Tailor: Generating and Perturbing Text with Semantic Controls

Anonymous ACL submission

Abstract

Making controlled perturbations is essential for various tasks (*e.g.,* data augmentation), but building task-specific generators can be expensive. We introduce Tailor, a task-agnostic generation system that perturbs text in a semantically-controlled way. With unlikelihood training, Tailor's generator is designed to follow a series of control codes derived from semantic roles. Through modifications of these control codes, Tailor can produce fine-grained perturbations. We implement a set of operations on control codes that can be composed into complex perturbation strategies, and demonstrate their effectiveness in three applications. First, Tailor facilitates the construction of high-quality contrast sets that are lexically diverse and less biased than original task test data. Second, paired with automated labeling heuristics, Tailor helps improve model generalization through data augmentation: we obtain an average gain of 1.73 on an (natural language inference) NLI challenge set by perturbing just ~5% of training data. Third, without any finetuning overhead, Tailor's perturbations effectively improve compositionality in fine-grained style transfer, outperforming fine-tuned baselines on 5 transfers.

1 Introduction

Controllable text generation through semantic perturbations, which modifies sentences to match certain target attributes, has been widely applied to a variety of tasks, *e.g.,* changing sentence styles (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 work trains controlled generators with task-specific data, *e.g.,* training a style transferer requires instances labeled with *positive* and *negative* sentiments (Madaan et al., 2020b). As a result, transferring to a new application is prohibitive, and

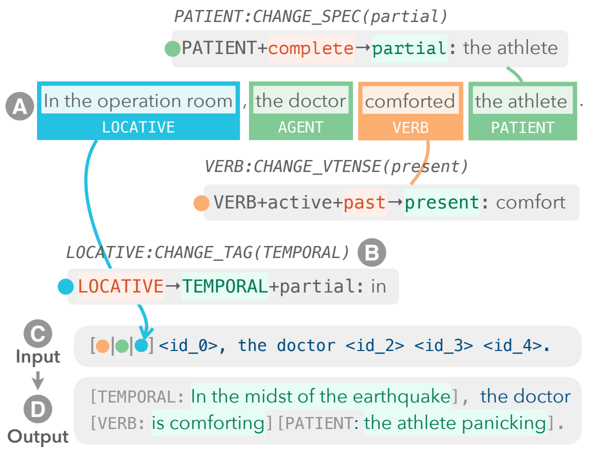
PATIENT:CHANGE_SPEC(partial)
PATIENT+complete→partial: the athlete
A
In the operation room
LOCATIVE
the doctor
AGENT
comforted
VERB
the athlete
PATIENT
VERB:CHANGE_VTENSE(present)
VERB+active+past→present: comfort
LOCATIVE:CHANGE_TAG(TEMPORAL)
B
LOCATIVE→TEMPORAL+partial: in
C
Input
[●|●|●] <id_0>, the doctor <id_2> <id_3> <id_4>.
D
Output
[TEMPORAL: In the midst of the earthquake], the doctor [VERB: is comforting] [PATIENT: the athlete panicking].

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.

requires costly annotation efforts and re-training for every task of interest.

In this work, we introduce Tailor, a system that supports application-agnostic perturbations without the need for retraining. At the core of Tailor is a *controlled generator* (§2) that flexibly generates full sentences from target semantic features. We combine structured **control codes** in our **inputs** to represent desired linguistic properties of outputs. As shown in Figure 1, each code builds on the PropBank semantic parse (Palmer et al., 2005) of the original sentence, and specifies a semantic role argument span. We use **unlikelihood training** (Welleck et al., 2020) to encourage control code following, by penalizing generations that are not aligned with designated codes.

The multi-dimensionality of semantic roles al-

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

lows Tailor to perform fine-grained changes to individual arguments in a sentence (*e.g.,* one can just change the patient in Figure 1). This is critical for generating datasets that evaluate and improve models' language understanding (Kaushik et al., 2020; Wu et al., 2021). Instead of representing a change with a single target property *positive*→*negative*, we can decompose it into specific linguistic transformations (*e.g.,* changing sentiment polarity through negation or antonym replacement).

To highlight perturbations that are feasible with Tailor, we identify and implement a list of primary *perturbation operations* (§3) on inputs to the generator; these can be easily composed to achieve more complex perturbations. Take Figure 1 as an example: while it would be nontrivial to train a generator to directly transform sentence A to D, it can be achieved through the composition of a series of perturbations: syntactic rewriting (changing verb tense), then sentence expansion (extending "the athlete"), and finally data recombination (*i.e.,* sourcing text with the `TEMPORAL` control).

Tailor's flexible control codes allow for broad, easily extendable applicability. We demonstrate Tailor's utility in three distinct applications: 1) We use Tailor to replicate existing **contrast sets** (§5) on four diverse tasks, with much less manual annotation effort. Our analysis suggests that these contrast sets not only have high rates of validity, but also promote lexical diversity and reduce dataset bias. 2) Augmenting training data with just a small ratio of Tailor perturbations (~5%) **improves the robustness** of natural language inference (NLI) models to inference heuristics, increasing performance on the HANS evaluation set by an average of 1.73 points (McCoy et al., 2019). 3) Without any finetuning, Tailor achieves impressive performance on **fine-grained and compositional style transfer** (§7) in the StylePTB benchmark (Lyu et al., 2021), even outperforming models trained on the dataset on 5 transfers.

2 Tailor's Controllable Generator

We provide an overview of the Tailor generator. We first outline the controllable **dimensions** useful for semantic perturbations (§2.1), and then explain how to embed them within **inputs** to the generator (§2.2). Finally, we describe how we use **unlikelihood training** to train our generator to follow the controls (§2.3).

2.1 Controllable Dimensions

To allow for control over sentence semantics at varying levels of granularity, we incorporate a combination of semantic roles and content keywords.

To denote shallow semantics, we use the PropBank semantic formalism, which represents sentences' meanings with predicate-argument structures (Palmer et al., 2005). Predicates reflect events (*what happened*), and are usually evoked by verbs, like "comforted" in Figure 1. Arguments, usually spans of tokens, realize the thematic roles of the predicates, including *core* arguments such as *who* (*e.g.,* "the doctor") and *to whom* ("the athlete"), as well as *adjunct* arguments like *where* ("In the operation room"), *how*, etc. PropBank semantic analyses provide well-established feature representations for meanings and are generalizable across different verb predicates and languages (Hajič et al., 2009), making it an appealing choice for representing high level semantics.

We further use *content keywords* to drive the generation of actual predicates and arguments. Depending on to what extent we would like to retrieve new text from the generator, 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 use cases.

Since the same set of thematic roles can be combined in different ways, we add further controls on *span ordering*. 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 on the predicate, 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 aim to integrate the aforementioned controls into an input format, and finetune language models to output full sentences reflecting them.

As shown in Table 1, we start our input with a bracketed *header*, a series of abstract *control codes* (Table 2) with each denoting the seman-

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

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

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 can accept, 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.

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: Overview of TAILOR's control codes. Primary controls build on predicate/argument labels, and others further affect the form and content of generations.

tic 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 to be infilled.

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 when 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. More details about input formats are in §A.1.

2.3 Training

We create our generator by finetuning T5-BASE (Raffel et al., 2020) on pairs of inputs and outputs derived from the gold semantic roles in OntoNotes 5.0 train (Pradhan et al., 2013), as in Table 1. In order to make our generator sensitive to the different input formats described in the previous section, for each original input, we randomly sample the number of arguments to mask, number of extra empty blanks, and keyword content/specificity for each role (details in §A.2).

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

In order to encourage reliance on controls, we incorporate **unlikelihood training** (Welleck et al., 2020) to penalize our generator for generating text that conflicts with inputs. That is, besides Table 1A–

(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.

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 anymore. As detailed in §A.1, we create three negative samples per input, which randomly perturb: 1) verb voice/tense and primary controls for arguments, 2) keyword contents, and 3) keyword specificities. After data processing, our training data consists of 223,619 positive and 541,424 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 to the input header.[3] Then, we modify the controls in this derived input to generate perturbed sentences with TAILOR, filtering out degenerate ones. We detail the changes on the controls below.

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* adds or removes text fragments from a sentence (Wu et al., 2021). We recreate these through deletions 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.

These primitive perturbation operations can be used in conjunction with external knowledge bases to achieve targeted edits.[4] Additionally, these operations can be composed to achieve more complex perturbation strategies, as shown in §5, §6, and §7.

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 §F for details).

4 Intrinsic Evaluation

Following POLYJUICE (Wu et al., 2021) and MiCE (Ross et al., 2020), we evaluate TAILOR generations on fluency, controllability, and closeness.[5]

Metrics. *Fluency* 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 aim for a value of 1.0, which indicates equivalent losses for the original and edited texts.

[3]External semantic role labelers can be used when gold annotations are not available. Our experiments use the open-sourced implementation of Shi and Lin (2019): https://demo.allennlp.org/semantic-role-labeling.

[4]For example, if combined with WordNet (Miller, 1998), TAILOR perturbations can recreate natural logic (MacCartney and Manning, 2014): In Table 3, doctor→adult creates an entailment relationship, with "doctor" a hyponym of "adult."

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

Generator	Closeness with the original			Controllability on predicates			Controllability on arguments		
	F1	Precision	Recall	Lemma	Tense	Voice	Role	Content	Specificity
Tailor	**64.3%**	**66.5%**	**73.4%**	**74.3%**	**80.3%**	**81.6%**	**70.5%**	**64.5%**	**64.5%**
Tailor$_{\text{MLE}}$	58.5%	59.5%	68.6%	72.2%	70.2%	76.1%	60.3%	45.1%	45.1%

Table 4: Tailor generates perturbations that are close to the original sentence, while reasonably following all the controls specified in Table 2. Through an ablation study where unlikelihood training is removed (Tailor$_{\text{MLE}}$), we see that the controllability and closeness are both core benefits from unlikelihood training.

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 inspected a subset of 98 generated spans, and verified that the cycle consistency measures positively correlate with true controllability measures (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 weigh spans by their lengths to arrive at the final F1.

Results. We evaluate Tailor by perturbing 1,000 randomly selected sentences from the OntoNotes 5.0 development set, created the same way as we create negative samples during training (details in §A.1).[6] Tailor generates fluent 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 (75%-80% of the time) and arguments (with 70% controllability on semantic roles, and ~65% on contents.) Through an ablation study comparing Tailor with a baseline that is fine-tuned on T5 *without* unlikelihood training (called Tailor$_{\text{MLE}}$), we show that unlikelihood training encourages controls and minimal perturbations, with the metrics increasing by up to 20%.

Further, as mentioned in §2.2, our input format supports modulating fluency and closeness at generation time. In §B, we quantify the effects of masking subsets of arguments or including more empty blank tokens on closeness and fluency.

5 Application 1: Contrast Set Creation

We use Tailor to replicate contrast and challenge sets for a variety of NLP tasks, including question answering (BoolQ: Clark et al., 2019; 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

As shown in Table 5, we take advantage of two key properties of Tailor:[7] First, Tailor can make perturbations that are **context-dependent**. 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 verb to noun to recreate the *UD Parsing contrast set* through 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").

Manually creating contrast sets is expensive,[8] whereas validating existing ones is more efficient (Wu et al., 2021). We consider our perturba-

[6]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.

[7]Details on implementing perturbation strategies are in §C.

[8]*e.g.,* Gardner et al. (2020) reported spending 10-15 minutes per perturbation for UD Parsing

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 (PP attachment) (Gardner et al., 2020)		65% (k=10)
Original	**Sentence:** Do [AGENT: you] [VERB: prefer] [PATIENT: ham or sausages] [ADVERBIAL: with your breakfast]? **PP attachment:** Verb ("with your breakfast" attaches to "prefer")	
Strategy	Swap attachment from verb to noun (*verb→noun*) `PATIENT:CHANGE_CONTENT(ham or sausages with),CHANGE_SPEC(partial);ADVERBIAL:DELETE`	
Perturb.	**Sentence:** Do [AGENT: you] [VERB: prefer] [PATIENT: ham or sausages with bacon on them]? **PP attachment**: Noun ("with bacon on them" attaches to "sausages")	
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 §C.

tion 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.[9] Table 5 shows that these TAILOR perturbation strategies generate contrast sets with high validity.[10]

5.2 Measuring Contrast Set Quality

We assess 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 original UD instances: we randomly sample one contrastive 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. The ratios are 0.783 and 0.883 for TAILOR and humans, respectively, for *noun→verb*, and are both 1.0 for *verb→noun*. Thus, TAILOR can help generate contrast sets without significantly reducing lexical diversity. TAILOR generations are also distinguishable from humans': their unique tokens only overlap for < 15% in *verb→noun*, and ~6% for *noun→verb*, suggesting that TAILOR can work as a collaborative tool to diversify the pool of tokens.

Gardner et al. (2021) show that making minimal perturbations reduces single-feature artifacts when $(1 + e_i)/s = 2$, where e_i is the probability that feature i is edited, and s is the probability that an edit changes the label. We manually label the same number of TAILOR-perturbed examples as in the original BoolQ contrast set, and find that TAILOR produces edits with an average value of $(1+e_i)/s = 1.74$, which is close to that produced by humans (1.94). Thus, making perturbations with TAILOR can help mitigate dataset biases (visualization in §C).

6 Application 2: Data Augmentation

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

Following Min et al. (2020), we create augmented data by perturbing SNLI hypotheses, such that *original hypothesis→premise* and *perturbed hypothesis→hypothesis*. We define five perturbation strategies for NLI (§D), all of which express high lexical overlap, an inference heuristic on which NLI models have been shown to rely (Dasgupta et al., 2018; Naik et al., 2018). These perturbations either preserve or alter the meaning

[9]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.

[10]As expected, TAILOR achieves higher validity changing PP attachment types *noun→verb* (82%) than *verb→noun*, as the arguments by design attach to verb predicates, while noun attachment is not an explicit part of the training objective and is therefore harder for the generator.

Training Data	SNLI	HANS Subset All	HANS Subset Entail.	HANS Subset Non-Entail.
SNLI Train	**91.12**	64.72	**98.95**	30.46
+ Tailor Perturb.	**91.12**	**66.45**	97.97	**34.92**

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

of original hypotheses. For example, we change sentence meaning by replacing keywords of core arguments with noun chunks of other arguments (*The judge behind the manager saw the doctors.* → *The doctors saw the manager.*) Following Min et al. (2020), we map meaning-preserving perturbations to label *entailment* and others to *neutral*.

We train classifiers built on RoBERTa-base (Liu et al., 2019) on different subsets of data: original SNLI train data (baseline) and SNLI train data with ~5% of hypotheses augmented with Tailor perturbations.[11] For each subset, we train 20 models, each with a different random seed. We evaluate each classifier on the in-domain SNLI test set and the out-of-domain HANS test set (McCoy et al., 2019), which is designed to diagnose inference heuristics built on superficial syntactic properties.[12]

As shown in Table 6, the augmentation leads to an out-of-distribution gain of **+1.73** points on overall HANS and **+4.46** points on the "non-entailment" subset. The gains are significant, with $t = -3.26$, $p = 0.002$ using Student's t-test. Thus, Tailor perturbations decrease reliance on a well-known, lexical-overlap inference heuristic for NLI.

7 Application 3: Style Transfer

Here, we show how Tailor can be applied to style transfer. We evaluate Tailor without any finetuning[13] 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*).

We evaluate Tailor on transfers for which Lyu et al. (2021) show model results in the paper, excluding some that our semantic-role-derived inputs are not well-suited (see §E). For each transfer, we create perturbations for each predicate in the original input, and report mean BLEU scores.[14] 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 §F), are given a BLEU score of 0.0.

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 7 caption). **CS-Sys-Gen** are ablations of CS-GPT* trained only on corresponding individual changes but evaluated on compositional transfers.[15]

We report a subset of the comparisons in Table 7 (b), and the full result in Appendix E. On compositional transfers, we find that Tailor outperforms the baseline system trained 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*.

8 Related Work

Controllable text generation has been widely used to influence various properties of generated text for text summarization (Peng et al., 2019), 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), topic (Keskar et al., 2019), and sen-

[11] We augment the original 549,367 SNLI train instances with 30,147 total new instances. See §D for more details.

[12] For HANS, which contains binary labels, we collapse *neutral* and *contradiction* predictions to *non-entailment*.

[13] 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.

[14] We report `Bleu_1` from `nlg-eval` (Sharma et al., 2017).

[15] CS-Sys-Gen refers to CS-GPT-Zero in Lyu et al. (2021).

(a) Single transfers	Single Finetune GPT-2*		Compos. Finetune CS-GPT-TV		No Finetune TAILOR	
	GPT-2*	RETRIEVEEDIT*	CS-GPT-TV	CS-GPT-TP	TAILOR	
	Test	Test	Test	Test	Test	Filtered Test
To Future Tense	0.895	**0.899**	0.727	0.810	0.873	0.889 (357/364)
ADJ or ADV Removal	0.647	**0.897**	—	—	0.781	0.843 (224/243)
PP Front to Back	0.398	0.541	—	—	**0.842**	0.969 (20/23)
Active to Passive	0.476	**0.681**	0.472	—	0.556	0.778 (98/137)

(b) Compositional transfers		Compos. Finetune	Multi-Single Finetune	No Finetune	
		CS-GPT*	CS-SYS-GEN*	TAILOR	
		Test	Test	Test	Filtered Test
Tense + Voice	ToPast+ActiveToPassive	0.409	0.337	**0.660**	0.660 (30/30)
	ToPast+PassiveToActive	0.474	0.365	**0.702**	0.702 (65/65)
	ToPresent+ActiveToPassive	**0.503**	0.445	0.315	0.614 (43/84)
	ToPresent+PassiveToActive	0.523	0.424	**0.699**	0.699 (95/95)
Tense + PPRemoval	ToPast+PPRemoval	**0.772**	0.542	0.738	0.797 (100/108)
	ToFuture+PPRemoval	0.738	0.465	**0.743**	0.792 (215/229)

Table 7: BLEU scores for a subset of single and compositional style transfers in STYLEPTB (more in §E). 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.

timent polarity (Dathathri et al., 2020), 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 representation 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, Huang and Chang (2021) design 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.

Also related are prior works creating minimally edited datasets through extensive human efforts, either through manual rewriting (Gardner et al., 2020; Kaushik et al., 2020), or perturbation functions and templates (*e.g.,* (Andreas, 2020; Li et al., 2020; Ribeiro et al., 2020; Wu et al., 2019)).

9 Conclusion

We propose TAILOR, a flexible system that enables complex and context-aware perturbations useful for various downstream applications. TAILOR demonstrates that it is possible to drive fine-grained perturbations with semantic features directly derived from an instance. Crucially, it also 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.,* AMR and syntactic roles as in §8).

While being widely applicable, TAILOR's effectiveness varies for different inputs. For example, some inputs derived from SRL predictors may miss rare semantic roles; Fortunately, this did not seem to be a bottleneck, as empirically most tasks modify common arguments already recognized by the predictors. 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 (as detailed in §F). Having noted these opportunities, we believe TAILOR is already a powerful tool for perturbations, and we opensource it at `[URL omitted]`.

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.

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.

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.

Ishita Dasgupta, Demi Guo, Andreas Stuhlmüller, S. Gershman, and Noah D. Goodman. 2018. Evaluating compositionality in sentence embeddings. *ArXiv*, abs/1802.04302.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2020. Plug and play language models: A simple approach to controlled text generation. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

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.

Aakanksha Naik, Abhilasha Ravichander, Norman Sadeh, Carolyn Rose, and Graham Neubig. 2018. Stress test evaluation for natural language inference. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2340–2353, Santa Fe, New Mexico, USA. 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*.

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, Minneapolis, Minnesota. Association for Computational Linguistics.

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*.

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.

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 of arguments. If a subset, we uniformly select the proportion of arguments to mask. To determine the number of extra blank tokens, 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 blank tokens (*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).[16]

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 epochs with early stopping. We use batch size 4 and default values for other parameters (learning

[16]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.

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.

Modulating fluency and closeness. As mentioned in §2.2, our input format supports modulating fluency and closeness at generation time. We can increase closeness by only masking the arguments we want to perturb. To quantify this effect, we randomly select only one argument to perturb for 1,000 sentences, but vary the number of arguments masked, and the number of empty blanks inserted. We maximize closeness 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 fluency by adding more empty blank token (*e.g.,* when we insert extra roles whose optimal locations are not known in advance). We experiment with this setting on another 1,000 sentences, and observe that adding six extra blanks increases the fluency ratio from 0.93 to 0.95.

C Contrast Set Details (§5)

In Table 8, we illustrate our perturbation procedures for creating contrast sets. Besides BoolQ and UD English [17] 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 that are about different targets.

As mentioned in §5, the Tailor-generated contrast sets contain fewer artifacts compared to the original BoolQ validation set. Here, we provide a straightforward visualization to show the effect. As shown in Figure 2, many tokens in the original BoolQ validation data are biased towards the positive class (with the red dots distributed in the > 0.5 region), while most tokens in the edited set fall within the confidence region denoting no significant feature-level biases.

[17]For UD Parsing contrast set generation, we use constrained decoding (Hokamp and Liu, 2017) to prevent generation of the original prepositional phrase.

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] 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] 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 8: 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.

Meaning Preserving Strategies	
Untangle relative clause: For verbs with args containing relative clauses (*i.e.,* roles with *R*-), delete context.	
Original	The [PATIENT: athlete] who was [VERB: seen] [AGENT: by the judges] [TEMPORAL: yesterday] called the manager
Edits	CONTEXT(DELETE_TEXT) → [VERB+passive+past: see | AGENT+complete: by the judges | PATIENT+complete: the athlete | TEMPORAL+complete: yesterday] <id_0> ~~who~~ <id_1> <id_2> <id_3> <id_4> ~~called the manager~~
Perturb.	The [PATIENT: athlete] was [VERB: seen] [AGENT: by the judges] [TEMPORAL: yesterday]
Shorten core: Change keywords for core args to root of original arg spans.	
Original	The [AGENT: athlete who was seen by the judges yesterday] [VERB: called] [PATIENT: the manager].
Edits	AGENT:CHANGE_CONTENT(The athlete ~~who was...~~) → [VERB+active+past: call | AGENT+complete: The athlete ~~who was seen by the judges yesterday~~ | PATIENT+complete: the manager] <id_0> <id_1> <id_2> <id_3> <id_4>
Perturb.	The [AGENT: athlete] [VERB: called] [PATIENT: the manager].
Change voice: Swap active/passive verb controls.	
Original	The [AGENT: athlete who was seen by the judges yesterday] [VERB: called] [PATIENT: the manager].
Edits	VERB:CHANGE_VOICE(passive)|AGENT:CHANGE_CONTENT(by the athlete who was...) → [VERB+active→passive+past: call | AGENT+complete: by the athlete who was seen by the judges yesterday | PATIENT+complete: the manager] <id_0> <id_1> <id_2> <id_3> <id_4>
Perturb.	[PATIENT: The manager] was [VERB: called] [AGENT: by the athlete who was seen by the judges yesterday].
Meaning Changing Strategies	
Replace core with subsequences: Change keywords of core args to noun chunks from other args.	
Original	[AGENT: The judge behind the manager] [VERB: saw] [PATIENT: the doctors].
Edits	[VERB+active→passive+past: call | AGENT+complete: by the athlete who was seen by the judges yesterday | PATIENT+complete: the manager] <id_0> <id_1> <id_2> <id_3> <id_4>
Perturb.	[AGENT: The doctors] [VERB: saw] [PATIENT: the manager].
Swap core: Swap agent/patient.	
Original	[PATIENT: The athlete] who was [VERB: seen] [AGENT: by the judges] called the manager.
Edits	SWAP_CORE → [VERB+passive+past: see | AGENT+complete: by the judges→athlete | PATIENT+complete: by the athlete→judges] <id_0> <id_1> <id_2> <id_3> <id_4>
Perturb.	[PATIENT: The judges] who were [VERB: seen] [AGENT: by the athlete] called the manager.

Table 9: Overview of perturbation strategies we apply to SNLI hypotheses in our augmentation experiments (§6).

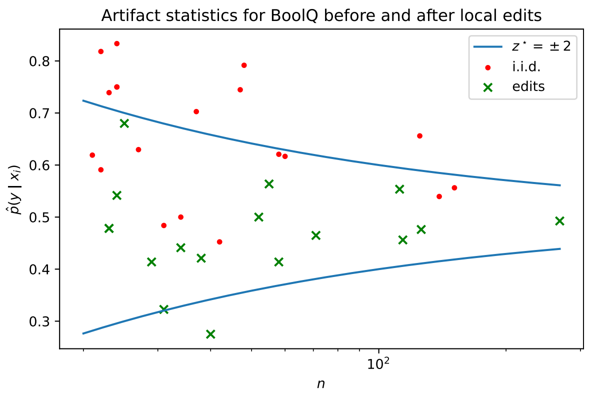

Figure 2: A comparison of the dataset artifacts in the original BoolQ validation set and contrast set created with Tailor. The figure is plotted in the same way as Figure 2 in (Gardner et al., 2021).

D Data Augmentation Details (§6)

Augmented data. Our five perturbation strategies are shown in Table 9. To create our augmented data, we first filter generations by perplexity scores from GPT-2 such that we retain 75% of generations. Then, for each hypothesis we perturb, we uniformly sample a successful perturbation. (An example of a failed perturbation would be one requiring both agent/patient roles, applied to a sentence without both roles.) This process results in a slight skew towards *entailment* labels (*i.e.,* ≈ 2.75:1, *entailment:neutral*). Future work can investigate to what extent label imbalance affects augmentation results.

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.

E Style Transfer Details (§7)

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*; 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.

F Degenerate Outputs

We observe that Tailor produces degenerate outputs for some inputs, as shown in Table 11. 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

(a) Single transfers	Single Finetune		Compos. Finetune		No Finetune	
	GPT-2	RetrieveEdit	CS-GPT-TV	CS-GPT-TP	Tailor	
	Test	Test	Test	Test	Test	Filtered Test
To Future Tense	0.895	**0.899**	0.727	0.810	0.873	0.889 (357/364)
To Past Tense	0.836	**0.935**	0.694	0.834	0.884	0.893 (216/218)
To Present Tense	0.754	**0.909**	0.733	0.826	0.710	0.847 (175/209)
ADJ or ADV Removal	0.647	**0.897**	—	—	0.781	0.843 (224/243)
PP Front to Back	0.398	0.541	—	—	**0.842**	0.969 (20/23)
PP Removal	0.763	**0.798**	—	0.760	0.717	0.857 (199/238)
Active to Passive	0.476	**0.681**	0.472	—	0.556	0.778 (98/137)

(b) Compositional transfers		Compos. Finetune	Multi-Single Finetune	No Finetune	
		CS-GPT*	CS-Sys-Gen*	Tailor	
		Test	Test	Test	Filtered Test
Tense + Voice	ToPast+ActiveToPassive	0.409	0.337	**0.660**	0.660 (30/30)
	ToFuture+ActiveToPassive	**0.496**	0.419	0.468	0.670 (90/131)
	ToFuture+PassiveToActive	0.528	0.399	**0.683**	0.683 (131/131)
	ToPast+PassiveToActive	0.474	0.365	**0.702**	0.702 (65/65)
	ToPresent+PassiveToActive	0.523	0.424	**0.699**	0.699 (95/95)
	ToPresent+ActiveToPassive	**0.503**	0.445	0.315	0.614 (43/84)
Tense + PPRemoval	ToFuture+PPRemoval	0.738	0.465	**0.743**	0.792 (215/229)
	ToPast+PPRemoval	**0.772**	0.542	0.738	0.797 (100/108)
	ToPresent+PPRemoval	**0.709**	0.545	0.691	0.704 (153/156)

Table 10: The full stylePTB results, extending Table 7.

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

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 11, 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.