Controlled Text Generation with Natural Language Instructions

Wangchunshu Zhou [1] **Yuchen Eleanor Jiang** [1] **Ethan Wilcox** [1] **Ryan Cotterell** [1] **Mrinmaya Sachan** [1]

Abstract

Large language models can be prompted to produce fluent output for a wide range of tasks *without* being specifically trained to do so. Nevertheless, it is notoriously difficult to control their generation in such a way that it satisfies user-specified constraints. In this paper, we present INSTRUCTCTG, a simple controlled text generation framework that incorporates different constraints by verbalizing them as natural language instructions. We annotate natural texts through a combination of off-the-shelf NLP tools and simple heuristics with the linguistic and extra-linguistic constraints they satisfy. Then, we verbalize the constraints into natural language instructions to form weakly supervised training data, i.e., we prepend the natural language verbalizations of the constraints in front of their corresponding natural language sentences. Next, we fine-tune a pre-trained language model on the augmented corpus. Compared to existing methods, INSTRUCTCTG is more flexible in terms of the types of constraints it allows the practitioner to use. It also does not require any modification of the decoding procedure. Finally, INSTRUCTCTG allows the model to adapt to new constraints without re-training through the use of in-context learning. Our code is available at `https://github.com/MichaelZhouwang/InstructCTG`.

1. Introduction

Large language models (LLMs) pre-trained on web-scale corpora (Radford et al., 2018; 2019; Brown et al., 2020) have demonstrated the ability to generate fluent and realistic text. Yet, many text generation applications require the model to generate text that not only possesses a high degree of naturalness but also adheres to task-specific constraints. For instance, a question generation system may require that the model incorporates certain keywords during generation, and a dialog system may be expected to generate responses that have a certain tone.

The dominant paradigm for controlled text generation involves incorporating the constraints into the decoding algorithm itself. Prior work either includes constraints in the search algorithm by modifying beam search (Hokamp and Liu, 2017; Anderson et al., 2017; Post and Vilar, 2018; Lu et al., 2021; 2022) or by converting the constraints into classifiers, which are then used to guide the decoding process (Qin et al., 2022; Kumar et al., 2022; Li et al., 2022; Amini et al., 2023). While effective at requiring the model to enforce the constraints on the generated text, existing decoding-time methods for controlled text generation suffer from a number of limitations. First, decoding-time methods modify the output sequence distribution and, thus, can result in lower-quality text (Lin et al., 2020). Second, they may require extra computation and lead to slower generation speed (Post and Vilar, 2018; Li et al., 2022). Third, they are often less able to deal with new kinds of constraints or a novel composition of existing constraints. Finally, they may require practitioners to design new, bespoke features or train additional classifiers to accommodate novel constraints.

In this paper, we present INSTRUCTCTG, a simple paradigm for controlled generation, which presents an alternative to decoding-time-constrained generation methods. In contrast, we adopt a training-time approach. Under INSTRUCTCTG, we construct a corpus in a weakly supervised fashion where each sentence in the corpus is treated as if it had been constructed under the influence of one or more latent constraints. We automatically impute these latent constraints using off-the-shelf NLP tools or simple rules. Then, INSTRUCTCTG incorporates the constraints into generation through a prompt that verbalizes the constraint into a natural language instruction. As a concrete example, in order to generate sentences with a length constraint, we collect a large corpus of sentences and automatically derive the length of each sentence by using a pre-trained tokenizer and counting the number of tokens. Then, in order to conduct controlled generation, we verbalize the length constraints as natural language instructions and instruction-tune the model using the augmented corpus (Wei et al., 2022; Sanh et al., 2022). To avoid the cost of always having to fine-tune for

[1]ETH Zürich. Correspondence to: Wangchunshu Zhou <wangchunshu.zhou@inf.ethz.ch>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

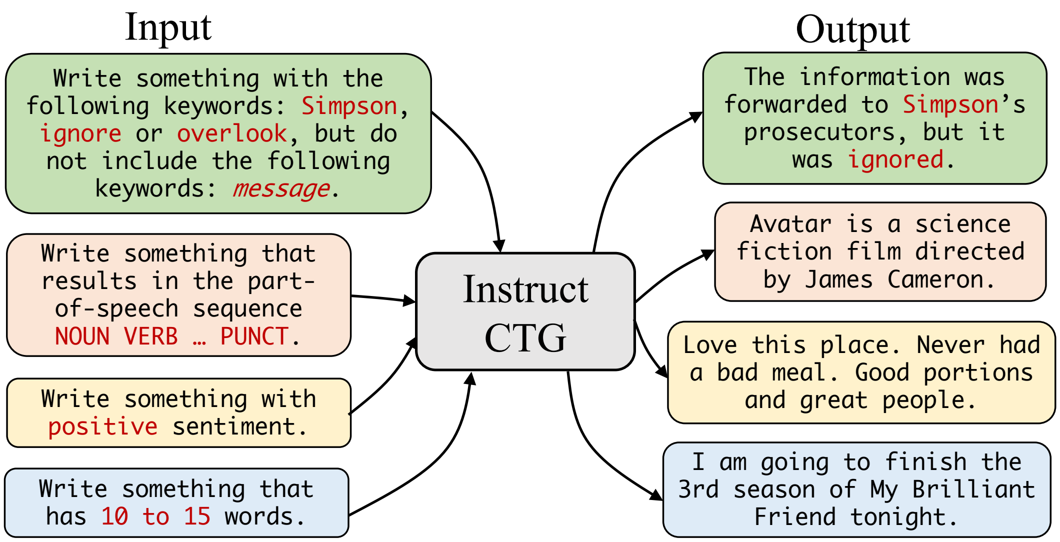

Figure 1: A cartoon of the INSTRUCTCTG framework. INSTRUCTCTG is an instruction-tuned model for controlled text generation. In our experiments, we consider LEXICAL, SYNTAX, SEMANTIC, STYLE, and LENGTH constraints.

every constraint, we also use meta in-context learning (Min et al., 2022) for some constraints where we only make use of a few demonstrations of the constraint–text pairs. Compared to methods that incorporate constraints into the decoding algorithm directly, INSTRUCTCTG gives the model access to the constraints before decoding, leading to improved generation quality and efficiency. A cartoon illustration of INSTRUCTCTG is given in Fig. 1.

To evaluate the effectiveness of INSTRUCTCTG, we conduct a wide range of experiments with different constraint types including keywords, syntax, semantic, style, and length, by fine-tuning T0 (Sanh et al., 2022), an instruction-tuned text-to-text pre-trained model. Our results show that INSTRUCTCTG achieves a high level of constraint satisfaction that is comparable to or better than existing decoding-based methods. The method is also faster in some cases. Additionally, INSTRUCTCTG demonstrates strong generalization abilities in the context of few-shot learning, specifically with regard to the constraints that are unseen during training.

2. INSTRUCTCTG

Our proposed framework, *Instruction-based Controlled Text Generation* (INSTRUCTCTG), is a prompt-based controlled text generation framework. INSTRUCTCTG verbalizes constraints into natural language and uses instructions that consist of the natural language verbalization and a few demonstrations to encourage the model to incorporate the constraints during generation. In this section, we describe the data synthesis method, input format, training method, and inference method of INSTRUCTCTG.

2.1. Data Collection

Typically, instruction tuning requires a large amount of labeled training data. However, most publicly available text generation datasets do not have constraint annotations. And, moreover, crowdsourcing a sufficient number of constraints–text pairs for instruction tuning would be expensive and time-consuming. With this as the state of play, we propose a weakly supervised data synthesis method to synthesize constraint–sentence pairs for INSTRUCTCTG. The main idea behind our data synthesis strategy is to view various kinds of constraints as latent prompts that implicitly inspired the generation of the naturally occurring sentences. However, instead of attempting to infer the latent prompts through, say, a sophisticated Bayesian model, we opt instead to use simple rules and off-the-shelf NLP tools to impute some approximation to the latent prompts. We next describe the constraints used in the training stage of INSTRUCTCTG.

a) Lexical Constraints Following previous work on constrained decoding, we refer to those constraints that require the generated text to include certain keywords in the output text as lexical constraints. To construct lexical constraints, we adopt the keyword extraction pipeline used in prior work on keyword-based text generation (Lin et al., 2020; Zhou et al., 2021). Specifically, given an input text sequence, we use a part-of-speech tagger[1] to extract tokens with the tags that encode some form of a noun or verb from the sentence and randomly sample a portion of them as keywords. We further lemmatize the extracted keywords for consistency across morphological variants. We also design

[1] https://spacy.io/models/en#en_core_web_sm.

a data augmentation method to enable INSTRUCTCTG to handle more complex requirements such as choosing one keyword from a set of keywords or not including a certain word in the output. Specifically, for each keyword, we use a BERT-based lexical substitution system (Zhou et al., 2019) to generate possible alternatives of the keyword in the context. We then create two more complex types of lexical constraints. The first is a disjunctive constraint that requires that the text must include (at least) one of the keywords from the set of the original and the alternative words. The second is an exclusion constraint that requires the text to *not* include any of the proposed alternative keywords. We randomly sample sentences from the C4 (Raffel et al., 2020) dataset to synthesize our keyword constraint–text pairs.

b) Syntactic Constraints We also consider syntactic constraints that require the output text to conform to certain syntactic dictates. Specifically, we focus on two types of syntactic constraints: **part-of-speech** sequence constraints and **syntactic tree** constraints. Our part-of-speech (POS) constraints are sequences of POS tags (e.g., POSS VERB DET NOUN) that require the model to generate a sequence of words of the same length whose POS tags match the target (e.g., "He bought a car"). Similarly, syntactic tree constraints consist of a linearized syntactic parse tree (e.g., (S (NP (PRP $*$)) (VP (VBD $*$) (NP (DT $*$) (NN $*$)))) where $*$ denotes a wild card) and require the model to generate a text sequence that matches the constraint when parsed with a pre-trained parser (Kitaev and Klein, 2018). Similar to the generation of the lexical constraints, we randomly sample sentences from the C4 dataset to synthesize syntactic constraint–text pairs.

c) Semantic Constraints To enable fine-grained control over the content and semantics of the output text, we also include three semantic constraints including a **topic/domain constraint**, a **sentiment constraint**, and a **political slant constraint**. Concretely, the topic constraint requires the model to produce texts related to a certain topic, for example, a text about a certain scientific field. Similarly, the sentiment constraint and the political slant constraint control the sentiment polarity and the political slant of the output texts, respectively. We first train a RoBERTa-based classifier (Liu et al., 2019) on labeled datasets for different semantic constraints with an additional NEUTRAL output class which is associated with random sentences from the C4 dataset. For instance, to enforce a sentiment constraint, we train a three-way classifier on sentences marked with POSITIVE, NEGATIVE, or NEUTRAL. The sentences labeled with POSITIVE and NEGATIVE are taken from the Amazon Review dataset (He and McAuley, 2016) and the sentences labeled with NEUTRAL class are drawn randomly from the C4 dataset and additionally filtered by a two-way classifier trained on the Amazon Review dataset to have no sentiment polarity. To simulate the population of sentiment-polarized sentences in web text, we construct the train set with 90% random sentences and 5% sentences of positive and negative sentiment each. We then apply the trained classifier on more randomly sampled sentences from C4 and use sentences that are labeled to be of positive or negative sentiment with high confidence to synthesize sentiment constraint–text pairs. Additionally, we use the Political Slant dataset (Voigt et al., 2018) for the political slant constraint and use the M2D2 dataset (Reid et al., 2022), which contains diverse topics in Wikipedia and arXiv categories, for topic control.

d) Style Constraints Apart from controlling keywords, syntax, and content, it is also desirable to control the style of the generated text. Here, we adopt the categories outlined in Jin et al. (2022) that include (*inter alia*) the formality, politeness, biasedness, toxicity, and simplicity of the output texts. Similar to sentiment constraints, we also train a style classifier with a NOSTYLE class with random sentences and collect stylized sentences from the C4 dataset to construct style constraint–text pairs. We use Grammarly's Yahoo Answers Formality Corpus (GYAFC) dataset (Rao and Tetreault, 2018) for formality constraints, the politeness transfer dataset collected by Madaan et al. (2020) for politeness control, the FlickrStyle stylized caption dataset (Gan et al., 2017; Li et al., 2018) for the control of style, the Wiki Neutrality Corpus (Pryzant et al., 2020) for biasedness control, the PWKP (Zhu et al., 2010) dataset for text simplification, and following the data collection method in Tran et al. (2020) for collecting offensive/non-offensive sentences from the C4 dataset.

e) Length Constraints Finally, we also consider a constraint that controls the length of the output text. The length constraint is simply the number of words of the output text after tokenization (measured using the Moses (Koehn et al., 2007) tokenizer). We consider an interval-based length constraint with an interval of 5, which requires the output text to contain between $5n$ and $5(n + 1)$ words for a given integer n. We construct length constraint–text pairs using randomly sampled sentences from C4 and the Moses tokenizer.

Using the process described above, we synthesize 1 million training constraints–text pairs for each category of constraint (lexical, syntactic, semantic, style, and length) and 50,000 pairs, each, for the development and test sets.

2.2. Constraint Verbalization

We verbalize the automatically derived constraints in different natural language formats. As shown in Table 1, for each constraint type, we design natural language templates to verbalize the constraints into natural language prompts. Verbalizing constraints has a few benefits: First, verbalizing the constraints in natural language effectively exploits the

LEXICAL	TARGET	The information was forwarded to Simpson's prosecutors, but it was ignored.
	CONSTRAINTS	Simpson ∧ (ignore ∨ overlook) ∧ information ∧ prosecutor ∨ ¬ message
	INSTRUCTION-1	Write something with the following keywords: Simpson, ignore or overlook, information, prosecutor, but do not include the following keywords: message.
	INSTRUCTION-2	Write something that contains the following words: Simpson, ignore or overlook, information, prosecutor but, does not contain the word message.
SYNTAX	TARGET	Avatar is a science fiction film directed by James Cameron.
	CONSTRAINTS	NOUN VERB DET NOUN NOUN NOUN VERB PREP NOUN NOUN PUNCT
	INSTRUCTION-1	Write something with a part-of-speech sequence NOUN VERB DET NOUN NOUN NOUN VERB PREP NOUN NOUN PUNCT.
	INSTRUCTION-2	Write something that results in the part-of-speech sequence NOUN VERB DET NOUN NOUN NOUN VERB PREP NOUN NOUN PUNCT after part-of-speech tagging with Spacy.
SEMANTIC	TARGET	Love this place. Never had a bad meal. Good portions and great people.
	CONSTRAINTS	the sentiment is positive
	INSTRUCTION-1	Write something with positive sentiment.
	INSTRUCTION-2	Write something positive.
STYLE	TARGET	I dunno, but gonna give a try.
	CONSTRAINTS	the style is informal
	INSTRUCTION-1	Write something that is informal.
	INSTRUCTION-2	Write something in an informal tone.
LENGTH	TARGET	I am going to finish the 3rd season of My Brilliant Friend tonight.
	CONSTRAINTS	length of 10-15 words
	INSTRUCTION-1	Write something that has 10 to 15 words.
	INSTRUCTION-2	Write something with 10 to 15 words

Table 1: Example texts, constraints, and the corresponding verbalized instructions. The instructions are highlighted in red. Note that there are multiple possible ways to verbalize instructions for each task.

prompt-based generation ability of the pre-trained model. Second, instruction-tuning with natural language verbalization of the constraints enables zero-shot constraint generalization, enabling the model to handle new constraints that are unseen during training by simply describing them in natural language. Moreover, we design multiple diverse natural language templates for each constraint used for training.

2.3. Composition of Constraints

Controlled text generation applications often require more than one constraint to be enforced at the same time during generation. To enable INSTRUCTCTG to handle such a composition of constraints, we verbalize the constraints individually and conjoin them with the conjunctive "and". For example, for the sentence "He bought a car" with both lexical and POS constraints, the final instruction will be "write something with keywords *buy*, *car* that has a part-of-speech tag sequence PRP VERB DET NOUN." We subsample the C4 dataset again to obtain another 1 million training examples with multiple constraints. For each sampled text, we first try to produce constraint verbalizations for as many constraint types as possible and then randomly sample a subset of the verbalizations with between 2 to 5 constraints uniformly at random.

2.4. Extension to Conditional Text Generation

The aforementioned data synthesis pipeline as described is applicable to unconditioned text generation. However, several major applications of text generation (e.g., paraphrase generation, summarization, etc.) deal with the generation of text conditioned on another piece of text. Therefore, we also extend our controlled generation pipeline to conditional text generation tasks.

In order to achieve this, instead of randomly sampling texts from a general unlabeled corpus like C4, we instead turn to datasets used for various conditional text generation tasks like paraphrase generation and summarization. In order to generate instructions for controllable conditional generation, we compose the original task-specific prompt with our natural language instructions for the set of control constraints. For instance, consider the following training instance for paraphrase generation: "How far is the earth from the sun?" $\mapsto$ "What is the distance between the sun and the earth?". In this case, the augmented instruction for a lexical constraint would be: "Write a paraphrase of 'how far is the earth from the sun with keyword *distance*' ". Our experiments consider paraphrase generation and question generation tasks and use the Quora Question Paraphrase dataset and the SQUAD question generation dataset (Rajpurkar et al., 2016; Du and

Cardie, 2018), respectively.

2.5. Instruction-based Meta In-context Learning

After the above data synthesis process, we use a combination of instruction tuning (Wei et al., 2022) and meta in-context learning (Min et al., 2022) where we prepend the natural language instructions for the constraints with 5 demonstrations, i.e., constraint–output pairs, of the same constraint or a composition of several constraints. To enable the model to do controlled text generation both with and without demonstrations, we omit the demonstrations from 50% of the examples during training. In those cases, we just perform instruction tuning. Given the collection of instructions (and demonstrations), we fine-tune the language model using maximum likelihood estimation and teacher forcing.

2.6. Decoding

At test time, unlike most previous work that requires complex decoding algorithms, INSTRUCTCTG can perform controlled text generation without any modification to the decoding process. Specifically, we can simply verbalize the target constraints (either using existing templates for seen constraints or writing a new description for unseen constraints) and optionally provide a few examples for the target constraint types. Then we can simply apply an out-of-box generation method, e.g., beam search.

3. Experiments

We now turn to the empirical portion of our paper.

3.1. Experimental Setup

Model We use T0-11B as our base model because it has previously been instruction-tuned on many NLP tasks. T0-11B is initialized from the 11B parameter version of T5+LM (Raffel et al., 2020), which is pre-trained on the C4 dataset with a text infilling objective and instruction-tuned on a collection of various NLP datasets.

Training Our model is trained on the mixture of training constraint types illustrated in Table 1. We hold out one semantic constraint (political slant) and two style constraints (humor and politeness) to evaluate INSTRUCTCTG's ability to generalize to unseen constraints. We fine-tune on all other constraints. Following Raffel et al. (2020), we assemble our multi-constraints training mixture by combining and shuffling all examples from all training constraints. We fine-tune our model with the Adam (Kingma and Ba, 2015) optimizer for 100000 steps with a learning rate of 1e-4, a batch size of 1024 text pairs, a dropout rate of 0.1, and a learning rate warmup of 8000 steps. Following Sanh et al. (2022), we perform checkpoint selection by choosing the checkpoint with the highest constraint satisfaction rate (see the next paragraph) on the validation splits of datasets of training constraints.

Evaluation We consider two evaluation metrics: constraint satisfaction rate and fluency. The constraint satisfaction rate is computed differently for each type of constraint. For the lexical and length constraints, we take it to be exact match, for the part-of-speech sequence constraint, we take it to be token-level accuracy, and, for the syntactic tree constraint, we take it to be labeled constituent F_1, and, finally, for the semantic and style constraints, we train a RoBERTa-based classifier to evaluate the controlled generation, following previous work on text style transfer (Xu et al., 2018). For our fluency metric, we use the perplexity of the generated text under OPT-30B, another language model trained on web text (Zhang et al., 2022).

Baselines We compare INSTRUCTCTG with both constrained search algorithms and score-based controlled generative models which are detailed below.

- **Dynamic Beam Allocation** (DBA; Post and Vilar, 2018) is a popular constrained search algorithm for lexically constrained text generation. It tracks constraint satisfaction status using a finite-state automaton.
- **NeuroLogic Decoding** (Lu et al., 2021) is one of the state-of-the-art constrained search algorithms. NeuroLogic decoding takes logic-based lexical constraints as input and uses prefix tries to track constraint satisfaction.
- **COLD** (Qin et al., 2022) is a scored-based constrained generation method that incorporates various kinds of constraints as energy functions and then performs constraint satisfaction through gradient-based sampling with Langevin dynamics in the logit space.
- **Diffusion-LM** (Li et al., 2022) is a diffusion model that differentiable constraints by considering them as a part of score functions.
- In **Constraint-specific fine-tuning (CFT)**, we fine-tune the model on the constraint–text pairs for each constraint type.

To ensure a fair comparison for CBS and NeuroLogic decoding, we fine-tune the pre-trained T0-11B model on the target sentences used for INSTRUCTCTG training with the unconditional text generation objective because they do not require constraint-specific training. For COLD, we use GPT-J as the backbone model because it requires a decoder-only language model for generation. For the Diffusion-LM, we initialize the transformer model with T0-11B parameters and train it

	LEXICAL		SYNTAX		SEMANTIC		STYLE		LENGTH		INFERENCE
	SUCC	FLUENCY↓	SUCC	FLUENCY↓	SUCC	FLUENCY↓	SUCC	FLUENCY↓	SUCC	FLUENCY↓	TIME
CFT	95.1	16.7	85.7	17.9	85.1	15.2	87.6	17.4	**100.0**	15.9	1.0 ×
DBA	96.8	23.4	-	-	-	-	-	-	-	-	1.3 ×
NeuroLogic	**97.9**	21.8	-	-	-	-	-	-	-	-	4.8 ×
COLD decoding	95.1	41.3	82.2	48.5	80.9	38.9	82.3	43.0	-	-	34.8 ×
Diffusion-LM	94.3	33.2	**88.6**	35.9	82.8	31.4	85.9	32.6	99.8	27.7	48.3 ×
INSTRUCTCTG	97.5	**12.6**	88.3	**14.4**	**88.2**	**12.2**	**90.6**	**13.1**	**100.0**	**11.9**	1.1 ×
w/o verbalization	95.5	16.3	85.0	17.9	85.5	14.7	88.4	15.9	**95.3**	14.1	1.1 ×
w/o demonstrations	96.8	13.3	87.8	15.3	87.8	13.0	90.1	13.6	**100.0**	12.2	1.0 ×
w/o multi-constraint training	97.2	12.9	87.0	14.9	86.8	13.1	89.2	14.2	**100.0**	12.4	1.1 ×
w/o multiple templates	97.3	12.8	88.1	14.6	84.7	12.8	90.2	13.9	**100.0**	**11.9**	1.1 ×

Table 2: Results on constraints seen during training. INSTRUCTCTG is faster and achieves better fluency (FLUENCY) compared to all baselines while maintaining competitive constraint satisfaction rate (SUCC) across all 5 constraint types.

on the same unlabeled data that is used for INSTRUCTCTG training. We also train a model with the same configuration as described in Li et al. (2022) with the same data. However, we found that it substantially underperforms with respect to our implementation. For the CFT baseline, we fine-tune T0-11B on the corresponding datasets for each constraint type.

3.2. Results on Seen Constraints

We first present the performance of INSTRUCTCTG on seen constraints in Table 2. We observe that INSTRUCTCTG achieves a comparable or better constraint satisfaction rate compared to many controlled text generation methods while generating much more fluent text.

Additionally, INSTRUCTCTG is over 4× faster than NeuroLogic decoding, a strong search-based algorithm, and over 40× faster than Diffusion-LM.

3.3. Results on Unseen Constraints

We also evaluate the performance of INSTRUCTCTG with a few demonstrations of constraints that are unseen during training. This is a unique feature of INSTRUCTCTG since existing search-based algorithms are only capable of lexical constraints and score-based methods require score functions trained on the train set of the target constraint. We present the results in Table 3, where we combine the few-shot performance of INSTRUCTCTG with the supervised results of existing methods. We find that despite only a few demonstrations and no update to the model, INSTRUCTCTG achieves a higher constraint satisfaction rate and better output quality compared to the baselines. Moreover, INSTRUCTCTG also performs competitively with CFT, an oracle baseline that fine-tunes the model on the unseen constraints. This shows that INSTRUCTCTG has strong few-shot constraint generalization ability, allowing the model to handle new constraints by only writing a description and a few examples of them.

We also conduct an analysis of the in-domain and the out-of-domain generalization ability of INSTRUCTCTG. For

	SEMANTIC		STYLE	
	SUCC	FLUENCY	SUCC	FLUENCY
CFT (Oracle)	84.7	15.8	88.1	16.6
COLD decoding	80.5	41.2	81.3	42.1
Diffusion-LM	82.1	33.9	85.3	30.3
INSTRUCTCTG	**83.4**	**15.3**	**87.3**	**14.7**
w/o verbalization	67.7	19.8	65.1	18.5
w/o demonstrations	79.8	16.5	83.1	15.7
w/o multiple templates	82.7	16.1	86.5	15.1

Table 3: Results on unseen constraints. We observe that INSTRUCTCTG achieves better performance compared to the score-based baselines trained on the unseen constraint datasets. Note that, in this setting, CFT is a skyline as it was *trained* constraints unseen by the other methods.

in-domain tests, we compare INSTRUCTCTG and the CFT baseline on texts with lengths greater than 60 words; both were trained only on texts whose lengths were less than 60 words. We find that CFT's accuracy drops from 100% to 91% when increasing the length bin to 100-105 words while INSTRUCTCTG 's accuracy only drops to 97%.

For an out-of-domain test, we consider a variant of INSTRUCTCTG that is trained on a mixture of lexical, syntactic, content, and style constraints, but not on the length constraint. We also consider the original T0-11B model. We find that INSTRUCTCTG results in 82% of the constraints satisfied without demonstrations, and 88% of the constraints satisfied with 5 demonstrations. In contrast, T0-11B only results in 65% of the constraints satisfied. This confirms that INSTRUCTCTG helps generalize on out-domain unseen constraints above baselines. These results highlight the importance of demonstrations, which is in line with findings in Table 3.

3.4. Analysis

Ablation Study We first ablate different design choices of INSTRUCTCTG to investigate their relative importance.

The results are shown in Table 2 and 3. First, we find that training the model without verbalizing constraints into natural language instructions leads to a substantial performance drop, especially on unseen constraints. Using multiple templates for verbalization also (marginally) improves performance. We also find that meta in-context learning helps the model generalize to unseen tasks. We hypothesize that this is because examples during training help the model learn content-invariant relationships between examples, constraints, and outputs. One can also observe that training with multiple constraints at the same time leads to improved performance. In addition, in preliminary experiments, we consider different numbers (1, 2, 5, and 10) of demonstrations and find that 5 demonstrations results in a reasonable performance–efficiency trade-off.

Results on Conditional Generation Tasks We also test INSTRUCTCTG on conditional natural language generation tasks. To do so, we further fine-tune our model on two conditional text generation tasks (i.e., the QQP paraphrase generation dataset and the SQUAD question generation dataset) with lexical constraint using INSTRUCTCTG following the procedure described in section 2.4. We consider disjunctive constraints only and randomly synthesize 3 to 5 keyword constraints for each output sentence. The results are shown in Table 4. We can see that INSTRUCTCTG achieves a comparable constraint satisfaction rate compared to previously proposed controlled generation methods while leading to substantial improvement in task-specific metrics. This confirms the effectiveness of INSTRUCTCTG on conditional text generation tasks.

Results on Composition of Constraints We also analyze the performance of INSTRUCTCTG under composition of constraints. The results are shown in Figure 2. In this experiment, we calculate the percentage of SUCC and FLUENCY that the model retains compared to SUCC and FLUENCY without constraint composition for each constraint type in the entire test set. We then average this percentage across all constraint types to obtain a final constraint satisfaction performance number. The results shown in Figure 2 demonstrate the ability of INSTRUCTCTG to effectively handle multiple constraints at the same time.

	PHARAPHRASE		QG	
	SUCC	ROUGE	SUCC	METEOR
FT	-	51.7	-	29.4
+ NeuroLogic decoding	**96.5**	48.5	**95.8**	28.6
Diffusion-LM	93.5	45.3	93.1	26.6
CFT	94.7	55.5	94.5	33.9
INSTRUCTCTG	96.1	**57.4**	**95.8**	**35.4**

Table 4: Results on conditional natural language generation tasks. Compared with the baselines, INSTRUCTCTG achieves better task-specific performance while having comparable success rate in following constraints (SUCC).

4. Related Work

We now turn to discuss related work.

4.1. Neural text generation

Neural text generation generally involves two stages: training and decoding. During the training stage, one estimates a probability model over natural language text. The probability model is typically parameterized autoregressively, i.e., it is factored into the product of per-token conditional probabilities in left-to-right order. The per-token conditional probabilities are parameterized with a shared neural network such as an LSTM (Hochreiter and Schmidhuber, 1997) or a Transformer (Vaswani et al., 2017). The parameters of the neural network are typically estimated by regularized maximum likelihood estimation. During the decoding stage, one aims to generate text using the trained model. Popular decoding algorithms include beam search, top-k sampling (Fan et al., 2018), nucleus sampling (Holtzman et al., 2020), and locally typical decoding (Meister et al., 2023).

4.2. Controlled text generation

Conventional decoding algorithms cannot incorporate constraints during generation. Therefore, they can not ensure the output satisfies certain requirements that are important to certain applications. To this end, there exist two distinct lines of research focusing on controlled text generation which focus on constrained search algorithms and score-based sampling methods respectively.

Constrained search algorithms enforce strict lexical constraints on the outputs by modifying the search space according to the constraints. Constrained beam search (CBS) algorithm (Anderson et al., 2017), first proposed to track constraint satisfaction using a finite-state automaton. However, CBS requires the model to maintain a finite-state machine with 2^C states (where C is the number of constraints), which results in significantly increased time complexity. To reduce the runtime overhead of CBS, Hokamp and Liu (2017) and Post and Vilar (2018) introduced grid beam search (GBS) and dynamic beam allocation (DBA), respectively. Lu et al. (2021) introduced NeuroLogic decoding, a constrained search algorithm that aims to satisfy rich logic-based lexical constraints expressed in conjunctive normal form and extend it to NeuroLogic A* decoding (Lu et al., 2022) by incorporating lookahead heuristics. While existing constrained search algorithms significantly improve the constraint satisfaction rate, they still lead to slower generation speed and lower quality text because they prune the output distribution space aggressively and, thus, tend to collapse into globally suboptimal results. Moreover, most

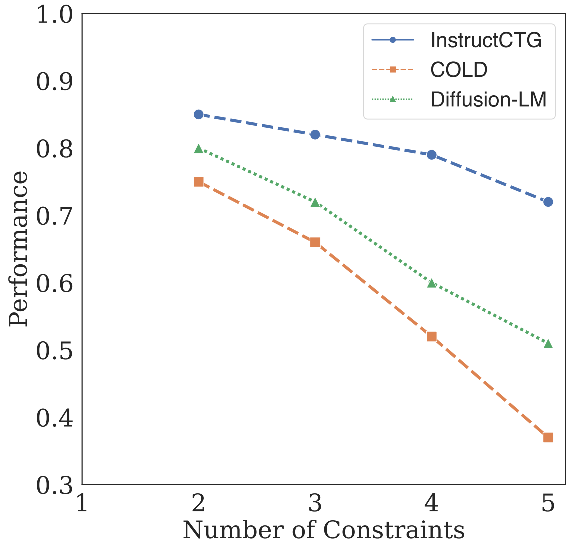

Figure 2: Results on the composition of multiple constraints. The aggregate performance number on the y-axis is measured by the following procedure: First, for each constraint type, we calculate the percentage of SUCC and FLUENCY that the model retains compared to the performance without constraint composition. We then average the percentages across all constraint types as the performance number.

constrained search algorithms cannot incorporate constraint types other than lexical constraints.

Score-based sampling methods attempt to incorporate constraints by converting constraints into differentiable score functions. Specifically, score functions for soft constraints such as sentiment control can be implemented as the cross entropy loss of corresponding classifiers. Hard constraints such as lexical constraints can be modeled by a differentiable n-gram matching function (Liu et al., 2022). Compared to constrained search algorithms, score-based sampling methods are more flexible because they can deal with more diverse constraint types and the composition of different constraints. However, score-based sampling methods do not come with constraint satisfaction guarantees and often lead to worse generation quality because the output distribution is modified (Qin et al., 2022). Moreover, score-based sampling methods lead to much slower generation speeds because multiple score-matching steps need to be performed. They also require carefully tuned relative weights between the scores of different constraints and the task-specific loss to achieve a good trade-off between the quality and the constraint satisfaction rate of output texts.

Apart from search-based and score-based methods, Dinu et al. (2019) proposed a similar specialized training method that appends lexical constraints to the input when training the model. Another related work is the Similarly, CTRL (Keskar et al., 2019) is pre-trained with structures that naturally co-occur with raw texts and can deal with certain kinds of constraints such as style and domain constraints.

5. Conclusion

In this work, we introduce INSTRUCTCTG, a simple framework for controlled text generation which exploits the instruction following ability of large language models as an alternative to modifying the decoding procedure of the model. INSTRUCTCTG is faster and achieves better generation quality compared to state-of-the-art controlled generation methods while maintaining similar quality. INSTRUCTCTG also shows a strong few-shot generalization ability to unseen constraints, the ability to model the composition of multiple constraints, and can be seamlessly extended to conditional natural language generation tasks.

Acknowledgements

Ryan Cotterell acknowledges support from the Swiss National Science Foundation (SNSF) as part of the "The Forgotten Role of Inductive Bias in Interpretability" project. EGW was supported by an ETH Postdoctoral Fellowship.

References

Afra Amini, Li Du, and Ryan Cotterell. 2023. Structured Voronoi sampling. *arXiv preprint arXiv:2306.03061*.

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2017. Guided open vocabulary image captioning with constrained beam search. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 936–945, Copenhagen, Denmark. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901.

Georgiana Dinu, Prashant Mathur, Marcello Federico, and Yaser Al-Onaizan. 2019. Training neural machine translation to apply terminology constraints. In *Proceedings of the 57th Annual Meeting of the Association for Com-*

putational Linguistics, pages 3063–3068, Florence, Italy. Association for Computational Linguistics.

Xinya Du and Claire Cardie. 2018. Harvesting paragraph-level question-answer pairs from Wikipedia. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1907–1917, Melbourne, Australia. Association for Computational Linguistics.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia. Association for Computational Linguistics.

Chuang Gan, Zhe Gan, Xiaodong He, Jianfeng Gao, and Li Deng. 2017. StyleNet: Generating attractive visual captions with styles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3137–3146.

Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th International Conference on World Wide Web*, pages 507–517.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

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.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Di Jin, Zhijing Jin, Zhiting Hu, Olga Vechtomova, and Rada Mihalcea. 2022. Deep learning for text style transfer: A survey. *Computational Linguistics*, 48(1):155–205.

Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. 2019. CTRL: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2676–2686, Melbourne, Australia. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.

Sachin Kumar, Biswajit Paria, and Yulia Tsvetkov. 2022. Gradient-based constrained sampling from language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2251–2277, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Juncen Li, Robin Jia, He He, and Percy Liang. 2018. Delete, retrieve, generate: a simple approach to sentiment and style transfer. 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 1865–1874, New Orleans, Louisiana. Association for Computational Linguistics.

Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori Hashimoto. 2022. Diffusion-LM improves controllable text generation. In *Advances in Neural Information Processing Systems*.

Bill Yuchen Lin, Wangchunshu Zhou, Ming Shen, Pei Zhou, Chandra Bhagavatula, Yejin Choi, and Xiang Ren. 2020. CommonGen: A constrained text generation challenge for generative commonsense reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1823–1840, Online. Association for Computational Linguistics.

Guangyi Liu, Zichao Yang, Tianhua Tao, Xiaodan Liang, Junwei Bao, Zhen Li, Xiaodong He, Shuguang Cui, and Zhiting Hu. 2022. Don't take it literally: An edit-invariant sequence loss for text generation. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2055–2078, Seattle, United States. 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. *arXiv preprint arXiv:1907.11692*.

Ximing Lu, Sean Welleck, Peter West, Liwei Jiang, Jungo Kasai, Daniel Khashabi, Ronan Le Bras, Lianhui Qin, Youngjae Yu, Rowan Zellers, Noah A. Smith, and Yejin Choi. 2022. NeuroLogic A*esque decoding: Constrained text generation with lookahead heuristics. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 780–799, Seattle, United States. Association for Computational Linguistics.

Ximing Lu, Peter West, Rowan Zellers, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. NeuroLogic decoding: (Un)supervised neural text generation with predicate logic constraints. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4288–4299, Online. Association for Computational Linguistics.

Aman Madaan, Amrith Setlur, Tanmay Parekh, Barnabas Poczos, Graham Neubig, Yiming Yang, Ruslan Salakhutdinov, Alan W Black, and Shrimai Prabhumoye. 2020. 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.

Clara Meister, Tiago Pimentel, Gian Wiher, and Ryan Cotterell. 2023. Locally Typical Sampling. *Transactions of the Association for Computational Linguistics*, 11:102–121.

Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2022. MetaICL: Learning to learn in context. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2791–2809, Seattle, United States. Association for Computational Linguistics.

Matt Post and David Vilar. 2018. Fast lexically constrained decoding with dynamic beam allocation for neural machine translation. 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 1314–1324, New Orleans, Louisiana. Association for Computational Linguistics.

Reid Pryzant, Richard Diehl Martinez, Nathan Dass, Sadao Kurohashi, Dan Jurafsky, and Diyi Yang. 2020. Automatically neutralizing subjective bias in text. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 480–489.

Lianhui Qin, Sean Welleck, Daniel Khashabi, and Yejin Choi. 2022. COLD decoding: Energy-based constrained text generation with Langevin dynamics. In *Advances in Neural Information Processing Systems*.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

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.

Sudha Rao and Joel Tetreault. 2018. Dear sir or madam, may I introduce the GYAFC dataset: Corpus, benchmarks and metrics for formality style transfer. 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 129–140, New Orleans, Louisiana. Association for Computational Linguistics.

Machel Reid, Victor Zhong, Suchin Gururangan, and Luke Zettlemoyer. 2022. M2D2: A massively multi-domain language modeling dataset. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 964–975, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M Rush. 2022.

Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*.

Minh Tran, Yipeng Zhang, and Mohammad Soleymani. 2020. Towards a friendly online community: An unsupervised style transfer framework for profanity redaction. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2107–2114, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30.

Rob Voigt, David Jurgens, Vinodkumar Prabhakaran, Dan Jurafsky, and Yulia Tsvetkov. 2018. RtGender: A corpus for studying differential responses to gender. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*.

Jingjing Xu, Xu Sun, Qi Zeng, Xiaodong Zhang, Xuancheng Ren, Houfeng Wang, and Wenjie Li. 2018. Unpaired sentiment-to-sentiment translation: A cycled reinforcement learning approach. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 979–988, Melbourne, Australia. Association for Computational Linguistics.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. OPT: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Wangchunshu Zhou, Tao Ge, Ke Xu, Furu Wei, and Ming Zhou. 2019. BERT-based lexical substitution. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3368–3373, Florence, Italy. Association for Computational Linguistics.

Wangchunshu Zhou, Dong-Ho Lee, Ravi Kiran Selvam, Seyeon Lee, and Xiang Ren. 2021. Pre-training text-to-text transformers for concept-centric common sense. In *International Conference on Learning Representations*.

Zhemin Zhu, Delphine Bernhard, and Iryna Gurevych. 2010. A monolingual tree-based translation model for sentence simplification. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1353–1361, Beijing, China.

A. Appendix

LEXICAL	CONSTRAINTS	I $\wedge$ time $\wedge$ television $\wedge$ game
	NEUROLOGIC	I watch television and play game for a time.
	DIFFUSION-LM	I play game and television at this time.
	INSTRUCTCTG	I enjoy watching television and playing video games in my free time.
SYNTAX	CONSTRAINTS	VERB DET ADJ NOUN PREP NOUN NOUN NOUN
	COLD	Look this white bear in the forest stone
	DIFFUSION-LM	Bring the new computer to storage room door
	INSTRUCTCTG	Read this new article about Aircraft System design.
SEMANTIC	CONSTRAINTS	the sentiment being negative
	COLD	This restaurant is not good. It would just soon be lacking of quality for the meat and sausage.
	DIFFUSION-LM	The service is very disappointing and the quality of the food like meat and wine is also not of good level.
	INSTRUCTCTG	This article is really boring. It is completely non sense to me and my collegues.
STYLE	CONSTRAINTS	the style being polite
	COLD	You are honored to come there for the meeting
	DIFFUSION-LM	I would be pleasure to receive my money back sincerely
	DIFFUSION-LM	Could you please help me check the status of my order

Table 5: Samples generated with INSTRUCTCTG and compared models for different control tasks.

A.1. Limitations and Potential Social Impacts

One limitation is that we only consider a few types of constraints in our experiments and only test INSTRUCTCTG with T0 as the backbone model. It would be helpful to test INSTRUCTCTG on more diverse constraint types and backbone models of different sizes to get an even better picture of its utility. As for potential negative social impacts, INSTRUCTCTG could be used by malicious users by writing instructions that encourage the model to generate toxic or biased texts. This is a common potential risk of controlled text generation methods.

A.2. Implementation Details

Training Data As described in Section 2.1 and 2.3, we train the fine-tune on 1 million examples containing a single constraint and 1 million examples containing multiple constraints (ranging from 2 to 5). When fine-tuning for conditional natural language generation tasks, we use all the training examples in the original datasets and synthesize training data with the lexical constraints using the method described in Section 2.1. We fine-tune the model with a learning rate of 5e-5 with other hyperparameters unchanged compared to that described in Section 3.1.

Baseline Training We train the CFT baseline with the exact hyperparameters used during the training of INSTRUCTCTG, which is empirically found to work well for both conditional fine-tuning and our approach. For other baselines, we start from the hyperparameters provided in the corresponding papers and do a grid search over other possible values on the development set.

Constraint Composition Evaluation In Section 3.4, we compose constraints from different categories to avoid potential conflicts. Specifically, we first select 1000 test examples that have at least one valid constraint in each category. We then test the model by randomly sampling and composing 2,3,4,5 constraints.

A.3. Examples of Generated Text

We also present some samples of INSTRUCTCTG and compared baselines in Table 5 for qualitative analysis.