4	0.9, 0.999	1e-7	1e-3	20

Table 6: Final hyperparameters used. "LR", "WD", and "BW" are learning rate, weight decay, and beam width, respectively.

ers, with model dimension 420, feed-forward dimension 650, 7 attention heads, and dropout rate 0.1. These hyperparameters were chosen (and then fixed) so as to allow the largest model that could be trained in a reasonable amount of time on our GTX 1080 Ti and RTX 2080 Ti GPUs.

We trained with Adam (Kingma and Ba, 2015). We linearly warm-up the learning rate during the first 4,000 training steps, and then use square-root learning rate decay as in Devlin et al. (2019) after warm-up. To stabilize training we accumulate gradients over 400 target sequences.

We show the training and prediction hyperparameter bounds we considered in Table 5. We selected combinations at random for 50 1-epoch trials for each model, evaluating on validation negative log-likelihood. Our final hyperparameter values are in Table 6.

E Human Evaluation Details

We selected 45 random examples from each of the E2E and WikioBio test-sets for use as crowd-worker prompts. Each example was rated by 3 crowd-workers, and each crowd-worker rated each of the 3 systems. We excluded the 11 responses that did not provide all 9 ratings (3 ratings for each of 3 examples). We show a screen-shot of the questions asked of Mechanical Turk crowd-workers, given a table and generated description, in Figure 4. We show results of significance tests in Table 7 and 8.

	Pr(>F)
faithfulness	0.9648
naturalness	0.4626
informativeness	0.4412
faithfulness	0.0182
naturalness	0.4689
informativeness	0.4360

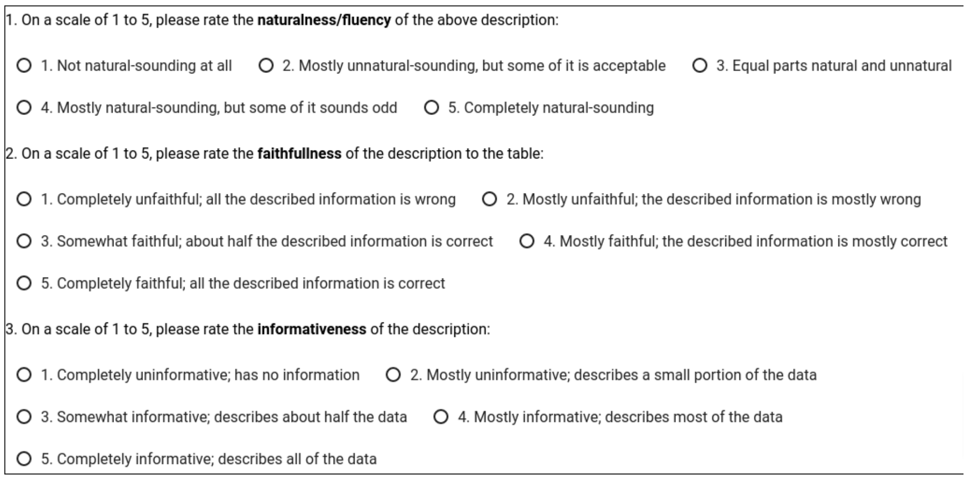
Table 7: System-type p-values under ANOVA for E2E (top) and Wikibio (bottom).

	p	CI
LRT-S2S-faithfulness	0.9	(-0.3295, 0.3295)
LRT-FULL-faithfulness	0.9	(-0.3056, 0.3533)
FULL-S2S-faithfulness	0.9	(-0.3056, 0.3533)
LRT-S2S-naturalness	0.6882	(-0.2325, 0.4706)
LRT-FULL-naturalness	0.6882	(-0.2325, 0.4706)
FULL-S2S-naturalness	0.9	(-0.3516, 0.3516)
LRT-S2S-informativeness	0.6474	(-0.4712, 0.2172)
LRT-FULL-informativeness	0.9	(-0.3918, 0.2965)
FULL-S2S-informativeness	0.8337	(-0.2648, 0.4235)
LRT-S2S-faithfulness	0.9	(-0.3243, 0.3544)
LRT-FULL-faithfulness	0.2869	(-0.5574, 0.1213)
FULL-S2S-faithfulness	0.2403	(-0.5724, 0.1062)
LRT-S2S-naturalness	0.8601	(-0.4315, 0.2812)
LRT-FULL-naturalness	0.6329	(-0.4917, 0.221)
FULL-S2S-naturalness	0.9	(-0.4165, 0.2962)
LRT-S2S-informativeness	0.9	(-0.297, 0.3572)
LRT-FULL-informativeness	0.7419	(-0.4249, 0.2294)
FULL-S2S-informativeness	0.6181	(-0.4549, 0.1993)

Table 8: p-value and 95% confidence intervals under Tukey HSD test (for pairwise difference of means) on E2E (top) and Wikibio (bottom).

F Sample Generations

We provide 5 random generations from each of FULL, LRT, and S2S+copy on the E2E and WikiBio test-sets in Tables 9 and 10.


1. On a scale of 1 to 5, please rate the **naturalness/fluency** of the above description:

○ 1. Not natural-sounding at all ○ 2. Mostly unnatural-sounding, but some of it is acceptable ○ 3. Equal parts natural and unnatural

○ 4. Mostly natural-sounding, but some of it sounds odd ○ 5. Completely natural-sounding

2. On a scale of 1 to 5, please rate the **faithfullness** of the description to the table:

○ 1. Completely unfaithful; all the described information is wrong ○ 2. Mostly unfaithful; the described information is mostly wrong

○ 3. Somewhat faithful; about half the described information is correct ○ 4. Mostly faithful; the described information is mostly correct

○ 5. Completely faithful; all the described information is correct

3. On a scale of 1 to 5, please rate the **informativeness** of the description:

○ 1. Completely uninformative; has no information ○ 2. Mostly uninformative; describes a small portion of the data

○ 3. Somewhat informative; describes about half the data ○ 4. Mostly informative; describes most of the data

○ 5. Completely informative; describes all of the data

Figure 4: Questions asked of Mechanical Turk crowd-workers, given a table and generated description.

The Blue Spice coffee shop is based near Crowne Plaza Hotel and has a high customer rating of 5 out of 5. The Cocum is a pub near Burger King. It has a high customer rating. Cocum is a pub near The Sorrento. The Giraffe is a restaurant near Rainbow Vegetarian Café in the city centre which serves Fast food. It is not family-friendly. The Cricketers is a family friendly coffee shop near Ranch. It has a low customer rating.
Blue Spice is a coffee shop near Crowne Plaza Hotel. It has a customer rating of 5 out of 5. Cocum pub has a high customer rating and is located near Burger King. Cocum is a pub near The Sorrento. Giraffe is a fast food restaurant near Rainbow Vegetarian Café in the city centre. It is not family-friendly. The Cricketers is a family friendly coffee shop near Ranch with a low customer rating.
Blue Spice is a coffee shop near Crowne Plaza Hotel. It has a customer rating of 5 out of 5. Cocum is a highly rated pub near Burger King. Cocum is a pub near The Sorrento. Giraffe is a fast food restaurant located in the city centre near Rainbow Vegetarian Café. It is not family-friendly. The Cricketers is a family friendly coffee shop near Ranch with a low customer rating.

Table 9: Random E2E samples from FULL (top), LRT (middle), and S2S+copy (bottom)
.

1. morarji desai (29 february 1896 – 10 april 1995) was a premiership - winning indian civil servant who served as prime minister of india between 1977 and 1979 . 2. charles casali (27 april 1923 – 8 january 2014) was a swiss football midfielder who played for switzerland in the 1954 fifa world cup . 3. jorgen thalbitzer (22 may 1920 – 29 march 1943) was a flying officer of the royal air force during world war ii . 4. jacob joseph “ jack ” lew (born august 29 , 1955) is an american diplomat . 5. dietrich - siegwart konrad friedrich fürchtegott von bonin (2 february 1917 – 15 april 1970) was a highly decorated rittmeister der reserves in the wehrmacht during world war ii .
1. morarji desai (29 february 1896 – 10 april 1995) was prime minister of india between 1977 and 1979 . 2. charles casali (27 april 1923 – 8 january 2014) was a swiss football midfielder who played for switzerland in the 1956 fifa world cup . 3. jorgen thalbitzer (22 may 1920 – 29 march 1943) was a danish flying ace during world war ii . 4. jacob joseph “ jack ” lew (born august 29 , 1955) is the 76th united states secretary of the treasury . 5. dietrich - siegwart konrad friedrich fürchtegott von bonin (2 february 1917 – 15 april 1970) was a highly decorated rittmeister der reserves in the wehrmacht during world war ii .
1. morarji desai (29 february 1896 – 10 april 1995) was the prime minister of india from 24 march 1977 to 15 july 1979 . 2. charles casali (27 april 1923 – 8 january 2014) was a swiss football midfielder who played for switzerland in the 1950 fifa world cup . 3. jorgen thalbitzer (22 may 1920 – 29 march 1943) was a danish flying ace of world war ii . 4. jacob joseph “ jack ” lew (born august 29 , 1955) is the 76th united states secretary of state for management and budget . 5. dietrich - siegwart konrad friedrich fürchtegott von bonin (2 february 1917 – 15 april 1970) was a highly decorated rittmeister der reserves in the wehrmacht during world war ii .

Table 10: Random WikiBio samples from FULL (top), LRT (middle), and S2S+copy (bottom).