NEUROLOGIC DECODING:
(Un)supervised Neural Text Generation with Predicate Logic Constraints

Ximing Lu[†‡]   **Peter West**[†‡]   **Rowan Zellers**[†‡]
Ronan Le Bras[‡]   **Chandra Bhagavatula**[‡]   **Yejin Choi**[†‡]

[†]Paul G. Allen School of Computer Science & Engineering, University of Washington
[‡]Allen Institute for Artificial Intelligence
`{lux32, pawest, rowanz, yejin}@cs.washington.edu`
`{ronanlb, chandrab}@allenai.org`

Abstract

Conditional text generation often requires lexical constraints, i.e., which words should or shouldn't be included in the output text. While the dominant recipe for conditional text generation has been large-scale pretrained language models that are finetuned on the task-specific training data, such models do not learn to follow the underlying constraints reliably, even when supervised with large amounts of task-specific examples.

We propose NEUROLOGIC DECODING, a simple yet effective algorithm that enables neural language models – supervised or not – to generate fluent text while satisfying complex lexical constraints. Our approach is powerful yet efficient. It handles any set of lexical constraints that is expressible under predicate logic, while its asymptotic runtime is equivalent to conventional beam search.

Empirical results on four benchmarks show that NEUROLOGIC DECODING outperforms previous approaches, including algorithms that handle a subset of our constraints. Moreover, we find that unsupervised models with NEUROLOGIC DECODING often outperform supervised models with conventional decoding, even when the latter is based on considerably larger networks. Our results suggest the limit of large-scale neural networks for fine-grained controllable generation and the promise of inference-time algorithms.

1 Introduction

Text generation applications often need to incorporate semantic constraints, i.e., what words *should* and *shouldn't* appear in the output generation. Consider the task of generating a recipe from a set of ingredients (Kiddon et al., 2016), such as 'garlic,' 'steak', and 'soy sauce' (Figure 1). A generated recipe should cover all of those ingredients, without hallucinating new ones (such as 'pork' or 'beans'). This restriction, like others in Figure 1 for other
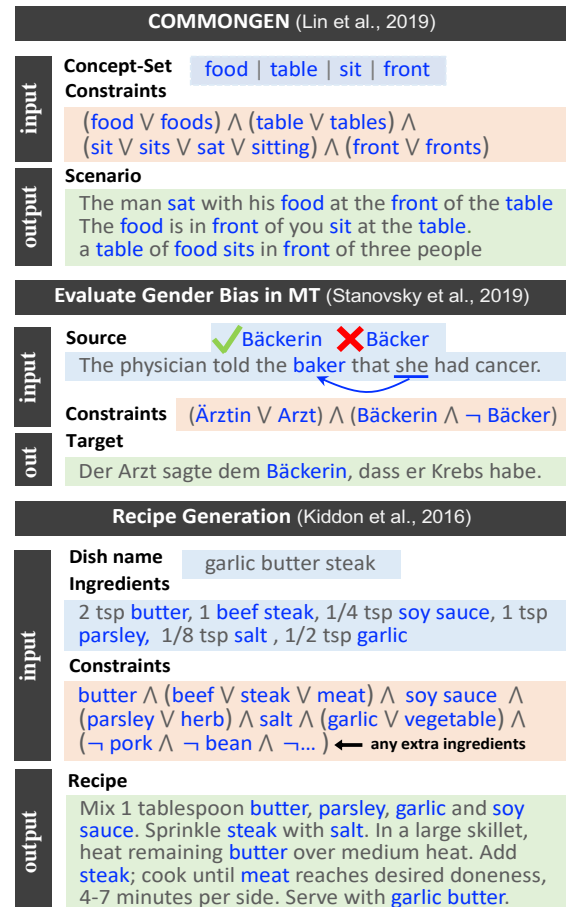

Figure 1: Overview of several constrained generation tasks. For instance, generating a short description from a set of concepts (COMMONGEN; Lin et al., 2020) requires using each of those words at least once; this can be expressed as a logical expression (here, '(food ∨ foods) ∧ . . .'). Our proposed NEUROLOGIC DECODING handles all predicate logic constraints efficiently, yet with the same asymptotic runtime as beam search.

applications, can be modeled by a set of lexical constraints expressed as a predicate logic formula.

The dominant paradigm today for performing such *constrained generation* is to start with a pretrained language model, and then finetune it on a dataset of task-specific examples. However, pretrained language models struggle at learning to

follow these constraints, even when the finetuning dataset is large. For example, for the aforementioned recipe generation task, a GPT2 model finetuned on hundreds of thousands of recipes still hallucinates extra ingredients. In stark contrast, humans need to see only a few examples (or even none) to generate the desired output satisfying all the logical constraints, e.g., writing a recipe that mentions each ingredient (butter, steak, etc.) without using new ones.

We hypothesize that this mismatch is due to a fundamental under-specification of finetuning. If we finetune one of today's state-of-the-art language models on a dataset, the likelihood of it generating sequences from the same distribution should increase. Yet there is no guarantee that this improvement in likelihood will come from improvements on the fundamental task of constrained generation, as opposed to picking up on dataset-specific patterns such as language style. In fact, we present analysis suggesting that 'worst-case' learning behavior is common in practice: when we increase the finetuning data fed to GPT2 by an order of magnitude, constraint-satisfaction with standard beam search shows only modest improvement.

To address this issue, we propose NEUROLOGIC DECODING, which effectively enforces the satisfaction of given lexical constraints by controlling the decoding stage of sequence generation. These constraints can be any predicate logic formula, which crucially includes both positive constraints (the word 'butter' must be generated somewhere) and negative constraints ('bean' cannot be generated). These simpler constraints can then be combined through logical connectives to handle more complex requirements such as inflection or synonyms ('beef' *or* 'steak' both satisfy the constraint of referring to the steak). While beam search aims to maximize the likelihood of the generated sequence, our method searches for optimal output sequences among the strings that also satisfy the given constraints. It does so efficiently: we convert the hard logic constraints into a soft penalty term in the decoding objective, and use a beam-based search to find approximately-optimal solutions; constraint states are tracked to reuse computation. NEUROLOGIC DECODING thus effectively and efficiently controls text generation without requiring any modification of the model structure or training pipeline.

We evaluate our method on four different text generation tasks: generative commonsense reasoning (COMMONGEN; Lin et al., 2020), recipe generation (Kiddon et al., 2016), data-grounded dialogue response generation (Wen et al., 2015), and reducing gender bias in machine translation (Stanovsky et al., 2019). Empirical results demonstrate that NEUROLOGIC DECODING ensures the satisfaction of given constraints while maintaining high generation quality, in turn leading to new SOTA results in both the supervised and zero-shot setting.

2 Method

In this section, we first rigorously define predicate logic constraint, and then present in detail the NEUROLOGIC DECODING algorithm.

2.1 Predicate Logic Constraint

Let us define a predicate $D(\mathbf{a}, \mathbf{y})$ to be a boolean function indicating the occurrence of key phrase $\mathbf{a}$ in a sequence $\mathbf{y}$, where $\mathbf{a}$ can be either unigram or multi-gram. $D(\mathbf{a}, \mathbf{y})$ will be true iff $\mathbf{a}$ occurs in $\mathbf{y}$.

$$D(\mathbf{a}, \mathbf{y}) \equiv \exists\, i,\ \mathbf{y}_{i:i+|\mathbf{a}|} = \mathbf{a}$$

NEUROLOGIC accepts lexical constraints in Conjunctive Normal Form (CNF):

$$\underbrace{\left(D_1 \vee D_2 \cdots \vee D_i\right)}_{C_1} \wedge \cdots \wedge \underbrace{\left(D_k \vee D_{k+1} \cdots \vee D_n\right)}_{C_m}$$

where each D_i represents a single positive or negative constraint, $D(\mathbf{a_i}, \mathbf{y})$ or $\neg D(\mathbf{a_i}, \mathbf{y})$, restricting whether key phrase $\mathbf{a_i}$ should be strictly included or omitted in $\mathbf{y}$, respectively. Any propositional logical formula can be converted to CNF, and thus handled by NEUROLOGIC. Notationally, we will refer to each individual constraint D_i as a *literal*, and the disjunction of literals as a *clause*, denoted as C_j, with L being the total number of clauses. Our method seeks optimal sequences in which all clauses are satisfied:

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y} \in \mathcal{Y}} P_\theta(\mathbf{y}|\mathbf{x}) \quad \text{where} \quad \sum_{i=1}^{L} C_i = L \quad (1)$$

Past work on constrained optimization introduces penalties (Fiacco, 1976) to approximate the constrained optimization problem with an unconstrained problem. Specifically, by adding a high-cost penalty term for violated constraints:

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y} \in \mathcal{Y}} P_\theta(\mathbf{y}|\mathbf{x}) - \lambda' \sum_{i=1}^{L} (1 - C_i) \quad (2)$$
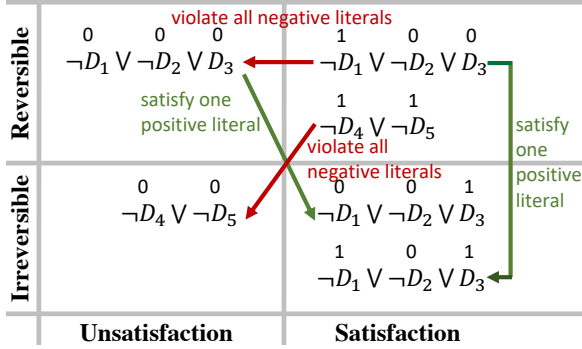
Figure 2: Clause states and possible transitions. D_i and $\neg D_i$ denote positive and negative literal respectively.

Intuitively, this objective balances sequence likelihood (term 1) and constraint satisfaction (term 2). The aim is to find sequences that do well at both dimensions. While exhaustive search is intractable, we use a beam-based search to find approximately-optimal solutions for this objective.

2.2 Constraint States

When considering whether a generation hypothesis satisfies some clause C_i during generation, there are fundamentally 4 possible states (as in figure 2)

S1 **reversible unsatisfaction**: If an unsatisfied clause C_i contains at least one positive literal, C_i could be satisfied in the future by fulfilling one of its positive literal(s).

S2 **irreversible unsatisfaction**: If an unsatisfied clause C_i contains negative literal(s) only, C_i will maintain unsatisfied in the future since the violation of negative literals could not be overturned.

S3 **reversible satisfaction**: If all satisfied literal(s) in a satisfied clause C_i are negative literal(s), C_i could switch back to unsatisfied in the future by violating all of its satisfied negative literal(s).

S4 **irreversible satisfaction**: If satisfied literal(s) in a satisfied clause C_i contains at least one positive literal, C_i will maintain satisfied in the future since the fulfilment of positive literals is irreversible.

To track the states of literals and clauses efficiently, we maintain two prefix tries. The first trie, $\mathcal{T}^+$, tracks *unsatisfied positive* literals from all clauses in states S1 and S3, while the other trie, $\mathcal{T}^-$, tracks *satisfied negative* literals from all clauses in state S3. We do not track anything from clauses in state S2 or S4, as those are already irreversible.

If a positive literal is satisfied, its clause in state S1 or S3 is henceforth irreversibly satisfied (state S4), thus we remove all literals of that clause from both tries and stop tracking. If a negative literal in state S3 is violated, we remote it from the trie $\mathcal{T}^-$. Once all negative literals of a clause in state S3 has been removed, the clause switches back to unsatisfied (state S1 or S2). If it has unsatisfied positive literal(s) in the trie $\mathcal{T}^+$, it becomes reversibly unsatisfied (state S1); otherwise it shall stay irreversibly unsatisfied (state S2).

2.3 Algorithm

Since exhaustive search to optimize the CNF constraints is intractable, NEUROLOGIC uses a beam-based search to approximate. The high-level intuition is that at each time step, NEUROLOGIC selects generation hypotheses in consideration of both the objective function and the diversity of the partially satisfied constraints. We achieve such by 3 steps: *pruning*, *grouping*, and *selecting* (illustrated in figure 3, and detailed below).

At each time step, the decoding model generates a distribution over all vocabulary V for k hypotheses in the current beam, resulting in a candidate score matrix of size $k \times |V|$. Along with generating score matrix, we produce a constraint state for each of the $k \times |V|$ new candidates h, based on the next token considered.

Pruning step: We first discard any h with irreversible unsatisfied clause (state S2) to focus only on candidates that might satisfy all constraints. Then, we filter candidates h to those in the top-tier of both satisfied constraints and sequence likelihood. Specifically, we drop any candidates not in the top-α in terms of likelihood $P_\theta(\mathbf{y}_t | \mathbf{y}_{<t})$, and not in the top-$\beta$ in terms of number of satisfied clauses $\sum_{i=1}^{L} C_i$. These are adjustable parameters, corresponding to maximum tolerance to sequence fluency and constraint satisfaction.

Grouping step: Next, we select the beam from the pruned candidates. Naively selecting k best candidates with respect to the objective function would not work well, since such greedy selection would bias toward sequences with high likelihood and easy-to-satisfy clauses at early timestep, which can lead to struggling with remaining hard-to-satisfy clauses later on. Therefore, the key intuition is to consider diverse partial solutions early on with respect to the set of irreversibly satisfied clauses, i.e., $\{C_i \mid C_i \in \text{state S4}\}$. We group candidates based on this set and select (in the next step) the best ones

Constraints $D_1(\text{cowboy}) \wedge D_2(\text{dog}) \wedge (D_3(\text{play music}) \vee D_4(\text{plays music})) \wedge (D_5(\text{catch}) \vee D_6(\text{catches}))$

Underbraces: C_1, C_2, C_3, C_4

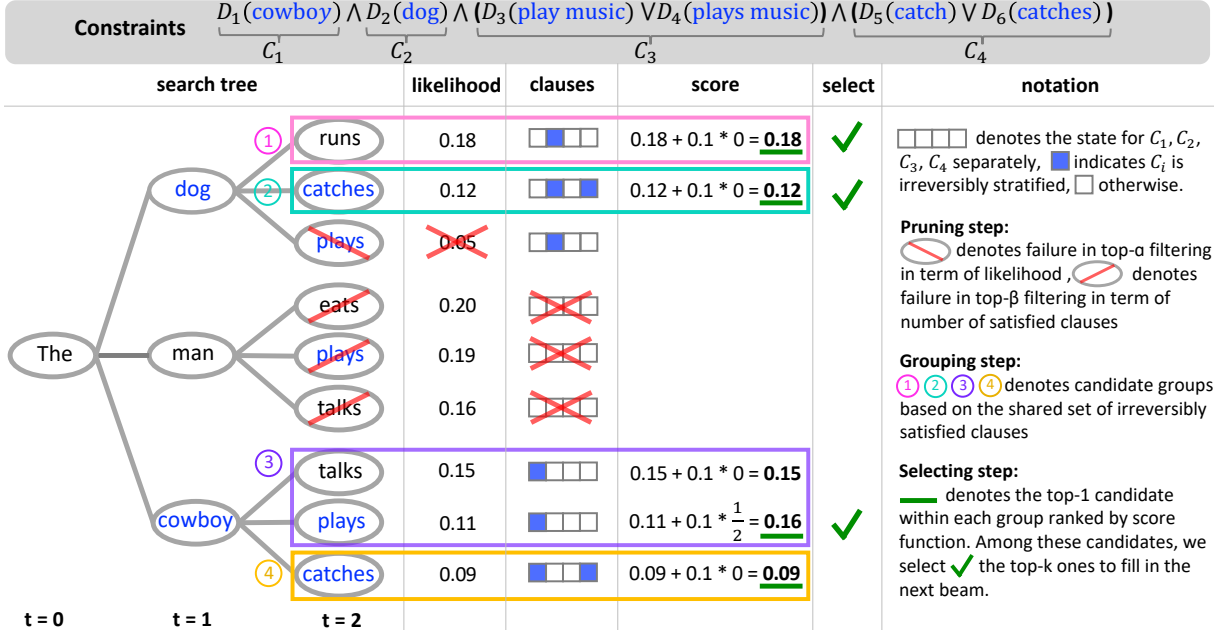| search tree	likelihood	clauses	score	select	notation
(1) runs	0.18		0.18 + 0.1 * 0 = **0.18**	✓	▢▢▢▢ denotes the state for C_1, C_2, C_3, C_4 separately, ▨ indicates C_i is irreversibly stratified, ▢ otherwise.
dog (2) catches	0.12		0.12 + 0.1 * 0 = **0.12**	✓	
plays	~~0.05~~				**Pruning step:** ⊘ denotes failure in top-α filtering in term of likelihood, ⬭⃥ denotes failure in top-β filtering in term of number of satisfied clauses
eats	0.20				
man plays	0.19				**Grouping step:** (1)(2)(3)(4) denotes candidate groups based on the shared set of irreversibly satisfied clauses
talks	0.16				
(3) talks	0.15		0.15 + 0.1 * 0 = **0.15**		**Selecting step:** ▬ denotes the top-1 candidate within each group ranked by score function. Among these candidates, we select ✓ the top-k ones to fill in the next beam.
cowboy plays	0.11		$0.11 + 0.1 * \frac{1}{2} = $ **0.16**	✓	
(4) catches	0.09		0.09 + 0.1 * 0 = **0.09**		

t = 0   t = 1   t = 2

Figure 3: Illustration of the NEUROLOGIC decoding procedure. In this example, $k = 3$, $\alpha = 8$, $\beta = 2$, $\lambda = 0.1$

from each group to fill the beam.

Selecting step: To select best ones from each group, we first rank candidates within a group by score function:

$$\mathbf{s} = P_\theta(\mathbf{y}_t|\mathbf{y}_{<t}) + \lambda \cdot \max_{\substack{D(\mathbf{a_i},\mathbf{y}) \\ \in \text{ state S1}}} \frac{|\hat{\mathbf{a}}_\mathbf{i}|}{|\mathbf{a_i}|} \qquad (3)$$

where $\hat{\mathbf{a}}_\mathbf{i}$ is $\mathbf{a_i}$'s matched prefix with ongoing generation. For example, for $\mathbf{y}$ = *"The boy climbs an apple"* and constraint $\mathbf{a_i}$=*"apple tree"*, we have $\hat{\mathbf{a}}_\mathbf{i}$=*"apple"*. The second term denotes maximal percentage of matched prefix in partially satisfied positive literals. Intuitively, this score function ranks candidates by likelihood and gives a partial reward to candidates moving towards satisfying a positive literal in an unsatisfied clause (state S1). λ is an adjustable parameter, controlling how much we favor candidates towards fulfilling another unsatisfied clause. We then proceed in rounds of filling the beam, visiting each group and taking the best scoring ones in rotation, until we reach k candidates. The group traversing order follows the descending order of the highest score in each group. In the end, we take the hypothesis with highest likelihood from the ones with maximal satisfied clauses.

3 Related Work

NEUROLOGIC distinguishes itself from past works in constrained decoding in 3 fundamental ways.

- First, NEUROLOGIC generalizes to arbitrary logical constraints by handling the full scope of CNF constraint, while previous works only allow a subset of this (typically conjunctions).
- Second, NEUROLOGIC effectively optimizes objective function through efficient and diverse search over output space, while previous works suffer from either myopic and narrow or inefficient exploration of the search space.
- Third, the asymptotic runtime of NEURO-LOGIC is $O(Nk)$[1], same with beam search, constant with respect to number of constraints $\mathcal{C}$. Some previous works suffer from exponential runtime, making applications infeasible.

A detailed comparison between NEUROLOGIC and previous methods is provided in table 1.

3.1 Previous Constrained Decoding Approach

Anderson et al. (2017) propose constrained beam search (**CBS**), where constraint satisfaction is tracked by a finite-state machine with $2^\mathcal{C}$ states (all possible satisfaction status for $\mathcal{C}$ constraints). Beam search is done over all states with k candidates per state. This method has an exponential complexity $O(Nk2^\mathcal{C})$, making many applications infeasible.

Hokamp and Liu (2017) propose grid beam search (**GBS**), which groups together hypotheses by number of constraints satisfied, giving $\mathcal{C} + 1$

---

[1]N denotes sequence length and k denotes beam size. In this paper, we the asymptotic runtimes is in terms of the number of calls to a deep generator that scores $P_\theta(\mathbf{y}_t|\mathbf{y}_{<t})$; this is because calling the generator is the most expensive part of decoding (as opposed to auxiliary bookkeeping).

Feature	Example		CBS	GBS	Post and Vilar	Hu et al.	CGMH	Sha	NEUROLOGIC
AND	$oil \wedge pork$ Include oil and pork		✓	✓	✓	✓	✓	✓	✓
Positive Set AND	$oil \wedge (pork \vee beef)$ Include oil and a protein		✓						✓
Any Predicate Logic Formula	$\neg oil \wedge (pork \vee beef)$ Oil-free, include a protein								✓
Runtime:			$O(Nk2^{\mathcal{C}})$	$O(Nk\mathcal{C})$	$O(Nk)$	$O(Nk)$	$O(E)$	$O(E)$	$O(Nk)$

Table 1: Expressivity and runtime of various decoding methods. *AND*: Output includes all terms in a set; *Positive Set AND*: Output includes at least one term from each set; *Predicate Logic Formula*: Any combination of positive and negative constraints. E is the number of editing steps, usually much greater than the sequence length N.

groups altogether. Each group stores at most k candidates that are expanded at each timestep. GBS has a faster runtime of $O(Nk\mathcal{C})$, but this approach biases towards sequences satisfying constraints greedily, and collapses into very similar search paths that are often times globally sub-optimal, which results in dropped language quality.

Post and Vilar (2018) propose dynamic beam allocation to reduce GBS's explicit dependence on $\mathcal{C}$. Beam search is done over a single beam, with the k slots of this beam dynamically allocated over the $\mathcal{C}+1$ groups explicitly used by GBS. This approach was made GPU-efficient by Hu et al. (2019a). Still, the language quality issue of GBS remains, and can be worse in practice as fewer hypotheses are considered at each step.

Miao et al. (2019) propose Constrained Generation by Metropolis-Hastings Sampling (**CGMH**). This approach begins by inserting all positive-constraint keywords in random order. Edits are randomly sampled to replace, insert, or delete words to make the sentence fluent; the probability of each action is computed on top of a language model. Sha (2020) proposes using gradient of a objective function to guide where and how to edit instead of random sampling. These approaches have runtime independent to number of constraints; yet they can involve repeated deletions and insertions, reducing efficiency. Generation quality is also sensitive to initial keyword order and sampled edits.

3.2 Applications of Constrained Generation

Lexically constrained generation can be broadly applied to prior conditional text generation tasks. Examples include incorporating pre-specified lexical constraints (Anderson et al., 2017; Post and Vilar, 2018), user-provided terminology constraints (Hasler et al., 2018; Dinu et al., 2019), noisy automatic constraints (Li et al., 2019) in translation output. A major use case of lexical constrained de-

coding is paraphrase generation (Hu et al., 2019a; Kajiwara, 2019; Hu et al., 2019b; Miao et al., 2019), by negatively constraining words in the source to enforce paraphrasing. Another use case is image captioning, with novel scenes or out-of-domain objects (Anderson et al., 2017), or requiring explicit grounding to objects in the scene (Ren et al., 2015; Krause et al., 2016). In addition, Balakrishnan et al. (2019) leverage constrained decoding to improve semantic correctness for response generation.

4 Experiments I: Constrained Commonsense Generation

COMMONGEN (Lin et al., 2020) is a benchmark dataset designed as a test of generative commonsense reasoning. Given a set of common concepts (e.g., dog, frisbee, catch, throw); the task is to generate a coherent sentence describing an everyday scenario using these concepts (e.g., "a man throws a frisbee and his dog catches it").

Problem Formulation The input is an unordered set of n concepts $\mathbf{x} = \{a_1, a_2, \ldots, a_n\}$, where each concept a_i is a common object (noun) or action (verb). The expected output is a simple, grammatical sentence $\mathbf{y} \in \mathcal{Y}$ that describes a common scenario using all given concepts in $\mathbf{x}$ with correct morphological inflections.

To apply NEUROLOGIC DECODING, we impose that each a_i must appear in output $\mathbf{y}$ under some morphological inflection. Let $\tilde{a}_i = \{\tilde{a}_1^i, \ldots \tilde{a}_{|\tilde{a}_i|}^i\}$ denote all inflections of a_i. $\mathbf{y}$ covers concept a_i, if at least one of $\{\tilde{a}_1^i, \ldots \tilde{a}_{|\tilde{a}_i|}^i\}$ appears. Formally,

$$\forall a_i \in \mathbf{x}, \ \exists \tilde{a}_j^i \in \tilde{a}_i, \ D(\tilde{a}_j^i, \mathbf{y})$$

where $D(\tilde{a}_j^i, \mathbf{y})$ is a boolean-value function indicating whether $\mathbf{y}$ contains $\tilde{a}_j^i$ or not, as defined above.[2]

---

[2]This gets converted into $\wedge_{i=1}^{n} \left(\vee_{j=1}^{|\tilde{a}_i|} D(\tilde{a}_j^i, \mathbf{y}) \right)$.

Model	ROUGE - L	BLEU - 3 & 4		METEOR	CIDEr	SPICE	Coverage
GPT-2	40.3 → 42.8	34.2 → 36.7	24.7 → 26.7	27.6 → 30.2	13.4 → 14.7	27.1 → 30.3	82.2 → 97.7
BERT-Gen	42.4 → 43.8	37.5 → 38.9	27.0 → 28.2	29.5 → 30.9	14.9 → 15.5	29.8 → <u>31.4</u>	89.2 → 97.3
UniLM	44.3 → **45.8**	40.6 → **42.8**	29.9 → **31.5**	30.1 → **31.7**	15.5 → <u>16.6</u>	30.6 → **32.5**	90.5 → 97.8
UniLM-v2	43.5 → 44.2	39.2 → 39.5	28.3 → 28.5	30.6 → <u>31.3</u>	15.2 → **16.8**	30.8 → 31.1	92.8 → 97.9
BART	43.3 → 44.7	39.9 → <u>41.3</u>	29.1 → <u>30.6</u>	30.4 → 31.0	15.2 → 15.9	30.6 → 31.0	95.0 → **98.7**
T5	43.9 → <u>44.8</u>	36.6 → 38.5	26.9 → 28.1	28.9 → 30.7	14.3 → 15.5	29.5 → 30.8	89.7 → <u>98.5</u>

Table 2: Experimental results of different supervised models on the CommonGen test set. Under each column, $\alpha \rightarrow \beta$ shows the performance using the conventional beam search (α) compared to the enhanced performance using NeuroLogic Decoding (β). NeuroLogic always improves the performance across all models and all metrics — with no exception. The best models are **bold** and second best ones are <u>underlined</u> within each metric.

Domain Adaption	Model	ROUGE - L	BLEU - 3 & 4		METEOR	CIDEr	SPICE	Coverage
No	GPT	26.7 → 41.3	3.0 → 25.1	1.1 → 15.9	9.2 → 28.8	0.9 → 11.7	8.0 → 29.7	8.4 → **97.4**
	GPT-2	19.7 → **42.9**	4.1 → <u>34.4</u>	1.5 → <u>23.5</u>	11.2 → <u>30.7</u>	0.4 → <u>13.6</u>	7.1 → <u>31.4</u>	8.3 → 96.0
Yes	GPT-2	29.8 → <u>42.4</u>	9.5 → **36.1**	4.0 → **25.1**	11.7 → **31.3**	1.7 → **13.9**	8.0 → **31.8**	9.3 → <u>96.1</u>

Table 3: Experimental results in zero-shot (unsupervised) setting on the CommonGen test set with and without language domain adaption. Under each column, $\alpha \rightarrow \beta$ shows the performance using the conventional beam search (α) compared to the enhanced performance using NeuroLogic Decoding (β).

Decode Method	ROUGE-L	BLEU-3/4		METEOR	CIDEr	SPICE	Coverage
Greedy Decoding	35.3	25.2	16.7	25.8	10.2	24.4	80.3
Top-k Sampling	33.8	22.5	14.4	24.9	9.2	22.7	79.4
Top-p Sampling	35.3	25.0	16.5	25.7	10.2	24.1	80.1
Beam Search	<u>40.3</u>	<u>34.2</u>	<u>24.7</u>	<u>27.6</u>	<u>13.4</u>	<u>27.1</u>	82.2
Hokamp and Liu	37.6	25.6	16.8	25.9	11.1	25.1	97.2
Post and Vilar	38.3	28.1	18.6	26.7	11.8	26.0	<u>97.4</u>
Hu et al.	38.2	27.8	18.4	26.7	11.7	26.1	<u>97.4</u>
NeuroLogic	**42.8**	**36.7**	**26.7**	**30.2**	**14.7**	**30.3**	**97.7**

Table 4: Performance of different decoding methods using supervised GPT2-L on the CommonGen test set.

Dataset The CommonGen dataset consists of 35,141 concept-sets (32,651 in *train*, 993 in *val*, 1,497 in *test*) associated with 77,449 sentences. The average size of the concept-sets in the test set is 4.04, with an average of four sentences per concept-set and an average sentence length of 13.34 words.

Approach and Baseline The standard pipeline of approaching this problem is to consider it as a conditional sentence generation task. We experiment with several recent pre-trained language models, including GPT-2 (Radford et al., 2019), UniLM (Dong et al., 2019), UniLM-v2 (Bao et al., 2020), BERT-Gen (Bao et al., 2020), BART (Lewis et al., 2020), and T5 (Raffel et al., 2019). All models are finetuned with their default hyperparameters. We compare with commonly used decoding methods, including beam search, sampling, and also previously proposed constrained decoding methods. We use several widely-used automatic metrics to automatically assess the performance, such as

BLEU, ROUGE, METEOR, which mainly focus on measuring surface similarities. We also include metrics specially designed for captioning task, such as CIDEr, and SPICE. Following Lin et al. (2020), we report the concept Coverage, which is the average percentage of input concepts that are present in lemmatizatized outputs.

4.1 Results I: NeuroLogic vs Other Decoding Methods

In Table 4, we first present comparisons across different decoding methods based on a supervised sequence-to-sequence model, GPT-2. The key observations are:

1. NeuroLogic outperforms all other previous decoding methods, both constrained and unconstrained, with respect to all metrics and often with a significant margin.

2. NeuroLogic not only attains high constraint satisfaction (Coverage), it also improves the generation quality as quantified over ROUGE, BLEU, METEOR, CIDEr, and SPICE.

3. In comparison, all previous constrained decoding methods (Hokamp and Liu, 2017; Post and Vilar, 2018; Hu et al., 2019a) attain high constraint satisfaction at the cost of generation quality; being outperformed here by conventional beam search with a large margin.

The second and the third points above demonstrate that the improved logical expressiveness of NeuroLogic together with the effective search strat-
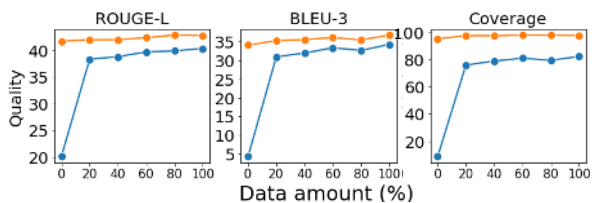
Figure 4: Performance (y-axis) of supervised GPT2-L on COMMONGEN, with a varying amount of training data for supervision (x-axis). The **orange line** denotes decoding with NEUROLOGIC, and the **blue line** denotes decoding with conventional beam search.
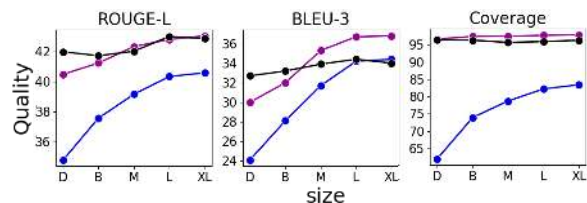

Figure 5: Performance (y-axis) of GPT-2 with varying model sizes (x-axis). The **purple line** and **blue line** denote decoding from a supervised model with and without NEUROLOGIC DECODING respectively. The **black line** denotes decoding with NEUROLOGIC in zero-shot (unsupervised) setting.

egy leads to generation that is both higher quality and satisfies the constraints the most effectively.

4.2 Results II: NEUROLOGIC across Different Supervised Models

Table 2 presents experiments across various state-of-the-art pre-trained language models. In this experiment, all models are supervised on the COMMONGEN training dataset. Under each column, $\alpha \to \beta$ shows the performance using the conventional beam search (α) compared to the enhanced performance using NEUROLOGIC DECODING (β).

As before, NEUROLOGIC always improves the performance across all models and all metrics with no exception – both in terms of constraint satisfaction as well as generation quality. The improvement is especially substantial when the generation quality is relatively low due to smaller model capability or less efficient model architecture or pre-training.

4.3 Results III: NEUROLOGIC with Unsupervised Models

In this experiment, we test how well NEUROLOGIC works with unsupervised pre-trained language models, with and without domain adaptation. Table 3 presents experimental results of zero-shot (i.e., unsupervised) constrained generation. With unconstrained decoding, we have zero controllability over

the unsupervised language models, as they ignore the problem input and generate irrelevant text. With NEUROLOGIC, on the other hand, we can dramatically improve the performance on all metrics. Fig 6 demonstrates some generated examples.

In zero-shot setting without any finetuning, the language style of pre-trained LMs might differ from that of COMMONGEN. To further improve the performance, we conduct language domain adaption by fine-tuning the language models on the training-set COMMONGEN language – ignoring all concept sets. We observe that after domain adaption, NEUROLOGIC in zero-shot setting outperforms unconstrained generation with supervised finetuned LMs, which suggests that inference-time algorithms can provide a more compute-efficient avenue to draw better from neural models.

4.4 Results IV: Ablation

The amount of training data Figure 4 compares the performance (y-axis) of supervised GPT-2 with NEUROLOGIC (**orange line**) and with conventional beam search (**blue line**) as a function of the increasing amount of training data (x-axis). Notably, even after being supervised on 100% of the training data, the supervised GPT-2 does not successfully learn the COMMONGEN constraints ('Coverage') and is even outperformed by the zero-shot GPT-2 (i.e., using 0% training data) with NEUROLOGIC.

The model size Figure 5 compares the performance (y-axis) of GPT-2 with varying model sizes (x-axis). Regardless of the model size, NEUROLOGIC (**purple line** and **black line**) boosts performance considerably over conventional beam search (**blue line**). More over, if using NEUROLOGIC, the performance of unsupervised models (**black line**) becomes comparable to that of supervised models (**purple line**). Remarkably, *unsupervised* models with NEUROLOGIC based on *smaller* networks (**black line**) often outperform *supervised* models with conventional beam search based on considerably *larger* networks (**blue line**).

5 Experiments II: Recipe Generation

We next study cooking recipe generation, a paragraph-level generation task. Given a dish name and a list of ingredients, the task is to generate cooking instructions for the given recipe.

Decode Method	ROUGE-L	BLEU-3/4		METEOR	Coverage	Extra
Top-k Sampling	27.5	15.2	9.5	19.2	84.8	16.0
Top-p Sampling	28.7	17.6	11.7	19.4	86.4	15.4
Beam Search	29.4	17.4	12.0	19.7	86.5	14.3
Post and Vilar	26.1	13.6	8.8	16.5	89.6	1.15
Hu et al.	26.1	13.6	8.8	16.5	89.6	1.13
NEUROLOGIC	32.1	19.5	13.8	19.8	95.8	0.6

Table 5: Experimental results of different decoding methods with RecipeGPT on the Recipe1M+ test set. *Coverage* indicates the average percentage of ingredients that are covered in the generated recipe, while *Extra* corresponds to the average ratio of hallucinated ingredients over the number of given ingredients.

Supervised?	Model	ROUGE-L	BLEU-4	METEOR
Yes	GPT-2	70.5	72.6	87.6	92.4	60.0	64.0
Yes	BART	72.9	70.2	89.5	87.0	60.2	54.2
Yes	T5	70.9	69.9	82.4	79.7	54.6	50.4
Yes	Kiddon et al.	-	90.6	77.8	62.1	54.4
No	GPT-2 + NEUROLOGIC	73.9	71.8	94.8	90.8	66.6	62.0

Table 6: Experimental results of dialogue generation, the right column is the performance for hotel system, and the left column is for restaurant system.

Problem Formulation The input is the recipe title, an unordered set of ingredients $E = \{e_1, ..., e_{|E|}\}$ where e_i can be a single- or multi-word ingredient phrase (e.g., 'onions', 'black pepper'). Let $\mathcal{G}$ denote the set of all ingredients. The expected output is a paragraph $\mathbf{y} \in \mathcal{Y}$ that describes multi-step cooking instructions.

To apply NEUROLOGIC DECODING, we constrain output $\mathbf{y}$ to contain all given ingredients e_i in E, and no other ingredients, i.e. no ingredients in $\mathcal{G} \setminus E$. Ingredients can be referred to with generic terms (e.g., 'vegetables' may refer to 'onions', or 'carrots') and we denote the generic name for ingredient e_i as e_i^T. Formally, the constraint is

$$\left(\forall e_i \in E, D(e_i, \mathbf{y}) \vee D(e_i^T, \mathbf{y}) \right)$$
$$\wedge \left(\forall e_i \in \mathcal{G} \setminus E, \neg D(e_i, \mathbf{y}) \right)$$

Dataset, Approach and Baseline We use Recipe1M+, a large-scale, structured corpus of over one million cooking recipes. On average each recipe has 118 words and 9 ingredients. RecipeGPT (Lee et al., 2020) is a GPT-2 model fine-tuned on Recipe1M+, for generating recipes. Its default decoding algorithms are beam search and sampling, which serve as the baselines for evaluating our method. In addition, we compare against previously proposed constrained decoding methods with RecipeGPT. Besides common evaluation metrics for generation task, we introduce explicit measures of given-ingredient coverage and usage of extra/hallucinated ingredients.

Result Table 5 presents the experimental results. We can see that NEUROLOGIC outperforms all baselines in all metrics. The delta is quite remarkable on coverage of given ingredients and usage of extra ingredients. With NEUROLOGIC, we are able

to cover almost all ingredients in generated instructions and guarantee not to use any other ingredients, which leads to more accurately controlled generation. By plugging NEUROLOGIC into existing generation system, we can get immediate boosts in controllability and generation quality with no extra computational cost.

6 Experiments III: Data-Grounded Dialogue Response Generation

In dialogue response generation for hotel and restaurant information systems (Wen et al., 2016), we generate a natural language response given a query type (e.g., informing or querying) and a list of facts to convey (e.g., a hotel's name and address).

Problem Formulation The input is a query type, an unordered set of facts $F = \{f_1, ..., f_{|F|}\}$, where each f_i contains attribute and value (i.e. accepts_credit_cards="yes", name="red victorian bed breakfast"). The expected output is a dialogue responses $\mathbf{y} \in \mathcal{Y}$ containing given information.

The constraint here is that all given facts f_i must be included in responses $\mathbf{y}$ in proper natural language form f_i^N. We use a very simple template to turn f_i to natural language form f_i^N. (i.e. the natural language form for accepts_credit_cards="no" is "doesn't accept credit cards"). Formally,

$$\forall \, f_i \in F, \, D(f_i^N, \mathbf{y})$$

Dataset, Approach and Baseline We use the hotel and restaurant dialogue system corpus and the same train-dev-test split from (Wen et al., 2016). There are 8 query types and 12 attribute types.

The standard paradigm for dialogue generation is to consider it as a conditional sentence generation task and finetune a seq2seq model. While this pipeline works effectively with existing data, once we have user queries with new query types or new attribute types, the seq2seq model would not be able to generate plausible responses. The

Model	Accuracy(%; ↑)	Δ_S (F1; ↓)
En-De		
Google Translate	59.4	12.5
Microsoft Translator	74.1	30.2
Junczys-Dowmunt et al.	60.5 → **91.0**	13.3 → **4.3**
Junczys-Dowmunt et al.+GT Gender	60.5 → **95.0**	13.3 → **2.4**
En-Fr		
Google Translate	63.6	26.7
Microsoft Translator	44.7	29.7
Junczys-Dowmunt et al.	53.0 → **81.0**	19.3 → **1.7**
Junczys-Dowmunt et al. +GT Gender	53.0 → **89.9**	19.3 → **1.5**

Table 7: Performance of Gender Bias Removal on WinoMT, adapted from Stanovsky et al.. Accuracy refers to correctly translating a person's gender, Δ_S is the difference in performance (F_1) between stereotypical and non-stereotypical gender roles (lower is better). The arrow ($\rightarrow$) shows the performance before and after NEUROLOGIC, where gender is either inferred from a coreference model (default) or provided (GT Gender).

situation can happen frequently with a dialogue generation system in application. Thus, we are interested in zero-shot dialogue generation. We give a hand-crafted initial prompt to a pre-trained LM based on the query type and apply NEUROLOGIC DECODING to force given facts to include in generation. The pre-trained LM we use here is GPT-2 (Radford et al., 2019).

The baseline we compare against is seq2seq fine-tuned LMs with vanilla beam search, including GPT-2 (Radford et al., 2019), BART (Lewis et al., 2020) and T5 (Raffel et al., 2019). We also compare with previous SOTA (Kiddon et al., 2016) on dialogue response generation.

Result Table 6 presents the experimental results. We can see that zero-shot generation with NEUROLOGIC outperforms or matches supervised baselines. This suggests that plugging NEUROLOGIC DECODING into pretrained LMs can lead to a powerful dialogue generation system, we do not actually need massive finetuning with extra computational cost to do that.

7 Experiment IV: Reducing Gender Bias in Machine Translation

Problem Formulation We adopt the task setup and dataset of Stanovsky et al. (2019). The input x is an English sentence describing a scenario with human entities $N = \{n_1, \ldots, n_{|N|}\}$ who are identified by roles. The desired output is a translation **y** which uses the correct gender inflections in the target language (here, German or French).

We obtain indicators of people's gender identity

through coreference resolution, linking each entity with their gendered pronoun.[3] We then constrain the correctly-gendered human entities to appear in output **y**. For a human entity n_i, let n_i^F denote its female inflection in the target language, and n_i^M denotes its male inflection. Let F denotes the set of human entities associated with female characters, and M denotes the set of entities associated with male. Formally, the constraint is

$$\Big(\forall n_i \in F, D(n_i^F, \mathbf{y}) \wedge \neg D(n_i^M, \mathbf{y})\Big) \wedge$$
$$\Big(\forall n_i \in M, D(n_i^M, \mathbf{y}) \wedge \neg D(n_i^F, \mathbf{y})\Big)$$

Dataset We use Stanovsky et al. (2019)'s dataset, which is built over the English-only coreference gender-bias studies: Winogender (Rudinger et al., 2018) and Wino-Bias (Zhao et al., 2018).

Result Our results are shown in Table 7. When provided gender markers given by a coreference model, NEUROLOGIC increases the accuracy of handling gender correctly by **30.5** percentage for German, and **28.0** percentage for French. This even outperforms commercial translation systems – the best result, over any language or system, is Microsoft Translator for German with 74.1% accuracy, whereas NEUROLOGIC enables the baseline model to get 91% accuracy. The performance increases again by an additional 4% (German) and 8.9% (French) when ground-truth gender markers are used during constrained decoding. Last, the diagnostic results also show that NEUROLOGIC is particularly effective at reducing (over)reliance on stereotypical gender roles, with a significant decrease in performance difference Δ_S between stereotypical and non-stereotypical gender roles. These results suggest that NEUROLOGIC DECODING is a plug-and-play approach for reducing gender bias in existing translation systems.

8 Conclusion

We propose NEUROLOGIC DECODING, an efficient and general method for generating with arbitrary predicate logic constraints. We demonstrate its intuitive application to 4 different tasks as an extension to existing models, showing broad and consistent improvement to decoding quality.

---

[3] We could use any off-the-shelf coreference resolution model for this. However, since the English examples in Stanovsky et al. (2019) follow the Winograd schemas format, we use a RoBERTa model finetuned on Winograd Schema Challenge for this, with 78.4% accuracy.

Acknowledgements

References

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.

Anusha Balakrishnan, Jinfeng Rao, Kartikeya Upasani, Michael White, and Rajen Subba. 2019. Constrained decoding for neural NLG from compositional representations in task-oriented dialogue. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 831–844, Florence, Italy. Association for Computational Linguistics.

Hangbo Bao, Li Dong, Furu Wei, Wenhui Wang, Nan Yang, Xiaodong Liu, Yu Wang, Songhao Piao, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2020. Unilmv2: Pseudo-masked language models for unified language model pre-training.

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

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 13063–13075. Curran Associates, Inc.

A. Fiacco. 1976. Sensitivity analysis for nonlinear programming using penalty methods. *Mathematical Programming*, 10:287–311.

Eva Hasler, Adrià de Gispert, Gonzalo Iglesias, and Bill Byrne. 2018. Neural machine translation decoding with terminology constraints. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 506–512, New Orleans, Louisiana. 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.

J. Edward Hu, Huda Khayrallah, Ryan Culkin, Patrick Xia, Tongfei Chen, Matt Post, and Benjamin Van Durme. 2019a. Improved lexically constrained decoding for translation and monolingual rewriting. 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 839–850, Minneapolis, Minnesota. Association for Computational Linguistics.

J. Edward Hu, Rachel Rudinger, Matt Post, and Benjamin Van Durme. 2019b. Parabank: Monolingual bitext generation and sentential paraphrasing via lexically-constrained neural machine translation. In *AAAI*.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, André F. T. Martins, and Alexandra Birch. 2018. Marian: Fast neural machine translation in C++. In *Proceedings of ACL 2018, System Demonstrations*, pages 116–121, Melbourne, Australia. Association for Computational Linguistics.

Tomoyuki Kajiwara. 2019. Negative lexically constrained decoding for paraphrase generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6047–6052, Florence, Italy. Association for Computational Linguistics.

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. 2016. Globally coherent text generation with neural checklist models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 329–339, Austin, Texas. Association for Computational Linguistics.

Jonathan Krause, Benjamin Sapp, Andrew Howard, Howard Zhou, Alexander Toshev, Tom Duerig, James Philbin, and Li Fei-Fei. 2016. The unreasonable effectiveness of noisy data for fine-grained recognition. *ArXiv*, abs/1511.06789.

H. Lee, Shu Ke, Palakorn Achananuparp, P. K. Prasetyo, Y. Liu, E. Lim, and L. R. Varshney. 2020. Recipegpt: Generative pre-training based cooking recipe generation and evaluation system. *Companion Proceedings of the Web Conference 2020*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation,

and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Huayang Li, Guoping Huang, and Lemao Liu. 2019. Neural machine translation with noisy lexical constraints.

Bill Yuchen Lin, Ming Shen, Wangchunshu Zhou, Pei Zhou, Chandra Bhagavatula, Yejin Choi, and Xiang Ren. 2020. Commongen: A constrained text generation challenge for generative commonsense reasoning. In *Findings of EMNLP*.

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

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.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 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. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *ArXiv*, abs/1910.10683.

Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39:1137–1149.

Rachel Rudinger, Jason Naradowsky, Brian Leonard, and Benjamin Van Durme. 2018. Gender bias in coreference resolution. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 8–14.

Lei Sha. 2020. Gradient-guided unsupervised lexically constrained text generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8692–8703, Online. Association for Computational Linguistics.

Gabriel Stanovsky, Noah A. Smith, and Luke Zettlemoyer. 2019. Evaluating gender bias in machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1679–1684, Florence, Italy. Association for Computational Linguistics.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Lina M. Rojas-Barahona, Pei-Hao Su, David Vandyke, and Steve Young. 2016. Multi-domain neural network language generation for spoken dialogue systems. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 120–129, San Diego, California. Association for Computational Linguistics.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1711–1721, Lisbon, Portugal. Association for Computational Linguistics.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2018. Gender bias in coreference resolution: Evaluation and debiasing methods. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 15–20.

Concept-Set   {lose, board, balance, fall, ride}

Supervised Setting

Decode with Beam Search

[GPT-2]: Someone loses balance and falls off his bike.

[UniLM]: A man is trying to keep his balance as he falls off a board.

[BART]: A man loses his balance and falls off the balance while riding a skateboard.

[T5]: a man loses his balance on the board and falls.

Decode with NEUROLOGIC

[GPT-2]: A man loses his balance as he rides a roller coaster and falls off the board.

[UniLM]: Someone loses balance on the ride and falls off the balance board.

[BART]: A man loses his balance on a ride and falls off the board.

[T5]: a rider loses his balance and falls off the board.

Zero Shot Setting

Decode with NEUROLOGIC

[GPT]: a woman lost her balance riding a horse, falling off the horse, and hitting her head on a board

[GPT-2]: The boy lost his balance riding the bike, falling off the bike and hitting his head on the board.

Concept-Set   {throw, knife, stand, target, front}

Supervised Setting

Decode with Beam Search

[GPT-2]: A man is holding a knife and standing in front of a target.

[UniLM]: A man stands next to a knife and throws it at the target .

[BART]: A man stands in front of a target and throws a knife.

[T5]: a man throws a knife in front of a target.

Decode with NEUROLOGIC

[GPT-2]: A man stands and throws a knife in front of a target.

[UniLM]: A man stands next to a knife and throws it at the front of the target .

[BART]: A man stands in front of a target and throws a knife.

[T5]: a man stands in front of a target and throws a knife.

Zero Shot Setting

Decode with NEUROLOGIC

[GPT]: the girl standing in front of him threw her knife at his target

[GPT-2]: The girl's target was standing in front of her, and she threw a knife at him.

Concept-Set   {bell, bike, sidewalk, ride, ring}

Supervised Setting

Decode with Beam Search

[GPT-2]: A man rides a bicycle down a sidewalk and rings a bell.

[UniLM]: A man rides his bike on a sidewalk and rings the bell on the sidewalk .

[BART]: A man rides his bike on the sidewalk and rings a bell.

[T5]: a ringing bell on a bicycle riding on the sidewalk

Decode with NEUROLOGIC

[GPT-2]: A man rides his bike down a sidewalk and rings a bell.

[UniLM]: A man rides his bike on the sidewalk and rings the bell on his bicycle .

[BART]: A man rides his bike on the sidewalk and rings a bell.

[T5]: a man rides a bike on the sidewalk as the bell rings.

Zero Shot Setting

Decode with NEUROLOGIC

[GPT]: the child's bell rang, and the sidewalk began to fill with people riding their bikes

[GPT-2]: The child rings the bell, rides the bike, and then goes to the sidewalk.

Figure 6: Generation examples of different models in supervised and zero-shot setting with and without NEURO-LOGIC DECODING, on COMMONGEN.