# Language Generation via Combinatorial Constraint Satisfaction: A Tree Search Enhanced Monte-Carlo Approach

**Anonymous Author(s)**
Affiliation
Address
email

## Abstract

Generating natural language under complex constraints is a principled formulation towards controllable text generation. We present a framework to allow specification of combinatorial constraints for sentence generation. We propose TSMH, an efficient method to generate high likelihood sentences with respect to a pre-trained language model while satisfying the constraints. Our approach is highly flexible, requires no task-specific training, and leverages efficient constraint satisfaction solving techniques. To better handle the combinatorial constraints, a tree search algorithm is embedded into the proposal process of the Markov chain Monte Carlo (MCMC) to explore candidates that satisfy more constraints. Compared to existing MCMC approaches, our sampling approach has a better mixing performance. Experiments show that TSMH achieves consistent and significant improvement on multiple language generation tasks.

## 1 Introduction

Supervised techniques still dominate in natural language generation tasks. Despite its success, supervised approaches need to be trained with massive datasets of input-output pairs, which is non-trivial to acquire. In addition, it is hard to guarantee that the output sentences satisfy constraints. Recent approaches first pre-train a language model on a general-purpose dataset, then fine-tune the neural net on a task-specific dataset [1, 2]. These approaches partially mitigate data hunger in training large neural networks. Nevertheless, they still require carefully crafted datasets for fine-tuning.

We present a combinatorial constraint satisfaction approach for language generation. In particular, we sample sentences that attain high likelihoods from a language model and satisfy task-specific constraints. Sampling sentences that attain high likelihoods in the language model ensures the sentence quality. Constraints guarantee that the sentences fit the specific language task. The constraints can be hard ones such as the grammar rules, or soft ones such as attaining positive sentiment scores.

Our method harnesses constraint satisfaction, rather than learning, to guide language generation. In fact, there is no task-specific training in our approach. Our approach is highly flexible since constraints can be switched quickly to be adapted to a different task, even faster than fine-tuning. It also allows us to leverage the latest developments of automated reasoning for language generation. Although the field of language generation is dominated by learning, reasoning should play an equally important role. Human beings can write beautiful words from reasoning over what is needed in the specific writing task, without learning from previous examples.

To better handle the combinatorial constraints, a tree search is embedded into the proposal process of the Markov chain Monte Carlo (MCMC) for constrained language generation, which suggests

Submitted to 34th Conference on Neural Information Processing Systems (NeurIPS 2020). Do not distribute.

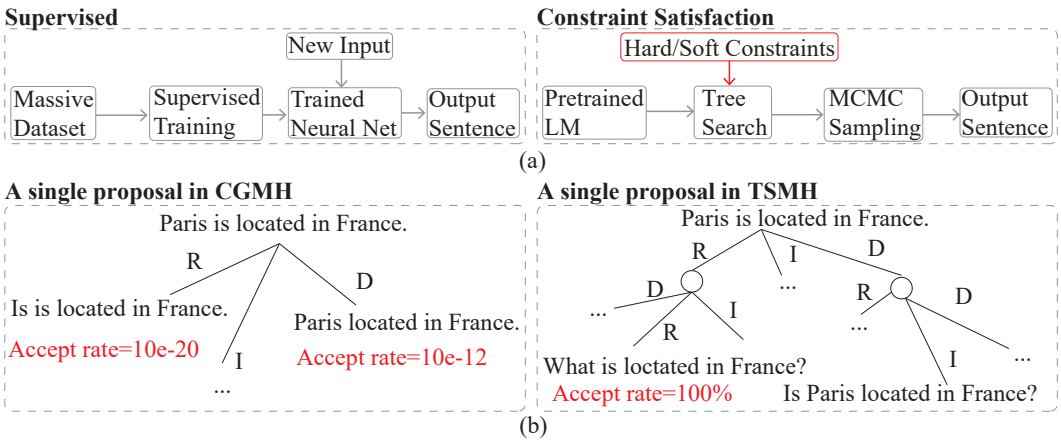


Figure 1: **(a)** Language generation via supervised method and constraint satisfaction. **(b)** Our TSMH traverses the probabilistic space of high-quality sentences more effectively than the baseline CGMH. "R, I, D" means replace, insert and delete operations.

candidate proposals that satisfy more constraints. Our approach is motivated by Sample-Search [3, 4, 5], which integrates backtrack search into importance sampling. Making multiple word-level changes within one proposal step of MCMC allows the direct transition between legitimate sentences, while previous approaches must go through infeasible intermediate states. Such moves are typically rejected by MCMC and therefore result in a slow mixing rate (See Figure 1(b)).

In literature, constrained language generation has been attacked in a supervised way in [6, 7, 8, 9, 10]. There are also various works which model language rules as decomposed tree structures [11] or sentiment tags [12]. Markov Logic network [13, 14] are also used to formulate grammar rules. The Euclidean distance in semantic space is considered as soft constraints in [15, 16, 17].

To summarize, we our contributions are: 1) We define the problem of constraint satisfaction driven natural language generation, and propose a sampling-based approach to tackle the problem with combinatorial constraints. 2) We propose a **T**ree **S**earch enhanced **M**etropolis-**H**astings approach (TSMH) framework, which mixes faster than standard MCMC in the presence of combinatorial constraints. 3) Experiment results on generating interrogative, imperative sentences with keywords, and sentences with given sentiments demonstrate that our TSMH is able to generate sentences that satisfy more hard and soft constraints as well as retain good fluency.

## 2 Language Generation via Combinatorial Constraint Satisfaction

We provide a general framework for the constrained natural language generation. In this framework, sentences are generated by sampling from a probability distribution that is proportional to the score of a pre-trained language model times the constraint score. Formally, let $\pi(x)$ be the probability that sentence $x$ is sampled, it should be propositional to:

$$\pi(x) \propto P_{\text{LM}}(x) \cdot \text{Constraint}(x). \tag{1}$$

Here, $P_{\text{LM}}(x)$ is the score of a language model [2, 18], which measures the quality of sentence $x$. Higher $P_{\text{LM}}(x)$ means the sentence $x$ is better in quality. $\text{Constraint}(x)$ is a task-specific penalty term, which are composed of hard and soft constraint terms:

$$\text{Constraint}(x) = \Phi_{\text{hard}}(x) \cdot \Phi_{\text{soft}}(x). \tag{2}$$

Both the hard constraint score $\Phi_{\text{hard}}(x)$ and the soft constraint score $\Phi_{\text{soft}}(x)$ are float values ranging from 0 to 1. The closer to 1, the more satisfied the constraints are.

Unlike supervised methods which require training with massive data, our framework can solve language generation tasks with no task-specific training. $P_{\text{LM}}(x)$ comes from a general language model, only trained on general-purpose language tasks. There is no fine-tuning of $P_{\text{LM}}(x)$ on the specific task. $\Phi_{\text{hard}}(x)$ is based on crafted constraints. $\Phi_{\text{soft}}(x)$ comes from either user-defined

functions, or pre-trained neural networks, which again is not fine-tuned on the specific task. The overall formulation is composed of the language model and the task-specific constraints. It allows us to sample sentences which are close to natural language while satisfying constraints.

### 2.1 Constraint Formulation

**Hard Constraints.** The hard constraint score of sentence $x$ is computed as: $\Phi_{\text{hard}}(x) = \beta^{M-\sum_i c_i(x)}$, where $\beta \in [0, 1]$. $c_i(x)$ is an indicator variable which takes 1 if the sentence $x$ satisfies the $i$-th constraint, and $M$ is the total number of hard constraints. We use propositional logic to define $c_i(x)$. Given a sentence $x$ with length $m$, let $w_i^V \in \{1, 0\}$ be an indicator variable that the $i$-th word in the sentence is in category $V$. For example, given a keyword `K`, we can enforce its existence in the sentence by: $c(x) = w_1^{\texttt{[K]}} \vee w_2^{\texttt{[K]}} \cdots \vee w_m^{\texttt{[K]}}$. here `[K]` is a set containing the keyword `K`. Furthermore, we can enforce the sentence type to be imperative by: $c(x) = w_1^{\texttt{[VERB]}} \vee (w_1^{\texttt{[ADV]}} \wedge w_2^{\texttt{[VERB]}})$ where the first word in the sentence should be either a verb: $w_1^{\texttt{[VERB]}}$ or an adverb followed by a verb: $w_1^{\texttt{[ADV]}} \wedge w_2^{\texttt{[VERB]}}$. Here, `[VERB]` and `[ADV]` represents the set of verbs and adverbs accordingly.

After defining every hard constraint, to efficiently evaluate if the sentence preserve all the constraints, we use **template** to represent a set of sentences where each word is either given or specified by a word category. We use the number of hard constraints a sentence satisfies at the template level, which helps us reduce the search tree size. For example, a template `[[K],[AUX],[OTH],[OTH]]` represent a series of sentences that the first word is the keyword `K`, the second word is an auxiliary verb and the last two words are the other words.

**Soft Constraints.** A soft constraint assigns a float value between 0 and 1 to indicate the constraint satisfaction degree. Soft constraint $\Phi_{\text{soft}}(x)$ can be derived quite flexibly, either from a user-defined function or a pre-trained neural network. For example, to ensure two sentences are semantically similar, the soft constraint can be the cosine similarity of their sentence vectors. Furthermore, to ensure the sentences generated with specific sentiment, the soft constraint can be the score of a sentiment analysis neural network, representing whether the sentence has the requested sentiment.

### 2.2 Tree Search Enhanced MCMC

Once the probability distribution $\pi(x)$ is defined, we use Markov chain Monte Carlo (MCMC) to sample sentences. Starting from one sentence $x$, MCMC moves to the next sentence $x^*$ by first generating a sample $x^*$ from the proposal distribution $Q(x^*|x)$ and then accept $x^*$ with the acceptance rate: $A(x^*|x) = \min\left\{1, \frac{\pi(x^*)Q(x|x^*)}{\pi(x)Q(x^*|x)}\right\}$, Previous work [19] proposes to use MCMC for constrained sentence generation, namely CGMH algorithm. Their proposal distribution only suggests sentences with one-word modification. Under the combinatorial constraints settings, the CGMH will run into the *low acceptance rate problem*, which is caused by the *locality* of the proposal distribution.

Our Tree Search enhanced Metropolis-Hastings (TSMH) still follows the classical MCMC procedure. The only difference is a new proposal distribution $Q(x^*|x)$ generated from a tree search process. The tree search defines a probability distribution over *templates* of sentence moves. Each template defines a subset of possible moves. The sentences within the same template satisfy the same hard constraints. The proposal probability distribution induced by the tree search algorithm biases towards templates that have high $\text{Constraint}(x)$ scores. The detailed steps are illustrated below:

1) Given a sentence, our algorithm will randomly select several words position for editing. 2) For all the selected word positions, we use Tree Search to efficiently enumerates all possible edit operations: to *insert, delete, or replace* the selected positions and what are the word category in the case of *insert* and *replace*. Every leaf branch of the search tree will be our sentence templates. 3) We extract all the sentence templates and count the number of constraints satisfied for each template. We randomly sample several template with respect to a probability distribution that favors templates satisfying more constraints. 4) we fill in the sampled template with words suggested by a language model. According to the language model score times the soft constraint score $P_{\text{LM}}(\hat{x}) \cdot \Phi_{\text{soft}}(\hat{x})$, we select one filled sentence $\hat{x}$ as proposal.

In summary, our approach alleviates the rejection problem of CGMH by enumerating all possibilities in the space of multiple word change at the template level. This process enables us to handle combinatorial constraints and the Tree search allows us to prune branches of low quality sentences.

| Tasks | Methods | #sample | step | Valid% | $\pi(x)$ | $P_{\text{GPT}-2}(x)$ | Accept% |
|---|---|---|---|---|---|---|---|
| Interrogative | CGMH | 300 | 1 | 18.33 | 2.60E-04 | 1.78E-18 | 5.45 |
| | TSMH(Ours) | 100 | 3 | **92.67** | **1.44E-03** | **5.51E-18** | **24.50** |
| Imperative | CGMH | 300 | 1 | 91.32 | 0.0004 | 9.86E-16 | 5.49 |
| | TSMH(Ours) | 100 | 3 | **97.75** | **0.0060** | **6.60E-15** | **15.66** |
| Sentiment | CGMH | 300 | 1 | 96.33 | 4.93E-19 | 4.57E-22 | 6.72 |
| | TSMH(Ours) | 100 | 3 | **96.67** | **7.94E-04** | **1.82E-18** | **11.09** |

Table 1: Our TSMH outperforms CGMH by generating sentences satisfying more constraints, are of good quality and are likely to be natural language. Column Valid% shows the percentage of generated sentences that satisfy all constraints, which TSMH clearly leads baselines. In addition, TSMH has better acceptance rates (Accept%). The language generated by TSMH is also of good quality and tend to attain higher stationary probability $\pi(x)$.

# 3 Experiments

We evaluate our method on three tasks: interrogative, imperative, and fixed sentiment sentences generation, and summarize the results in Table 1. In each task, we construct the specified type of sentences by sampling starting from keywords and enforcing task-specific constraints. We select the sentence with the highest $\pi(x)$ value among the sentences generated by each algorithm as the output. In general, our method TSMH outperforms baselines and generates sentences that satisfy more constraints, are of good quality and are likely to be close to the natural language. For the metrics in Table 1, Valid% denotes the percentage of generated sentences that satisfy all constraints. $\pi(x)$ is the the stationary probability value. $P_{\text{GPT}-2}(x)$ is the pre-trained GPT-2 language model score, which measures the quality of the sentences. Accept% means the acceptance rate of MCMC.

**Interrogative Sentence Generation.** We enforce that sentences with a high probability to be sampled must satisfy grammar constraints of being interrogative and contain a few given keywords. According to the results, in the experiment with keywords, 92.67% of the output sentences of our TSMH algorithm satisfy all the constraints, while merely 18.33% satisfy constraints for the baseline. This demonstrates that our TSMH generates sentences with more constraints satisfied. In addition, our method has a higher $\pi(x)$ (stationary probability value) and acceptance rate, suggesting that the tree search embedded help MCMC to mix faster.

**Imperative Sentence Generation.** We enforce grammar constraints of being an imperative sentence: the starting word should be either a verb $w_1^{[\text{VERB}]}$ or an adverb followed by a verb $w_1^{[\text{ADV}]} \wedge w_2^{[\text{VERB}]}$. We also enforce keyword constraints in this task. As shown in Table 1, our method has a higher valid percentage of 97.75% compared to 91.32% of the baseline, showing that the sentences generated by our method can satisfy more constraints. Our method has a higher $\pi(x)$ (stationary probability value) and acceptance rate, suggesting our approach has a better mixing behavior.

**Sentence Generation with Given Sentiment.** In this task, we require the sentences to contain the specified keywords and have positive sentiments [20]. We enforce the sentences to attain high scores from a sentiment analysis neural network. We also enforce keyword constraints as hard constraints. We need to emphasize that, our method uses a model pre-trained on a separate dataset for sentiment analysis, which is kept intact in our experiment. No additional fine-tuning to the sentiment analysis model was performed. Our method has a higher sentiment score, suggesting that our method generates sentences with more positive sentiments (better aligned with the target of this experiment). Our model also leads in terms of language model scores, suggesting the language quality is better.

# 4 Conclusions

We propose a framework for constrained language generation via sampling and combinatorial constraint satisfaction. Our strategy is to sample sentences from the constrained space with probability proportional to the language model scores. To handle the combinatorial constraints, a tree search is embedded into the proposal process of MCMC. Experiments demonstrate that our approach generates sentences that satisfy more constraints are likely to be close in quality to the natural language.

# References


[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019.

[2] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[3] Vibhav Gogate and Rina Dechter. Approximate counting by sampling the backtrack-free search space. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*, pages 198–203, 2007.

[4] Vibhav Gogate and Rina Dechter. Samplesearch: A scheme that searches for consistent samples. In *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics, AISTATS*, pages 147–154, 2007.

[5] Vibhav Gogate and Rina Dechter. Samplesearch: Importance sampling in presence of determinism. *Artif. Intell.*, 175(2):694–729, 2011.

[6] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112, 2014.

[7] Mathias Berglund, Tapani Raiko, Mikko Honkala, Leo Kärkkäinen, Akos Vetek, and Juha Karhunen. Bidirectional recurrent neural networks as generative models. In *Advances in Neural Information Processing Systems*, pages 856–864, 2015.

[8] Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. Toward controlled generation of text. In *Proceedings of the 34th International Conference on Machine Learning, ICML*, pages 1587–1596, 2017.

[9] Huangzhao Zhang, Hao Zhou, Ning Miao, and Lei Li. Generating fluent adversarial examples for natural languages. In Anna Korhonen, David R. Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 5564–5569. Association for Computational Linguistics, 2019.

[10] Ning Miao, Yuxuan Song, Hao Zhou, and Lei Li. Do you have the right scissors? tailoring pre-trained language models via monte-carlo methods. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 3436–3441. Association for Computational Linguistics, 2020.

[11] Jay Yoon Lee, Sanket Vaibhav Mehta, Michael Wick, Jean-Baptiste Tristan, and Jaime G. Carbonell. Gradient-based inference for networks with output constraints. In *The Thirty-Third AAAI Conference on Artificial Intelligence*, pages 4147–4154, 2019.

[12] Jinyue Su, Jiacheng Xu, Xipeng Qiu, and Xuanjing Huang. Incorporating discriminator in sentence generation: a gibbs sampling method. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 5496–5503, 2018.

[13] Matthew Richardson and Pedro M. Domingos. Markov logic networks. *Machine Learning*, 62(1-2):107–136, 2006.

[14] Tushar Khot, Niranjan Balasubramanian, Eric Gribkoff, Ashish Sabharwal, Peter Clark, and Oren Etzioni. Exploring markov logic networks for question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 685–694, 2015.

[15] Shrimai Prabhumoye, Yulia Tsvetkov, Ruslan Salakhutdinov, and Alan W. Black. Style transfer through back-translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 866–876, 2018.

[16] David Belanger and Andrew McCallum. Structured prediction energy networks. In *Proceedings of the 33nd International Conference on Machine Learning*, pages 983–992, 2016.

[17] Michael S Amato and Maryellen C MacDonald. Sentence processing in an artificial language: Learning and using combinatorial constraints. *Cognition*, 116(1):143–148, 2010.

[18] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. LSTM neural networks for language modeling. In *13th Annual Conference of the International Speech Communication Association*, pages 194–197, 2012.

[19] Ning Miao, Hao Zhou, Lili Mou, Rui Yan, and Lei Li. CGMH: constrained sentence generation by metropolis-hastings sampling. In *The Thirty-Third AAAI Conference on Artificial Intelligence*, pages 6834–6842, 2019.

[20] Yao Fu, Hao Zhou, Jiaze Chen, and Lei Li. Rethinking text attribute transfer: A lexical analysis. In *the 12th International Conference on Natural Language Generation (INLG)*, October 2019.