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

Maosen Zhang†, Nan Jiang†, Lei Li‡, and Yexiang Xue†

†Purdue University ‡ByteDance AI Lab, China

Language Generation

- Supervised approaches require massive datasets. They can be fine-tuned on task-specific dataset for better performance.
- We sample sentences with high likelihood from a language model and satisfy task-specific constraints. No extra training and fine-tuning is required.

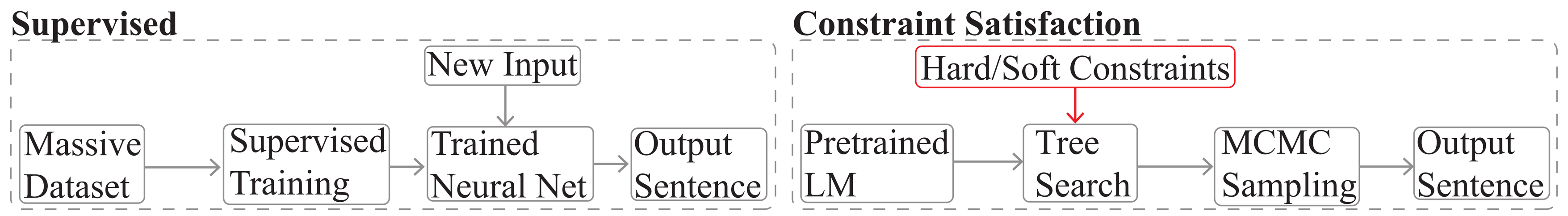

Figure 1. Language generation via supervised method and constraint satisfaction.

Sampling Probability for Sentence

Sentence sampling probability distribution is proportional to:

- the pre-trained language model score $P_{\text{LM}}(x)$, measuring the quality of sentence;
- the constraint satisfaction score: $\text{Constraint}(x) = \Phi_{\text{hard}}(x) \cdot \Phi_{\text{soft}}(x)$ where the hard/soft constraints are from the task requirements.

Let x be a sentence, $\pi(x)$ be the probability that x is sampled:

$$\pi(x) \propto P_{\text{LM}}(x) \cdot \Phi_{\text{hard}}(x) \cdot \Phi_{\text{soft}}(x). \quad (1)$$

Hard Constraint Score

Hard constraints score for sentence x:

$$\Phi_{\text{hard}}(x) = \beta^{M - \sum_i c_i(x)}, \quad \beta \in (0, 1]$$

M is the total number of hard constraints. $c_i(x)$ is an indicator function which takes 1 if the sentence x satisfies the i-th constraint. we use propositional logic to define hard constraint $c_i(x)$.

Literal w_j^V for Hard Constraints. Let $w_j^V \in \{1, 0\}$ be an indicator function that the j-th word in the sentence is in category V. Here V can be:

- a set of keywords: $V = \{today, tomorrow, yesterday\}$;
- a set of words with the same grammar type, like all the adverbs: is, am, are;
- a set of user-defined type, `[QWH]`: when, where, what, why.

Hard Constraint on a Sentence $c_i(x)$.

- Keywords `[K]` in a Sentence: $c(x) = w_1^{\texttt{[K]}} \vee w_2^{\texttt{[K]}} \cdots \vee w_m^{\texttt{[K]}}$
- imperative sentence: $c(x) = w_1^{\texttt{[VERB]}} \vee (w_1^{\texttt{[ADV]}} \wedge w_2^{\texttt{[VERB]}})$
  - The first word is a verb: $w_1^{\texttt{[VERB]}}$;
  - OR the first two words are an adverb followed by a verb: $w_1^{\texttt{[ADV]}} \wedge w_2^{\texttt{[VERB]}}$.
- Interrogative Sentence: $w_1^{\texttt{[QWH]}} \wedge \left((w_2^{\texttt{[AUX]}} \wedge \neg w_3^{\texttt{[AUX]}}) \vee (w_3^{\texttt{[AUX]}} \wedge \neg w_2^{\texttt{[AUX]}}) \right)$
  - The first word is in `[QWH]`;
  - AND the second or third word in the sentence is in `[AUX]`.

Soft Constraint Score

- *Sentence similarity*. It ensures the generated sentence x is semantically close to the reference. The similarity can be the cosine distance between two sentence vectors, which are given by pre-trained semantic understanding model.
- *Sentiment score*. It ensures the generated sentence is close to the given sentiment. The score is given by a pre-trained sentiment analysis model.

Motivation: Breaking the Low Acceptance Barrier

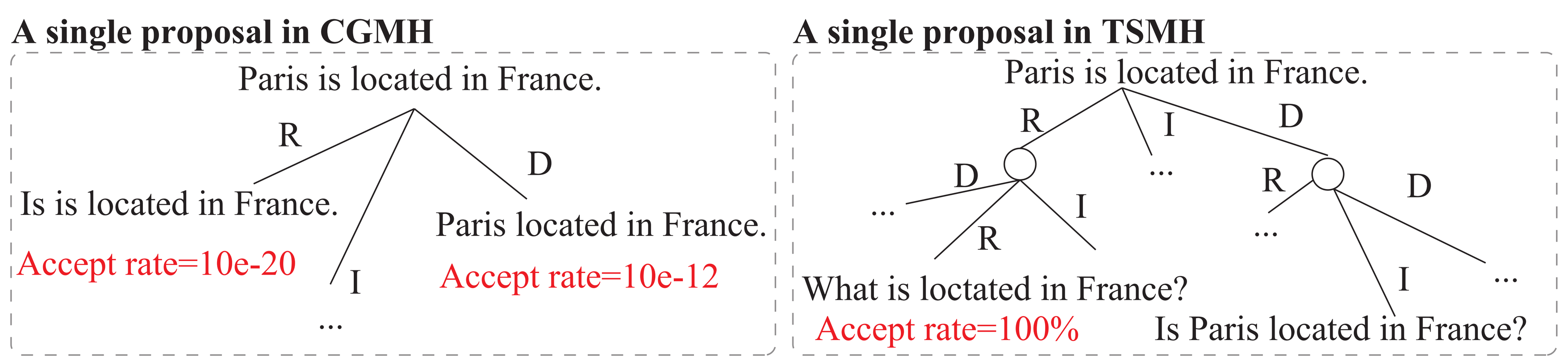

Figure 2. Our method, tree search embedded MCMC (TSMH), outperforms CGMH in generating sentences with complex combinatorial constraints. (Left) CGMH must pass intermediate sentence states, which have very low acceptance rate to reach the intermediate sentence "*Is Paris located in France?*" starting from sentence "*Paris is located in France*". This results in the poor performance of CGMH when handling combinatorial constraints. (Right) By embedding a tree search into MCMC, TSMH can reach the an intermediate sentence from the starting sentence in one step, and with an acceptance rate of 100%. R, I, D mean *replace, insert, delete*.

Efficient Evaluation of Multiple Hard Constraint

To reduce the search tree size, we use **template** to represent a set of sentences satisfying the same hard constraints. To evaluate if the sentence preserve all the constraints, we only check the the template for every set of sentences.

For example, a template: `[[K], is, located, in, [K']]`, represent a series of sentences that the first word is the keyword `K`, the fourth word is another keyword

Experiment - Case Study

keywords	waste, heat, water
CGMH	what waste is there, it seems now?
TSMH(Ours)	where was the waste - water heater?
keywords	median, temperature, winter
CGMH	what do you mean we have median temperature winter and spring, anyways?
TSMH(Ours)	what is the median temperature range in the winter months?
keywords	catholics, concentrated, france
CGMH	the catholics are now mainly concentrated there.
TSMH(Ours)	why are the french roman catholics so densely concentrated in southern france?

Table 1. Case study of generating interrogative sentences with keywords.

Experiment - Summary

Our method TSMH outperforms CGMH by generating sentences that satisfy more constraints, are of good quality and are likely to be natural language.

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

Table 2. Comparison with CGMH over all tasks. Column Valid% shows the percentage of generated sentences that satisfy all constraints, Colum Accept% acceptance rates. Column $P_{\text{GPT-2}}(x)$ language model scores. Column $\pi(x)$,sentence sampling probability.

Extended Experiments

Methods	$\pi(x)$	Valid%	$\log P_{\text{LM}}$
UQA [1]	0.0024	50%	-92.75
TSMH(Ours)	**0.0063**	**83.17%**	**-58.27**

Table 3. Comparison with UQA [1]. TSMH outperforms UQA in terms of the constraint satisfaction, and language model score. UQA is trained on specific interrogative sentences.

Methods	$\pi(x)$	$P_{\text{GPT-2}}(x)$	Sentiment
CtrlGen [2]	3.19E-07	4.64E-22	0.4614
TSMH (Ours)	**1.16E-03**	**7.07E-19**	**0.5254**

Table 4. Compare with CtrlGen [2] over the N2P subtask with acceptance rate, language score and sentiment score metrics. CtrlGen requires training the autoencoder.

Code

check out code at: `https://github.com/Milozms/TSMH`

References

[1] Patrick S. H. Lewis, Ludovic Denoyer, and Sebastian Riedel. Unsupervised question answering by cloze translation. 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 4896–4910. Association for Computational Linguistics, 2019.

[2] 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.

{maosen,jiang631,yexiang}@purdue.edu, lileilab@bytedance.com