Tailor: Generating and Perturbing Text with Semantic Controls

Anonymous ACL submission

Abstract

Controlled text perturbation is useful for evaluating model generalizability and improving model robustness to dataset artifacts. However, current techniques rely on training a perturbation model for every targeted attribute, which is expensive and hard to generalize. We present Tailor, a semantically-controlled text generation system. Tailor builds on a pretrained seq2seq model, and produces textual outputs conditioning on **control codes** derived from semantic representations. We craft a set of operations to modify the control codes, which in turn steer generation towards targeted attributes. These operations can be further composed into higher-level ones, allowing for flexible perturbation strategies. Tailor can be applied in various scenarios. We use it to automatically create high-quality contrast sets for four distinct natural language processing (NLP) tasks. These contrast sets contain fewer spurious biases and are complementary to manually annotated ones in terms of lexical diversity. We show that Tailor helps improve model generalization through data augmentation, with a 5.8-point gain on an NLI challenge set, by perturbing just ~2% of training data.

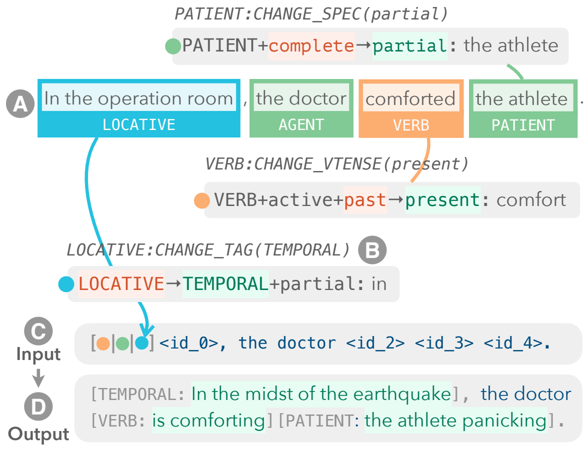

Figure 1: A compositional perturbation using Tailor.[1] Given (A) an original sentence, we abstract each span into a structured header that contains its semantic roles and keywords. We specify desired perturbations by modifying each control code (*e.g.,* changing role LOCATIVE→TEMPORAL in (B), verb tense past→present, and patient keyword specificity complete→partial). Given these *perturbed control codes* in the input (C), Tailor generates a new sentence (D) that reflects the desired perturbations.

1 Introduction

Controllable text generation through semantic perturbations modifies sentences to match certain target attributes, such as verb tense or sentiment (*e.g., positive→negative*). It has been widely applied to a variety of tasks, *e.g.,* style transfer (Reid and Zhong, 2021), mitigating dataset biases (Gardner et al., 2021), explaining model behaviors (Ross et al., 2020), and improving model generalization (Teney et al., 2020; Wu et al., 2021). Existing efforts train task-specific generators, *e.g.,* training a sentiment style transferer requires instances annotated with *positive* and *negative* labels (Madaan et al., 2020b). As a result, costly annotated data and re-training are required for every task of interest.

This work introduces Tailor, a system that supports application-agnostic perturbations. At its core is a *controlled generator* (§2) that flexibly generates outputs from target semantic attributes. We combine structured **control codes** with the **inputs** to represent desired linguistic properties of outputs. As shown in Figure 1, each code builds on the PropBank semantic analysis (Palmer et al., 2005) of the original sentence, and specifies an argument span and its semantic role. To encourage control code following, we train with **unlikelihood training** (Welleck et al., 2020) and penalize generations that are not aligned with designated codes.

The use of semantic roles allows Tailor to perform fine-grained changes to individual arguments in a sentence (*e.g.,* one can just change the patient

[1]We opensource Tailor at [URL omitted].

<table>
<tr><th></th><th>Input</th><th>Target Output</th><th>Description</th></tr>
<tr><td>A</td><td>[VERB+active+past: comfort | AGENT+complete: the doctor | PATIENT+partial: athlete | LOCATIVE+partial: in] <id_0>, <id_1> <id_2> <id_3>.</td><td>[LOCATIVE: In the operating room], [AGENT: the doctor] [VERB: comforted] [PATIENT: the athlete].</td><td>Mask all roles</td></tr>
<tr><td>B</td><td>[VERB+active+past: comfort | LOCATIVE+partial: in] <id_0>, the doctor <id_1> <id_2> the athlete <id_3>.</td><td>[LOCATIVE: In the operating room], the doctor [VERB: comforted] the athlete.</td><td>Empty blanks</td></tr>
<tr><td>C</td><td>[VERB+active+past: comfort | LOCATIVE+partial: in] <id_0>, the doctor <id_1> the athlete.</td><td>[LOCATIVE: In the operating room], the doctor comforted the athlete.</td><td>Mask subset of arguments</td></tr>
<tr><td>N</td><td>[VERB+passive+present: comfort | PATIENT+complete: the doctor | AGENT+partial: athlete | TEMPORAL+partial: in] <id_0>, <id_1> <id_2> <id_3>.</td><td>[TEMPORAL: In the operating room], [PATIENT: the doctor] [VERB: comforted] [AGENT: the athlete].</td><td>Negative sample</td></tr>
</table>

Table 1: Example input/output formats for sentence "In the operating room, the doctor comforted the athlete." A–C show different input formats the generator accepts, each with a *header* containing control codes and *context* with blanks denoting where to insert new texts. The last input (N) is a *negative* sample for unlikelihood training.

in Figure 1). This is critical for generating datasets to evaluate and improve models' language understanding (Kaushik et al., 2020; Wu et al., 2021). Instead of relying on a single target property *positive*↛*negative*, we can decompose it into specific linguistic transformations (*e.g.,* changing sentiment through negation or antonym replacement).

To highlight perturbations that TAILOR facilitates, we craft a list of primitive *perturbation operations* (§3) on inputs to the generator; these can be easily composed to achieve more complex perturbations. In Figure 1, TAILOR transforms sentence A to D through a series of perturbations: syntactic rewriting (changing verb tense), then sentence expansion (extending "the athlete"), and finally data recombination (*i.e.,* generating new text that contains "in" but follows the TEMPORAL control). Compared to existing approaches that require training a separate model for every step or annotating a dataset that represents this transformation end-to-end, such compositions make TAILOR more cost-effective and generalizable. In fact, on nine fine-grained and compositional STYLEPTB perturbations (Lyu et al., 2021), TAILOR achieves performance compatible with task-specific baselines, and even outperforms them on five transfers (§F).

TAILOR's flexible control codes allow for broad, easily extendable applicability. We demonstrate its utility in evaluating and improving NLP model robustness, showing that TAILOR can help replicate existing **contrast sets** on four diverse tasks. By abstracting manual perturbation types in prior work into TAILOR strategies, we generalize the changes to larger datasets while saving manual annotation efforts. Our analysis suggests that these contrast sets not only have high rates of validity, but also reduce spurious biases in datasets. In addition, TAILOR-produced contrast sets complement human annotated ones in terms of lexical diversity: only ~10% of their unique tokens overlap with manually created contrast sets. We also explore TAILOR's utility in data augmentation. We find that augmenting training data with just a small portion of TAILOR perturbations (~2%) improves the **robustness** of natural language inference (NLI) models to inference heuristics, increasing performance on the HANS evaluation set by an average of 5.81 points (McCoy et al., 2019) and outperforming a previous syntactic augmentation method for NLI.

2 TAILOR's Controllable Generator

Here we provide an overview of the TAILOR generator. To allow for control over sentence meaning at varying granularity levels, we incorporate three types of controls outlined in We first outline three types of controls that allows for specifying sentence meanings at varying granularity (§2.1), and then explain how to embed them within **inputs** to the generator (§2.2). We train the generator to follow control codes with **unlikelihood training** (§2.3).

2.1 Three Types of Controls

We use the following three types of controls to specify the shallow semantics, the actual content, and the ordering of various phrases in a sentence.

Semantic roles to denote shallow semantics. We rely on the PropBank semantic formalism (Palmer et al., 2005), as it provides well-established representations of meanings that are generalizable across different predicates and languages (Hajič et al., 2009). It represents sentence meanings with predicate-argument structures. Predicates are usually evoked by verbs and reflect events (*what happened*), like "comforted" (Fig-

Type	Predicate control: `VERB+active+past`: comfort
Signals	**Primary predicate label** (Always `VERB`) **Lemma** (Any verb lemma) **Voice** (`active`, `passive`)[2] **Tense** (`past`, `present`, `future`)
Type	Argument control: `PATIENT+partial`: athlete
Signals	**Primary argument label** (`AGENT`, `PATIENT`, `TEMPORAL`, `LOCATIVE`, `MANNER`, `CAUSE`, etc.) **Content** (* symbol or any text) **Specificity** (`complete`, `partial`, `sparse`)

Table 2: Tailor's control codes. Primary controls build on predicate/argument labels, and others affect the form and content of generations (More in §A.1).

ure 1); whereas arguments, usually spans of tokens, realize thematic roles of the predicates, including *core* arguments such as *who* (*e.g.,* "the doctor") and *to whom* ("the athlete"), as well as *adjunct* ones like *where* ("In the operation room") and *how*.

Keywords for steering the generated content of actual predicates and arguments. The keywords can either be sparse (*e.g.,* adding a random temporal constraint), or fully specified (adding a fixed "in the midst of the earthquake"). As later shown in Table 3, such control is important for supporting different perturbation strategies and applications.

Span ordering for determining how the thematic roles should be combined. We use predicate form to control the order of core arguments. For example, to distinguish "the athlete was comforted by the doctor" from the semantically equivalent "the doctor comforted the athlete," we target the former ordering through a *passive* control, and the latter through an *active* control. Additionally, we use the location of blank tokens (<id_*> in Figure 1 and Table 1) to determine the position of generated arguments (Wu et al., 2021) — *e.g.,* where "in the operating room" appears in the generation.

2.2 Input Format Design

We integrate the aforementioned controls into the input format detailed in §A.1, and finetune seq2seq models to output corresponding full sentences.

As in Table 1, we start our input with a bracketed *header*, a series of abstract *control codes* (Table 2) with each denoting the semantic role and keywords for a span to realize. We map original semantic roles in PropBank to human-readable labels (*i.e.,* `ARG0` → `AGENT`) in order to leverage knowledge learned by pretrained models about roles' meanings (Paolini et al., 2021). After the header, we append the *context*, consisting of text to be preserved and blanks specifying where new text should be generated. Given such inputs, we train our generator to output text augmented with control codes and brackets, which together specify which generated spans correspond to which control signals. For example, in Table 1C, "[`LOCATIVE`: In the operating room]" represents the target span of control code "`LOCATIVE+partial: in`", and it is generated at the location of *blank* <id_0> right before the preserved *context* "the doctor".

Note that we explicitly separate the header from the context. This is to detach the placement of a role from its semantic representation, such that given any combination of target roles in the header — whose optimal ordering is usually unknown — the generator can recombine them in the most fluent way. We further remove possible correlations between the control codes and the blanks in the context in two ways: First, we order the control codes in an input-independent way (see §A.1) to discourage the generator from solely following their relative orders. Second, we insert extra empty blanks into the context (*e.g.,* <id_3> in Table 1B), so the generator can learn to generate spans in the blank locations that result in the most fluent text.

With this flexibility in argument reordering comes the challenge of making strict controls on a single argument: Even if we only want to change verb tense, the generator may reorder other arguments. To trade off generation flexibility and strict control, which facilitates minimal perturbations (Ross et al., 2020), we further vary the number of arguments encoded in the header. As in Table 1C, our generator can take inputs that only mask a subset of arguments, such that, *e.g.,* any changes on the `LOCATIVE` constraint or the `VERB` do not affect the agent and patient.

2.3 Training

We finetune T5-base (Raffel et al., 2020) on input-output pairs derived from gold semantic roles from OntoNotes 5.0 train (Table 1; Pradhan et al., 2013).[3] To make our generator sensitive to the different input formats, for each original input, we randomly sample the numbers of arguments to mask and extra empty blanks, and keyword content/specificity for each role (§A.2).

Standard maximum likelihood estimation (MLE) is insufficient for training our generator to follow

[2]We use http://spacy.io/ for verb or POS detection.

[3]On par with T5, the blanks are in the form of <extra_id_*>; we refer them as <id_*> for simplicity.

(a) Syntactically controlled rewriting	
Strategy	CHANGE_VTENSE(present) → [VERB+active+past→present: comfort]
Perturb.	In the operation room, the doctor comforts the athlete.
Strategy	CHANGE_VVOICE(passive) → [VERB+active→passive+past: comfort]
Perturb.	In...room, the athlete was comforted by the doctor.
Strategy	CHANGE_IDX(4:0) → <id_0> In the operation room <id_0>
Perturb.	The doctor comforted the athlete in the operation room.
Strategy	CORE(SWAP_CORE) → [AGENT+complete: the athlete→doctor | PATIENT+complete: the doctor→athlete]
Perturb.	In the operation room, the athlete comforted the doctor.

(b) Sentence expansion and abstraction	
Strategy	LOCATIVE:CHANGE_SPEC(partial) → [LOCATIVE+complete→partial: in the operation room]
Perturb.	Under the dim light in the operation room, the doctor comforted the athlete.
Strategy	LOCATIVE:DELETE → ~~[LOCATIVE+complete: in the operation room]~~
Perturb.	~~In the operation room,~~ the doctor comforted the athlete.

(c) Data recombination (with external labels and/or contents)	
Strategy	CAUSE:CHANGE_CONTENT(because he was in pain) →[CAUSE+complete: because he was in pain]
Perturb.	In the operation room the doctor comforted the athlete because he was in pain.

Table 3: We design a list of primitive operations on input controls to guide perturbations with the Tailor generator.

the control codes, as there may exist signals beyond the codes for the generation form. Consider the input: [VERB+active+past: comfort | AGENT+partial: athlete | PATIENT+complete: the doctor] In the operating room, <id_0>, <id_1> <id_2>. A generator trained with MLE may ignore controls AGENT and PATIENT and instead output text "The doctor comforted the athlete" rather than "The athlete comforted the doctor," as the former is more natural given context "in the operation room."

To encourage reliance on controls, we incorporate **unlikelihood training** (Welleck et al., 2020) to penalize generations that conflict with input controls. That is, besides Table 1A–C which are used for MLE, we also create "negative" samples by randomly perturbing the control codes in our header (as in Table 1N, last row), such that most spans in the target output are not aligned with the control codes. We create up to three negative samples per input by randomly perturbing 1) verb voice/tense and primary controls for arguments, 2) keyword contents, and 3) keyword specificities (§A.1). Our final training data consists of 223K positive and 541K negative examples.

3 Creating Perturbations with Tailor

With Tailor, we can create diverse perturbations by varying controls in inputs. Given an original sentence, we transform it to an input for Tailor by extracting its semantic parses, masking spans we wish to modify, and adding their control codes.[4] Then, we modify these signals to generate perturbed sentences with Tailor, filtering out degenerate ones.

Primitive perturbation operations. While the input can be modified arbitrarily, we provide an easily-extendable set of macros as in Table 3, which capture three common themes in the literature: First, *syntactic rewriting* primarily involves shuffling text to create paraphrases (Zhang et al., 2019) or adversarial examples (Iyyer et al., 2018). We implement such shuffling through operations that perturb predicate forms, move blank tokens, and swap keyword contents of arguments. Second, *expansion and abstraction* add or remove text fragments from a sentence (Wu et al., 2021). We recreate these through deletions of and operations on keywords. Finally, *data recombination* involves recombining existing textual fragments, within or across inputs (Akyürek et al., 2020; Andreas, 2020). With CHANGE_CONTENT, we can integrate additional context (*e.g.,* from corresponding paragraphs in question answering tasks) into generations.

While our control codes are mostly derived from semantic roles, these primitive operations broadly cover both syntactic and semantic changes. They can also be used in conjunction with external knowledge bases to achieve targeted edits.[5], or be composed to achieve more complex perturbation strategies. as shown in §5, §6, and Appendix §F.

Filtering generations. We notice that the Tailor generator produces degenerate outputs for some inputs; we exclude these using heuristics on content and perplexity scores (see §C for details).

4 Intrinsic Evaluation

Following previous work (Wu et al., 2021; Ross et al., 2020), we evaluate Tailor generations on

[4] External semantic role labelers can be used when gold annotations are not available. Our experiments use the opensourced implementation of Shi and Lin (2019): demo.allennlp.org/semantic-role-labeling, with a test F1 of 86.5 on the Ontonotes 5.0 dataset (Pradhan et al., 2013).

[5] For example, if combined with WordNet (Miller, 1998), Tailor perturbations can recreate natural logic (MacCartney and Manning, 2014): In Figure 1, we can create an entailment relationship by replacing doctor with its hyponym adult.

Generator	Closeness			Pred. Controllability			Arg. Controllability		
	$F1$	Precision	Recall	Lemma	Tense	Voice	Role	Content	Spec.
TAILOR	**64.3**	**66.5**	**73.4**	**74.3**	**80.3**	**81.6**	**70.5**	**64.5**	**64.5**
TAILOR$_{MLE}$	58.5	59.5	68.6	72.2	70.2	76.1	60.3	45.1	45.1

Table 4: Intrinsic evaluation performance in percentage. TAILOR generates perturbations that are close to the original sentence, while reasonably following all the controls specified in Table 2. Ablating unlikelihood training (TAILOR$_{MLE}$) hurts all metrics across the board.

sentence likelihood, controllability, and closeness.[6] We additionally evaluate TAILOR's unique ability to make fine-grained and compositional perturbations.

Metrics. *Likelihood* measures whether the generated text is grammatically correct and semantically meaningful. Following Ross et al. (2020), we ask whether perturbing a sentence with TAILOR drastically changes its likelihood. We compute the loss value for both the original and edited texts using a pretrained GPT-2, and report the ratio of edited / original. We desire for a value of 1.0, which indicates equivalent losses for the the two.

Controllability measures if the generator responds to the designated control criteria. We rely on cycle consistency to evaluate the controls in Table 2, checking *e.g.,* whether the predicted semantic roles on the generated text from an SRL predictor match the control codes in the input (*i.e.,* whether "in the midst of the earthquake" in Figure 1 gets detected with a `TEMPORAL` tag). Since SRL predictions can be noisy, we manually inspect a subset of 98 generated spans and verify that cycle consistency measures positively correlate with ground-truth controllability, with Matthews correlation coefficient $\phi = 0.49$ (more details in §B).

Closeness captures whether the generated sentence involves only necessary changes. Since our generator takes controls on the argument span level, we measure closeness with a weighted F1 score on the expected-to-change and actually-changed spans in the original sentence. We identify expected changes from perturbation operations; in Figure 1A, all spans should be changed except for agent "the doctor." Then, we deem a span actually edited if $\geq 50\%$ tokens within a span is changed (*e.g.,* "operation room" in `LOCATIVE`). We empirically picked the threshold as it tolerates cases where we only change keyword sparsity or when the stopwords remain in the generation. We weigh spans by their lengths to arrive at the final F1.

Compositionality. We evaluate TAILOR without any finetuning on the STYLEPTB benchmark (Lyu et al., 2021), which builds on the Penn Treebank and assesses both *single*, fine-grained transfers (*e.g., To Future Tense*) and compositional ones that concurrently edit multiple dimensions (*e.g., To Future Tense+ Active To Passive*). We report mean BLEU scores and compare to the transfer-specific baselines reported in the STYLEPTB paper.

Data. We use STYLEPTB for compositionality (see §F), and evaluate TAILOR on other metrics by perturbing 1,000 randomly selected sentences from the OntoNotes 5.0 valid set, created the same way as negative samples during training (§A.1).[7]

4.1 Results

TAILOR generates perturbations with a loss ratio of 0.982, indicating no notable change in language modeling loss after the edit. As shown in Table 4, its generations also tend to be close to the original sentence (F1 = 64.3%), with reasonably correct predicates (74.3%-81.6% of the time) and arguments (70.5% controllability on semantic roles and 64.5% on contents.) TAILOR also demonstrates the ability to make compositional changes; it achieves results comparable to those of fine-tuned baselines on 8/9 tested transfers, and even outperforms the fine-tuned baseline on 5 of them (§F, Table 11).

Effect of Unlikelihood Training: We compare TAILOR with a baseline that is finetuned on T5 *without* unlikelihood training (called TAILOR$_{MLE}$ in Table 4). Across all metrics, unlikelihood training outperforms TAILOR$_{MLE}$, with more controllable and minimal perturbations (up to a 20% increase).

Modulating likelihood and closeness: As mentioned in §2.2, our input format supports modulating likelihood and closeness. We can increase closeness by only masking the arguments we want to perturb. To quantify this effect, we randomly select a single argument to perturb for 1K sentences,

[6] We omit the diversity evaluation in POLYJUICE, as the keyword content control inherently impacts lexical diversity.

[7] Because these perturbations are generated randomly, some result in sets of controls that are *impossible* to follow. Thus, these results represent a lower bound on TAILOR's controllability in downstream applications, for which strategies would be designed in a more principled, targeted manner, restricting the perturbations to result in more plausible sets of controls. See §B for more details.

Dataset & Task		Top-K validity
BoolQ contrast set (Gardner et al., 2020)		82% (k=1)
Original	**Paragraph:**...his bride was revealed...Deadpool also discovers that he has a daughter...from a former flame. **Question:** does [AGENT: Deadpool] [VERB: have] [PATIENT: a kid in the comics]? (**Answer:** True)	
Strategy	Change entity (`AGENT:CHANGE_CONTENT(his bride)`)	
Perturb.	**Question:** does [AGENT: his bride] [VERB: have] [PATIENT: a kid in the comics]? (**Answer:** False)	
UD parsing contrast set (Gardner et al., 2020)		65% (k=10)
Original	**Sentence:** [AGENT: It] [VERB: has] [PATIENT: a diverse range of food at all prices and styles]. **PP attachment**: Noun ("at all prices and styles" attaches to "food")	

Strategy	Swap attachment from noun to verb (*noun→verb*)
	`PATIENT:CHANGE_CONTENT(a diverse range of food)`
	`LOCATIVE:CHANGE_CONTENT(at),CHANGE_SPEC(partial)`	

Perturb.	**Sentence:** [AGENT: It] [VERB: has] [PATIENT: a diverse range of food] [LOCATIVE: at every turn]. **PP attachment**: Verb ("at every turn" attaches to "has")	
MATRES contrast set (Gardner et al., 2020)		71% (k=1)
QA implication (Ribeiro et al., 2019)		81% (k=1)

Table 5: A demonstration of how we recreate contrast sets. Using primitive operations in Table 3, TAILOR supports context-aware and compositional changes. More examples (*e.g.,* changing PP attachment *noun→verb*) are in §D.

but vary the number of masked arguments and the number of inserted blanks. Closeness is maximized when we only mask the target argument to perturb in the format of Table 1B (with $F1 = 67.4\%$), whereas masking two extra arguments and inserting six extra blanks decreases closeness by 3% and 6%, respectively. On the other hand, we can trade off closeness to prioritize likelihood by adding more blanks (*e.g.,* insert extra roles whose optimal locations are not known in advance). On another 1K sentences, we observe that adding six extra blanks increases the likelihood ratio from 0.93 to 0.95.

5 Contrast Set Creation

Manually creating contrast sets is expensive, *e.g.,* Gardner et al. (2020) reported spending 10-15 minutes per perturbation for UD Parsing, whereas labeling existing data is more efficient (Wu et al., 2021). We show that TAILOR can save human labors by automatically generating contrast set instances, such that annotators only have to label them, on four tasks: boolean question answering (BoolQ: Clark et al., 2019), extractive QA (SQuAD: Rajpurkar et al., 2016), dependency tree parsing (UD English: Nivre et al., 2016), and temporal relation extraction (MATRES: Ning et al., 2018).

5.1 Replicating Contrast Sets with TAILOR

We take advantage of two key properties of TAILOR: First, TAILOR can make **context-dependent** changes. To recreate the *BoolQ contrast set*, we replicate *change events* in Gardner et al. (2020) by replacing content keywords in questions with words in the paragraph that have the same semantic roles. For example, the paragraph in Table 5 indicates "his bride" can serve as an `AGENT`. Second, TAILOR allows for **compositonal** changes. As in Table 5, we change prepositional phrase (PP) attachments from noun to verb to recreate the *UD Parsing contrast set* by removing the prepositional phrase from the patient keyword (*e.g.,* "a diverse range of food ~~at all prices and styles~~") and introducing an adjunct argument with the preposition as partial keyword (*e.g.,* `LOCATIVE` "at"). These strategy details are in §D.1.

Contrast set validity. We consider our perturbation strategies successful if they help reduce human labor, *i.e.,* a contrast set author can easily label or take inspiration from TAILOR's generations. Two authors sampled 100 original instances per task, inspected the *top-K* TAILOR perturbations, and labeled an instance to be **valid** if there is at least one perturbation that changes the groundtruth answer while being fluent or requiring only minor fixes.[8] Table 5 shows that these TAILOR perturbation strategies generate contrast sets with high validity.[9]

5.2 Measuring Contrast Set Quality

We sanity check that TAILOR-generated contrast sets can be used to reveal model errors. For example, a T5-BASE model finetuned on BoolQ (with test accuracy 83%) has a performance of 65% on both

[8] Because we exercised controls at different granularity (*i.e.,* UD requires sourcing contents from the generator while others mostly require syntactic rewrites with predetermined content), we set $k = 10$ for UD—an upper bound for not overloading the human inspector—and $k = 1$ for other tasks.

[9] TAILOR achieves higher validity changing attachment from *noun→verb* (82%) than *verb→noun* (48%). Discussion in §D.

our TAILOR contrast sets and Gardner et al. (2020)'s (more in §D.2). However, this metric is only a proxy for the quality of evaluation data, since it can be made intentionally low if we generate all examples to target a known model error. Thus, we directly analyze the quality of TAILOR-generated contrast sets by measuring their **lexical diversity** and impact on **feature-level artifacts**, both of which play important roles in dataset debiasing.

We measure lexical diversity on UD Parsing contrast sets because it involves sufficient generation of new content. We compare TAILOR- and human-generated (Gardner et al., 2020) contrastive edits for the same 100 UD instances: we randomly sample one edit for each valid instance, heuristically extract modified PPs, and compute diversity as the ratio of unique to total new tokens in the PPs, filtering stopwords. For *noun→verb*, the ratios are respectively 0.78 and 0.99 for TAILOR and humans; for *verb→noun*, both are 1.0. Thus, TAILOR can help generate contrast sets without significantly reducing lexical diversity. TAILOR outputs are distinguishable from humans': their unique tokens only overlap for < 15% in *verb→noun*, and ∼6% for *noun→verb*, suggesting that TAILOR can be used as a collaborative tool to diversify generation.

We ask, using Gardner et al. (2021)'s statistical test, whether TAILOR perturbations can reduce dataset artifacts. Figure 2 plots the numbers of occurrences of each word against the conditional probability of the positive label given that word, on BoolQ validation data (red dots) and the contrast created by TAILOR (green dots). All features above or below the blue line show statistically significant correlation with positive labels and thus are considered dataset artifacts. While many words in the original data show such a bias, most in TAILOR perturbations fall within the confidence region. Thus, TAILOR can help create less biased evaluation data.

5.3 Discussion

Across the four tasks, we are able to replicate all perturbation strategies described in the original contrast sets. While TAILOR requires manual effort to implement perturbation strategies, we believe the overall saved annotation effort outweighs this initial cost. First, with the manual perturbations abstracted into TAILOR strategies, they can be generalized to larger datasets without requiring additional annotation effort. This is important especially for tasks whose single-instance annotation time is significant (*e.g.,* UD Parsing). Second, given that TAILOR generations are distinguishable from human ones, they may have the potential to compensate for human omissions and thereby increase test case variety, which has been shown to be beneficial in prior work (Ribeiro et al., 2020). Third, the implementation overhead itself diminishes as more strategies are implemented. In BoolQ, while Gardner et al. (2020) manually created "a diverse set of perturbations, including adjective, entity, and event changes" (see their Appendix B.9), these are all a type of *data recombination* in Table 3, and we were able to unify their implementations with TAILOR into the aforementioned match-and-replacement.

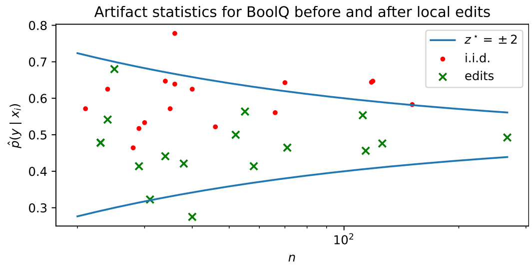

Figure 2: Dataset artifacts in original BoolQ validation set vs. contrast set created with TAILOR.

6 Data Augmentation

We explore whether TAILOR can be combined with noisy automated labeling for data augmentation. For the Stanford Natural Language Inference (SNLI) task (Bowman et al., 2015), we show that data augmentation with TAILOR perturbations increases model robustness to inference heuristics.

Min et al. (2020) find that augmenting SNLI training data by swapping hypotheses' subject/objects (*e.g., This collection contains 16 El Grecos.* ↛ *16 El Grecos contain this collection*) improves performance on HANS, a challenge set for diagnosing fallible syntactic heuristics in NLI models (McCoy et al., 2019). Following this, we use TAILOR to perturb hypotheses with the `SWAP_CORE` operation such that *original hypothesis* → *premise* and *perturbed hypothesis* → *new hypothesis.*

We finetune ROBERTA-BASE (Liu et al., 2019) on different data: original SNLI train data (unaugmented baseline), SNLI train augmented with Min et al. (2020) (augmented baseline, referred to as *Syntactic Perturb.* in Table 6), and SNLI train augmented with TAILOR perturbations. We augment ∼2% of SNLI train.[10] For each subset, we train 20 models with different random seeds. We evaluate

[10]We augment the 549,367 SNLI train instances with 10,987 new instances. See §E for more details.

Training Data	SNLI	HANS Subset All	Entail.	Non-entail.
SNLI Train	**91.1**	64.7	**99.0**	30.5
+ Syntactic Perturb.	91.0	67.5	95.8	39.2
+ **Tailor Perturb.**	**91.1**	**70.5**	81.3	**59.7**

Table 6: Tailor augmentations lead to statistically significant gains on the HANS challenge set, without decreasing in-domain accuracy.

each classifier on the in-domain SNLI test set and the out-of-domain HANS test set.[11]

As shown in Table 6, augmentation with Tailor leads to 5.8-point gain on HANS overall, HANS and a **29.2**-point gain on "non-entailment," compared to the unaugmented baseline. The improvements are significant, with $t = -6.42$, $p < 10^{-3}$ using Student's t-test. Thus, Tailor perturbations decrease reliance on the lexical-overlap-based inference heuristic for NLI. Furthermore, Tailor outperforms *Syntactic Perturb.*, an augmented baseline designed specifically for NLI. We hypothesize that although they create augmentations through similar transformations, Min et al. (2020)'s approach is limited to inputs with specific syntactic configurations, whereas Tailor's `SWAP_CORE` argument is applicable to any `AGENT` and `PATIENT` arguments. Thus, Tailor is useful for improving model robustness – more so than template-based approaches.

7 Related Work

Controllable text generation has been widely used to influence various properties of generated text for data augmentation (Lee et al., 2021), style transfer (Reid and Zhong, 2021; Madaan et al., 2020a), adversarial example generation (Iyyer et al., 2018), etc. Most generators take simple labels like tense (Hu et al., 2017) or topic (Keskar et al., 2019), which underspecify desired transformations. Recent work has explored using syntactic signals for paraphrasing (Iyyer et al., 2018; Kumar et al., 2020), which are similar to ours in their high-dimensional specification. To the best of our knowledge, Tailor is the first to incorporate fine-grained *semantic* controls. Structured generation methods, which reconstruct sentences based on semantic representations, are also closely related. Abstract Meaning Representation (Banarescu et al., 2013; Mager et al., 2020) is an alternative worth exploring, as it may further enable controls on entity recursions (Damonte and Cohen, 2019), though expressing such relationships is nontrivial.

Controlled generators have also been successfully used to perturb text for model training, evaluation, and explanation. They usually rely on application-specific labels (Ross et al., 2020; Madaan et al., 2020b; Sha et al., 2021; Akyürek et al., 2020) or require pairs of original and perturbed sentences (Wu et al., 2021), which are expensive to generalize. Recently, several works explore explicitly modeling syntactic structures in controlled text generation (Chen et al., 2019; Bao et al., 2019; Sun et al., 2021). For example, Huang and Chang (2021) designed SynPG, a paraphraser that can mimic parse tree structures learned from non-paired data. In contrast, we focus on fine-grained *semantic* perturbations that can be composed into various changes beyond paraphrasing.

Also related are the creation of minimally edited datasets, either through manual rewriting (Gardner et al., 2020; Kaushik et al., 2020), or creating perturbation templates (Andreas, 2020; Li et al., 2020; Ribeiro et al., 2020; Wu et al., 2019); Tailor reduces the human efforts these studies require.

8 Conclusion

We propose Tailor, a flexible system that enables task-agnostic, complex and context-aware perturbations. Crucially, it shows that language models can be finetuned to learn representations of control codes, if paired with unlikelihood training, which encourages reliance on structured controls, rather than surrounding natural text. Beyond the perturbation oriented tasks, we envision Tailor supporting broader controlled generation tasks, and encourage future work to explore alternative control signals for different objectives (*e.g.,* syntactic roles in §7).

While being widely applicable, Tailor's effectiveness varies for different inputs. For example, some inputs derived from SRL predictors may miss rare semantic roles; empirically, this did not seem to be a bottleneck, as exposing biases in downstream tasks usually do not require rarity at the semantic role level (*e.g.,* the syntactic heuristics in NLI only requires swapping agents and patients). Moreover, some text leads to occasional degeneration. Future work can explore the effect of penalizing generation at the span levels (vs. sequences) or more strategically balancing positive and negative samples. Having noted these opportunities, we believe Tailor is already a powerful tool for perturbations, and we opensource it at `[URL omitted]`.

[11]For HANS, we follow the standard practice and collapse *neutral* and *contradiction* predictions to *non-entailment*.

References

Ekin Akyürek, Afra Feyza Akyürek, and Jacob Andreas. 2020. Learning to recombine and resample data for compositional generalization. *arXiv preprint arXiv:2010.03706*.

Jacob Andreas. 2020. Good-enough compositional data augmentation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7556–7566, Online. Association for Computational Linguistics.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria. Association for Computational Linguistics.

Yu Bao, Hao Zhou, Shujian Huang, Lei Li, Lili Mou, Olga Vechtomova, Xin-yu 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, Florence, Italy. Association for Computational Linguistics.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. 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, Florence, Italy. Association for Computational Linguistics.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. 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 2924–2936, Minneapolis, Minnesota. Association for Computational Linguistics.

Marco Damonte and Shay B. Cohen. 2019. Structural neural encoders for AMR-to-text generation. 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 3649–3658, Minneapolis, Minnesota. Association for Computational Linguistics.

Matt Gardner, Yoav Artzi, Victoria Basmov, Jonathan Berant, Ben Bogin, Sihao Chen, Pradeep Dasigi, Dheeru Dua, Yanai Elazar, Ananth Gottumukkala, Nitish Gupta, Hannaneh Hajishirzi, Gabriel Ilharco, Daniel Khashabi, Kevin Lin, Jiangming Liu, Nelson F. Liu, Phoebe Mulcaire, Qiang Ning, Sameer Singh, Noah A. Smith, Sanjay Subramanian, Reut Tsarfaty, Eric Wallace, Ally Zhang, and Ben Zhou. 2020. Evaluating models' local decision boundaries via contrast sets. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1307–1323, Online. Association for Computational Linguistics.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. AllenNLP: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 1–6, Melbourne, Australia. Association for Computational Linguistics.

Matt Gardner, William Merrill, Jesse Dodge, Matthew E Peters, Alexis Ross, Sameer Singh, and Noah Smith. 2021. Competency problems: On finding and removing artifacts in language data. *arXiv preprint arXiv:2104.08646*.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 1–18, Boulder, Colorado. Association for Computational Linguistics.

Chris Hokamp and Qun Liu. 2017. Lexically constrained decoding for sequence generation using grid beam search. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1535–1546, Vancouver, Canada. Association for Computational Linguistics.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2017. Toward controlled generation of text. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1587–1596. PMLR.

Kuan-Hao Huang and Kai-Wei Chang. 2021. Generating syntactically controlled paraphrases without using annotated parallel pairs. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1022–1033, Online. Association for Computational Linguistics.

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, New Orleans, Louisiana. Association for Computational Linguistics.

Divyansh Kaushik, Eduard H. Hovy, and Zachary Chase Lipton. 2020. Learning the difference that makes A difference with counterfactually-augmented data. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

N. Keskar, Bryan McCann, L. Varshney, Caiming Xiong, and R. Socher. 2019. Ctrl: A conditional transformer language model for controllable generation. *ArXiv*, abs/1909.05858.

Ashutosh Kumar, Kabir Ahuja, Raghuram Vadapalli, and Partha Talukdar. 2020. Syntax-guided controlled generation of paraphrases. *Transactions of the Association for Computational Linguistics*, 8:329–345.

Kenton Lee, Kelvin Guu, Luheng He, Timothy Dozat, and Hyung Won Chung. 2021. Neural data augmentation via example extrapolation. *ArXiv*, abs/2102.01335.

Chuanrong Li, Lin Shengshuo, Zeyu Liu, Xinyi Wu, Xuhui Zhou, and Shane Steinert-Threlkeld. 2020. Linguistically-informed transformations (LIT): A method for automatically generating contrast sets. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 126–135, Online. Association for Computational Linguistics.

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

Yiwei Lyu, Paul Pu Liang, Hai Pham, Eduard Hovy, Barnabás Póczos, Ruslan Salakhutdinov, and Louis-Philippe Morency. 2021. StylePTB: A compositional benchmark for fine-grained controllable text style transfer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2116–2138, Online. Association for Computational Linguistics.

Bill MacCartney and Christopher D Manning. 2014. Natural logic and natural language inference. In *Computing meaning*, pages 129–147. Springer.

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

Nishtha Madaan, Inkit Padhi, Naveen Panwar, and Diptikalyan Saha. 2020b. Generate your counterfactuals: Towards controlled counterfactual generation for text. *arXiv preprint arXiv:2012.04698*.

Manuel Mager, Ramón Fernandez Astudillo, Tahira Naseem, Md Arafat Sultan, Young-Suk Lee, Radu Florian, and Salim Roukos. 2020. GPT-too: A language-model-first approach for AMR-to-text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1846–1852, Online. Association for Computational Linguistics.

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy. Association for Computational Linguistics.

George A Miller. 1998. *WordNet: An electronic lexical database*. MIT press.

Junghyun Min, R. Thomas McCoy, Dipanjan Das, Emily Pitler, and Tal Linzen. 2020. Syntactic data augmentation increases robustness to inference heuristics. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2339–2352, Online. Association for Computational Linguistics.

Qiang Ning, Hao Wu, and Dan Roth. 2018. A multi-axis annotation scheme for event temporal relations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1318–1328, Melbourne, Australia. Association for Computational Linguistics.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal Dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1659–1666, Portorož, Slovenia. European Language Resources Association (ELRA).

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Giovanni Paolini, Ben Athiwaratkun, Jason Krone, Jie Ma, Alessandro Achille, RISHITA ANUBHAI, Cicero Nogueira dos Santos, Bing Xiang, and Stefano

Soatto. 2021. Structured prediction as translation between augmented natural languages. In *International Conference on Learning Representations*.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using OntoNotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 143–152, Sofia, Bulgaria. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

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

Machel Reid and Victor Zhong. 2021. Lewis: Levenshtein editing for unsupervised text style transfer. *arXiv preprint arXiv:2105.08206*.

Marco Tulio Ribeiro, Carlos Guestrin, and Sameer Singh. 2019. Are red roses red? evaluating consistency of question-answering models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6174–6184, Florence, Italy. Association for Computational Linguistics.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912, Online. Association for Computational Linguistics.

Alexis Ross, Ana Marasović, and Matthew E Peters. 2020. Explaining nlp models via minimal contrastive editing (mice). *arXiv preprint arXiv:2012.13985*.

Lei Sha, Patrick Hohenecker, and Thomas Lukasiewicz. 2021. Controlling text edition by changing answers of specific questions. *CoRR*, abs/2105.11018.

Shikhar Sharma, Layla El Asri, Hannes Schulz, and Jeremie Zumer. 2017. Relevance of unsupervised metrics in task-oriented dialogue for evaluating natural language generation. *CoRR*, abs/1706.09799.

Peng Shi and Jimmy Lin. 2019. Simple bert models for relation extraction and semantic role labeling. *arXiv preprint arXiv:1904.05255*.

Jiao Sun, Xuezhe Ma, and Nanyun Peng. 2021. AESOP: Paraphrase generation with adaptive syntactic control. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5176–5189, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Damien Teney, Ehsan Abbasnedjad, and Anton van den Hengel. 2020. Learning what makes a difference from counterfactual examples and gradient supervision. *arXiv preprint arXiv:2004.09034*.

Chantal van Son, Oana Inel, Roser Morante, Lora Aroyo, and Piek Vossen. 2018. Resource interoperability for sustainable benchmarking: The case of events. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. 2020. Neural text generation with unlikelihood training. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

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.

Tongshuang Wu, Marco Tulio Ribeiro, Jeffrey Heer, and Daniel Weld. 2019. Errudite: Scalable, reproducible, and testable error analysis. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 747–763, Florence, Italy. Association for Computational Linguistics.

Tongshuang Wu, Marco Tulio Ribeiro, Jeffrey Heer, and Daniel S. Weld. 2021. Polyjuice: Generating counterfactuals for explaining, evaluating, and improving models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Yuan Zhang, Jason Baldridge, and Luheng He. 2019. PAWS: Paraphrase adversaries from word scrambling. 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 1298–1308, Minneapolis, Minnesota. Association for Computational Linguistics.

Appendices

A Tailor Generator Details

A.1 Input and Output Formats

All headers in inputs to the Tailor generator begin with predicate controls, followed by core argument controls (first `AGENT`, then `PATIENT`), and then randomly ordered adjunct argument controls (`LOCATIVE`, `TEMPORAL`, etc.). Secondary controls are always given in the order of *control code+voice+tense:lemma* for verbs and *control code+keyword specificity:keyword content* for arguments. We also blank the auxiliary verbs of the predicate in an input, using `spacy` to detect them. We exclude discontinuous arguments (*e.g.,* those with raw SRL labels `B-C-*`), as well as those with referents (*e.g.,* those with raw SRL labels `B-R-*`), from input headers. We map `ARG0` → `AGENT` and `ARG1` → `PATIENT`. For other numbered arguments, we create human-readable labels by using argument functions included in the PropBank frame for the given predicate (Palmer et al., 2005).

On the output side, we ask the model to generate the full sentence (Table 1). We add the semantic roles for all the generated arguments, to help the generator build explicit mappings between the input control codes and the output spans – this can be important when the input codes are ambiguous (*e.g.,* a `TEMPORAL` argument and a `LOCATIVE` argument that both have keywords "in"). To use generations in downstream applications, we remove these control codes to obtain cleaned outputs using regular expression matching.

A.2 Training details

Training inputs. During training, we randomly select, with equal probabilities, whether to mask all arguments or a subset. If a subset, we uniformly select the proportion of arguments to mask. To determine the number of extra blanks, we uniformly select a value less than 10 and set the number of blanks to be the maximum of that selected value and the number of arguments to mask. Any extra blanks (*i.e.,* remaining after masking arguments) are inserted between subtrees of the predicate.

We also randomly select keyword contents and keyword specificities. For each argument span, we extract, using `spacy`, four keyword types from the span: *noun chunks*, *random subtrees*, *exact* keywords, and *prefixes*. For prefixes, we uniformly select a number of tokens to include as the keyword (from 1 to the entire span). Once we extract all keyword candidates, we create corresponding keyword specificities: A keyword is *complete* if it contains all tokens in the original span, *partial* if it contains at least all but 5 tokens, and *sparse* otherwise. Then, we uniformly select a keyword content/specificity pair for each span from the set of keyword candidates (including the `*` symbol).[12]

To generate unlikelihood samples, we use three perturbation strategies on inputs: 1) Change *semantic roles* by swapping thematic role control codes (agent/patient), changing adjunct argument control codes to a uniformly selected other adjunct control code, and changing verb tense/voice. We swap verb tense/voice because the control code `VERB` does not have natural candidate swaps, given that predicates are the building block for semantic parses. We also swap the control codes in the target output. 2) Change keyword *contents* by replacing verb lemmas and keywords for both the predicate and all arguments. To make content swaps, we first gather the most commonly occurring keyword contents for each argument and predicate in Ontonotes 5.0 train, extracted according to the same process as described above for creating training inputs. For each primary control code and keyword specificity (*e.g.,* `TEMPORAL+partial`), we store the 15 most commonly occurring keyword contents. To create the negative inputs, for each span, we uniformly sample from these stored keywords given the span's control code and keyword specificity. This perturbation is designed to discourage the generator from ignoring the keyword content and merely generating commonly occurring text for particular semantic roles. 3) Change keyword *specificities* by uniformly selecting a different specificity. We weight each unlikelihood sample equally, with a reward of -1 (vs +1 for positive samples).

Hyperparameters. We train the Tailor generator using `Transformers` (Wolf et al., 2020) for 10

[12] Because of how keywords are sampled, we notice that the generator is sensitive to the case of keyword contents. For example, if the keyword for a temporal span is *In 1980* instead of *in 1980*, Tailor is biased towards generating it at the beginning of the sentence. We hypothesize that because some of the keywords we sample during training are cased (*e.g., exact* will lead to a cased keyword for a capitalized span beginning a sentence), the generator learns a bias towards generating spans with uppercase keyword at the beginning of the sentence. In applying the generator to perturbations, the case of keyword contents can be used to manipulate the order of generated roles when a certain order of generated contents is desired; otherwise, uncased keywords can be used.

epochs with early stopping. We use batch size 4 and default values for other parameters (learning rate of 5e-5, Adam optimizer).

B Intrinsic Evaluation Details

Effectiveness of cycle consistency. To evaluate to what extent cycle consistency reflects true controllability, we conducted additional manual annotation on role-following. We sampled 25 sentences from the Ontonotes 5.0 development set, transformed them into inputs with varying numbers of masked arguments and blank tokens, and created up to two perturbed inputs per sentence by randomly replacing their blanked adjunct arguments with other candidate semantic roles (using `CHANGE_TAG`). The candidate roles were extracted from the frameset for each predicate verb. We also changed the keyword specificity to `SPARSE`, to make these role swaps more plausible.

We collected Tailor and Tailor $_{MLE}$ generations from both the original and perturbed inputs, and one author manually validated the generated span for each specified argument (98 in total). Our annotations were *following* or *not following* the control (*i.e.,* the span matches/does not match the designated semantic role), or the set of controls can be *impossible to follow* if the human annotator could not think of any generation that would satisfy the control codes, due to a conflict between the role, keywords, and blank placement. We then computed the Matthews correlation coefficient (MCC) between the controllability of the role label as measured by the SRL predictor with the gold controllability annotations for the subset of roles without annotation *impossible*. The MCCs are 0.49 and 0.51 for Tailor $_{MLE}$ and Tailor, respectively, suggesting that the cycle consistency measures positively correlate with true controllability measures.

Additionally, we measure to what extent the controllability measures from cycle consistency correlate with whether a set of controls is *impossible* to follow. The MCCs are -0.33 for both Tailor and Tailor $_{MLE}$; thus, incorrect role-following as measured by cycle consistency is positively correlated with controls that are impossible to follow. 14/98 instances were manually annotated as having impossible-to-follow controls, suggesting that a nontrivial proportion of the generations for which our intrinsic evaluation measures in §4 found to be unaligned with designated role control codes may be explained by impossible-to-follow controls.

C Degenerate Outputs

We observe that Tailor produces degenerate outputs for some inputs, as shown in Table 8. We hypothesize that this is a byproduct of unlikelihood training: The generator may learn to reduce the likelihood of negative sequences by generating tokens that are very unlikely to appear in natural text. Certain generation hyperparameters, such as the number of beams, can reduce the number of degenerate outputs. While we perform unlikelihood training at the sequence level, future work can investigate the effect of penalizing generation at the level of tokens or spans, which may provide finer-grained signals for which spans should be considered unlikely, as well as more strategically balancing positive and negative samples.

Filtering. To exclude degenerations when using Tailor generations in downstream applications, we employ a combination of heuristics and perplexity-based filtering. As shown by the examples in Table 8, degenerate outputs are easy to detect: We can simply search for whether the output includes "sanatate." We also use cutoffs in perplexity scores computed with GPT-2 to filter degenerations, as degenerations have significantly lower perplexities than non-degenerate outputs: For generations for 300 randomly sampled validation inputs, the Tailor generator produced generations with a mean perplexity of -346.46 for degenerate outputs (12/300) compared to -86.747 for others.

D Contrast Set Details (§5)

D.1 Perturbation Strategies

In Table 7, we illustrate our perturbation strategies for creating contrast sets. Besides BoolQ, already introduced in §5, the *Matres contrast set* Gardner et al. (2020) relies on within-sentence context: As a task that requires detecting and changing the temporal order of two verbs, our perturbations heavily rely on their syntactic relationships. For example, to change the *appearance order* of verbs in text (as described in (Gardner et al., 2020)), we would take the parent verb as the base predicate, and `MOVE` the text span containing the child verb.

For *QA implication* (Ribeiro et al., 2019), we combine Tailor with semantic heuristics: by defining mappings between WH-words and answer types (*e.g.,* "who" and "the Huguenots"), we can easily create new questions about different targets.

	Dataset & Task	Top-K validity
MATRES contrast set (Gardner et al., 2020)		71% (k=1)
Original	**Sentence:** Volleyball is a popular sport in the area, and [AGENT: more than 200 people] would be [VERB: watching] [PATIENT: the game], the chief said. **Order:** watching happens *after* said	
Perturbation strategy: Change tense		
Edits	`VERB:CHANGE_VFORM(past)` → [`VERB+active+present→past`: watch] Volleyball is...200 people <id_0> the game, the chief said.	
Perturbed	**Sentence:** Volleyball is a popular sport in the area, and [AGENT: more than 200 people] [VERB: watched] [PATIENT: the game], the chief said. **Order:** watched happens *before* said	
Perturbation strategy: Change order		
Edits	`PATIENT:MOVE` → [`VERB+active+past`: say | `AGENT+complete`: Volleyball...the game] <id_0> , the chief said <id_0> .	
Perturbed	**Sentence:**[AGENT: the chief] [VERB: said] [PATIENT: Volleyball is a popular sport in the area, and more than 200 people would be watching the game]. **Order:** said happens *before* watch	
BoolQ contrast set (Gardner et al., 2020)		82% (k=1)
Original	**Paragraph:**...his bride was revealed in the webcomic...Deadpool also discovers that he has a daughter by the name of Eleanor, from a former flame of Deadpool named Carmelita. **Q:** does [AGENT: Deadpool] [VERB: have] [PATIENT: a kid in the comics]? (**A:** True)	
Perturbation strategy: Change entity		
Edits	`AGENT:CHANGE_CONTENT(his bride);` → [`VERB+active+present`: have | `AGENT+complete`: Deadpool→his bride] does <id_0> <id_1> a kid in the comics?	
Perturbed	**Q:** does [AGENT: his bride] [VERB: have] [PATIENT: a kid in the comics]? (**A:** False)	
UD parsing contrast set (pp attachment) (Gardner et al., 2020)		65% (k=10)
Original	**Sentence:** Do [AGENT: you] [VERB: prefer] [PATIENT: ham, bacon or sausages] [ADVERBIAL: with your breakfast]? **PP attachment:** Verb ("with your breakfast" attaches to "prefer")	
Perturbation strategy: Swap attachment to Noun		

Edits	`PATIENT:CHANGE_CONTENT(ham, bacon or sausages with),CHANGE_SPEC(partial)`
	`ADVERBIAL:DELETE`
	→ [`VERB+active+present`: prefer | `PATIENT+complete→partial`: ham, bacon or sausages with~~| ADVERBIAL+complete: with your breakfast~~] <id_0> you <id_1> <id_2> <id_3>?	

Perturbed	**Sentence:** Do [AGENT: you] [VERB: prefer] [PATIENT: ham, bacon or sausages with bacon on them]? **PP attachment**: Noun ("with bacon them" attaches to "sausages")	
Original	**Sentence:** [AGENT: It] [VERB: has] [PATIENT: local boutiques and a diverse range of food at all prices and styles]. **PP attachment**: Noun ("at all prices and styles" attaches to "food")	
Perturbation strategy: Swap attachment to Verb		

Edits	`PATIENT:CHANGE_CONTENT(local boutiques and a diverse range of food)`
	`LOCATIVE:CHANGE_CONTENT(at),CHANGE_SPEC(partial)`
	→ [`VERB+active+present`: have | `PATIENT+complete`: local boutiques and a diverse range of food ~~at all prices and styles~~ | `LOCATIVE+partial: at`] <id_0> you <id_1> <id_2> <id_3>?	

Perturbed	**Sentence:** [AGENT: It] [VERB: has] [PATIENT: local boutiques and a diverse range of food] [LOCATIVE: at every turn]. **PP attachment**: Verb ("at every turn" attaches to "has")	
QA implication (Ribeiro et al., 2019)		81% (k=1)
Original	**Q:** [MANNER: How] did [AGENT: the Huguenots] [VERB: defend] [PATIENT: themselves]? **A:** their own militia	
Perturbation strategy: Swap answer to be agent		
Edits	`AGENT:CONTENT(who); MANNER:CONTENT(their own militia),SPEC(partial)` → [`VERB+active+past`: defend | `AGENT+complete`: the Huguenots→who | `PATIENT+complete`: themselves | `MANNER+complete→partial`: how→their own militia] <id_0> <id_1> <id_2> <id_3>?	
Perturbed	**Q:** [AGENT: Who] has [VERB: defended] [PATIENT: themselves] [MANNER: by setting up their own militia]? **A:** the Huguenots	

Table 7: A demonstration of how we recreate contrast sets for different tasks (§5). Using primitive operations in Table 3, Tailor supports context-aware and compositional changes.

Input	Degenerate Output
[VERB+passive+past: lower | AGENT: * | PATIENT+partial: corporate bonds | TEMPORAL+complete: this year] One indication of a growing number of junk defaults , Mr. Asquith says , is that about half of the $ 3 billion of <id_0> that <id_1> <id_2> <id_3> to a default rating <id_4> <id_5> are junk bonds sold during the market 's big issue years of 1984 through 1986 ..	pastra pastra sanatate sanatate pastraurmatoarele sanatateurmatoarele pastradatorita sanatatedatorita pastracresterea sanatate urmeaza sanatateinformatiile sanatatecresterea pastra urmeaza urmeaza pastrainformatiileinformatiiledatoritadatoritaurmatoarele.
[VERB+active+present: visit | AGENT: * | PATIENT+partial: Galilee | TEMPORAL: *] <id_0> went to <id_1> <id_2> <id_3> .	AG pastra pastra sanatate sanatate pastraurmatoarele sanatateurmatoareleurmatoarele pastrainformatiile sanatate-informatiileinformatiile pastradatorita sanatatedatoritadatori-taurmatoareledatoritainformatiile dumneavoastra sanatate urmeaza sanatatecresterea

Table 8: Example inputs from the validation set for which the Tailor generator outputs degenerate text.

For *UD English* (Nivre et al., 2016), we use constrained decoding (Hokamp and Liu, 2017) to prevent generation of the original prepositional phrase. Our strategy for changing prepositional phrase (PP) attachments from *verb→noun* is similar to that of *noun→verb*, introduced in §5. We use the following composition of perturbation operations: append the preposition to the patient keyword (*e.g.,* "ham or sausages with"), change patient keyword specificity from complete→partial (to generate a new PP attaching to the patient), and delete the argument with original verb attachment (*e.g.,* ADVERBIAL "with your breakfast").

We note that Tailor achieves higher validity changing attachment from *noun→verb* (82%) than *verb→noun* (48%). This result is expected, as all semantic role labeling arguments attach to verb predicates; thus, introducing controls for an SRL argument (*e.g.,* LOCATIVE with keyword content "at") to generate a preopositional phrase with verb attachment ("at every turn") reflects the training objective of the generator. On the other hand, our *verb→noun* strategy involves appending the preposition to the keyword control for an argument, and none of our controls explicitly reflect the target attachment of a prepositional phrase within an argument (*e.g.,* keyword controls do not specify whether "with" should attach to "sausages" vs "ham"). Furthermore, preposition keywords within an SRL argument do not deterministically lead to noun attachments in our training data–Sometimes a preposition within an argument may reflect verb attachment (*e.g.,* in the case of "Do [AGENT: you] [VERB: prefer] [PATIENT: eating with a fork or eating with a knife]?"; here, "eating with a fork or eating with a knife" is the patient of "prefer" but prepositional phrase "with a fork" attaches to verb "eating.") Because the training objective of our generator does not provide deterministic signal for noun attachment outputs, we do not expect our *verb→noun* strategy to always result in generations with noun attachment. Our *verb→noun* strategy is instead intended to *facilitate* the collection of text with noun attachment. Future work can investigate incorporating auxiliary signals about target configurations of keyword contents in outputs (*e.g.,* that a preposition should depend on a particular word in the span).

Dataset	Task Eval Original	Contrast Set Human ↓	Contrast Set Tailor ↓
BoolQ	82.8	64.8 (-17.5)	64.7 (-17.6)
SQuAD	91.8	66.1 (-25.7)	55.3 (-36.5)
MATRES	70.3	49.4 (-20.9)	42.3 (-28.0)

Table 9: Accuracies of predictors on original task evaluation data and contrasts sets. The performance drops on contrast sets (vs. original test accuracies), shown in parentheses, are similar for Tailor-generated contrast sets and expert-created sets (Gardner et al., 2020; Ribeiro et al., 2019).

D.2 Predictor Performance Evaluation

The performances of downstream predictors on original task evaluation data and contrast sets, both Tailor-generated and human-expert-generated, are shown in Table 9.[13] For SQuAD, we evaluate a fine-tuned RoBERTa, the most downloaded model hosted on Huggingface,[14] and use the QA implication challenge set (Rajpurkar et al., 2016) as the human contrast set. Since we could not find readily available predictors for BoolQ and MATRES, we formulate these tasks as a text-to-text task and fine-tune T5-base for 10 epochs; we evaluate the

[13] We report accuracy on the test set for MATRES and held-out validation sets for BoolQ and SQuAD, which do not have publicly available test sets.

[14] https://huggingface.co/deepset/roberta-base-squad2

Premise	Tailor-Generated Hypothesis
A lady in shorts is riding a bike.	A bike is riding a lady in shorts.
A band plays drums in the parade.	Drums are playing a band in the parade.
A young woman eating doritos on mars.	Doritos is eating a young woman on mars
A crowd of people is outside watching a surfer.	A surfer is outside watching a crowd of people.
A lady is holding a viola in the woods.	A viola is holding a lady in the woods.
A girl in striped swimsuit is jumps into the ocean to catch fish	Fish is jumps into the ocean to catch a girl in striped swimsuit
A person is training a choir for the upcoming competition.	For the upcoming competition is training a choir has been person
The photographer gathers the bridal party before the ceremony.	The bridal party is gathering the photographer before the ceremony

Table 10: Examples of augmented data in NLI augmentation experiments (§6). We use original SNLI hypotheses as premises in the augmented data and use `SWAP_CORE` with Tailor to generate new hypotheses.

checkpoint with the lowest validation loss.[15]

The drops in predictors' accuracies on the Tailor-generated contrast sets (compared to original test accuracies) show that they can be used to reveal model errors not reflected in original validation data. However, this result should be interpreted with caution, as it is not directly reflective of dataset quality. For instance, if the contrast data tests one error type or is adversarially constructed to include instances where predictors fail, then lower accuracy does not necessarily mean exposing more model errors. Thus, we treat these performance metrics as secondary to other direct metrics of dataset quality, discussed in §5, and run this analysis on a small number of contrast set instances as a sanity check. That said, the fact that predictors perform poorly on Tailor-generated contrast sets even without including an adversarial component in our contrast set creation suggests that Tailor can be useful for creating evaluation data to find model errors.

E Data Augmentation Details (§6)

Augmented data. To create our augmented data, we filter generations by perplexity scores from GPT-2 such that we retain 75% of generations. Examples of augmented inputs are shown in Table 10.

Classifiers. We train all SNLI classifiers, which build on RoBERTa-base (Liu et al., 2019), using `AllenNLP` (Gardner et al., 2018). We train for 10 epochs using the Adam optimizer with a learning rate of 2e-05 and batch size 32; we use early stopping with a patience of 3.

[15]For MATRES, we format inputs by surrounding verbs with marker "<el>" and "</el>" and train the predictor to output the label in natural language, *e.g.,* "Mr. Erdogan has long <el> sought </el> an apology... After that raid An Israeli raid on this ship <el> left </el> nine passengers dead..." → "before".

F Tailor's fine-grained and compositional perturbations on StylePTB

Here, we show how Tailor can be applied to fine-grained style transfer. We evaluate Tailor without any finetuning[16] on the StylePTB benchmark (Lyu et al., 2021), which builds on the Penn Treebank and assesses fine-grained stylistic changes, both on *single* transfers (*e.g., To Future Tense*) and compositional ones that concurrently edit multiple stylistic dimensions (*e.g., To Future Tense+ Active To Passive*).

Transfers Evaluated. We evaluate on the transfers in StylePTB for which Lyu et al. (2021) report results, as their baselines require training separate models for each transfer. Within this subset of transfers, we exclude *PP Back to Front* and *Passive to Active* from evaluation, as they contain < 5 test inputs. We also exclude the transfers *Substatement Removal*, *Information Addition*, *Adjective Emphasis*, and *Verb/Action Emphasis*, for which our semantic-role-derived inputs are not well-suited. For example, *Substatement Removal* involves removing substatements that represent "referring" and "situations," both of which are technical philosophical concepts that cannot be straightforwardly detected through semantic roles. As another example, *Information Addition* requires adding unordered keyword contents to a sentence (eg *the work force provides the third arm of the alliance*;

[16]This evaluation is zero-shot in spirit, as Tailor is not trained on any paired transfers present in StylePTB. However, it is unclear if the test inputs in StylePTB overlap with the Ontonotes 5.0 training data, since the two do share some data points (van Son et al., 2018), and StylePTB does not seem to preserve original PTB splits. This leakage may advantage the external SRL predictor in parsing StylePTB test inputs. Still, this advantage should be minor, as the evaluated transfers do not require complex semantic role parsing.

(a) Single transfers	Single Finetune		Compos. Finetune		No Finetune	
	GPT-2	RetrieveEdit	CS-GPT-TV	CS-GPT-TP	Tailor	Tailor, Filtered
To Future Tense	89.5	**89.9**	72.7	81.0	87.3	88.9, 357/364
To Past Tense	83.6	**93.5**	69.4	83.4	88.4	89.3, 216/218
To Present Tense	75.4	**90.9**	73.3	82.6	71.0	84.7, 175/209
ADJ or ADV Removal	64.7	**89.7**	—	—	78.1	84.3, 224/243
PP Front to Back	39.8	54.1	—	—	**84.2**	96.9, 20/23
PP Removal	76.3	**79.8**	—	76.0	71.7	85.7, 199/238
Active to Passive	47.6	**68.1**	47.2	—	55.6	77.8, 98/137

(b) Compositional transfers		Compos. Finetune	Multi-Single Finetune	No Finetune	
		CS-GPT*	CS-Sys-Gen*	Tailor	Tailor, Filtered
Tense + Voice	ToPast+ActiveToPassive	40.9	33.7	**66.0**	66.0, 30/30
	ToFuture+ActiveToPassive	**49.6**	41.9	46.8	67.0, 90/131
	ToFuture+PassiveToActive	52.8	39.9	**68.3**	68.3, 131/131
	ToPast+PassiveToActive	47.4	36.5	**70.2**	70.2, 65/65
	ToPresent+PassiveToActive	52.3	42.4	**69.9**	69.9, 95/95
	ToPresent+ActiveToPassive	**50.3**	44.5	31.5	61.4, 43/84
Tense + PPRemoval	ToFuture+PPRemoval	73.8	46.5	**74.3**	79.2, 215/229
	ToPast+PPRemoval	**77.2**	54.2	73.8	79.7, 100/108
	ToPresent+PPRemoval	**70.9**	54.5	69.1	70.4, 153/156

Table 11: BLEU scores for single and compositional style transfers in StylePTB. Baseline results are taken from Tables 14-16 and 19-20 in Lyu et al. (2021). * represents the same type of models finetuned on different subsets of styles, *e.g.,*CS-GPT* in (b) includes CS-GPT-TV, trained on all *Tense+Voice* compositional transfers, and CS-GPT-TP, on *Tenses+PP Removal*. A single Tailor model helps achieve comparable performance on single transfers compared to finetuned baselines, and is more capable on multiple compositional transfers.

add keywords: *force black* → *the work force provides the third arm of the* ***black*** *alliance* ***force***. While the Tailor generator was only trained with ordered arguments, one could extend the keyword contents to also include unordered target tokens.

Perturbation strategies. For transfers modifying only verb tense (*e.g., To Future Tense*), we mask the verb, modal arguments, and negation arguments, as these are relevant to verb conjugations, and make relevant perturbations on the secondary verb control specifying tense. For transfers modifying verb voice, we mask the verb, agent, and patient. For transfers requiring removal of certain parts of speech (POS)—*i.e., ADJ or ADV Removal*, *PP Removal*, and all compositional *Tense + PP Removal* sub-transfers —we first use `spacy` to detect such POS, next mask all arguments containing them, and finally perturb the keyword contents to remove the POS for these arguments. For *PP Front to Back*, we mask the argument at the beginning of the original text and implement the change using `CHANGE_IDX`.

We use cased keywords (A.2) to encourage generations with similarly ordered arguments as the original sentence, except for the *PP Front to Back* transfer, which calls for differently ordered arguments. For transfers modifying verb form only, we set the number of extra blanks to be 2 to allow for generation of helper verbs; for other transfers, we allow for 0 extra blanks to preserve the original order of generated spans. We decode perturbed sentences greedily using beam search (with beam width 10) and preventing repeated bigrams.

For each transfer, we create perturbations for each predicate in the original input, and report mean BLEU scores.[17] Because this process results in multiple perturbations (one per verb), we choose the one with the lowest perplexity from GPT-2 to represent the transfer. Unsuccessful transfers, either due to a failure of perturbation strategy (*e.g.,* no verbs are found by our SRL predictor) or due to a degenerate output (see §C), are given a BLEU score of 0.0.

Baselines. We work with baselines reported by Lyu et al. (2021): **GPT-2** and **RetrieveEdit** are the best-performing single-transfer models evaluated but require separate models to be trained for each transfer. **CS-GPT*** are models trained on compositional subsets of data (*e.g., Tense+Voice*, detailed in Table 11 caption). **CS-Sys-Gen** are ablations of CS-GPT* trained only on corresponding individual changes but evaluated on compositional transfers.[18]

Result. On compositional transfers, we find that Tailor outperforms the baseline system trained

[17] We report `Bleu_1` from `nlg-eval` (Sharma et al., 2017).
[18] CS-Sys-Gen refers to CS-GPT-Zero in Lyu et al. (2021).

without compositional fine-tuning, CS-SYS-GEN, on 8/9 compositions, and even outperforms CS-GPT* — models with compositional finetuning — on 5 cases. It also achieves compatible or better results than GPT-2 and RETRIEVEEDIT on single transfers. Low TAILOR performance on some transfers (*e.g., ToFuture+ActiveToPassive*) appears to be driven by unsuccessful transfers, rather than generations that do not follow controls, as indicated by the higher performances on the subset where unsuccessful transfers are removed (*Filtered Test*). Importantly, TAILOR achieves these gains with a *single model* and *without any transfer-specific finetuning*.