Guaranteed Generation from Large Language Models

Minbeom Kim Thibaut Thonet Jos Rozen Hwaran Lee Kyomin Jung Marc Dymetman

NAVER LABS Europe · NAVER · ICLR

Goals of Guaranteed Generation

1. 💯 % strict **guarantees** on constraints satisfaction.
2. **Preserve original distribution** as much as possible.
3. Achieve (1) & (2) with **limited inference costs**.

Basic Notations

Base LLM: $a(y)$, sample $y \sim a(y)$

The guarantee: binary hard constraint $b(y) \in \{0, 1\}$
 E.g. $b(y)$ is a toxicity detector (0 means toxic, 1 means non-toxic)
 E.g. $b(y)$ is a verifier of certain keywords

Guaranteed sampler: sampler that **never** violates $b(y)$

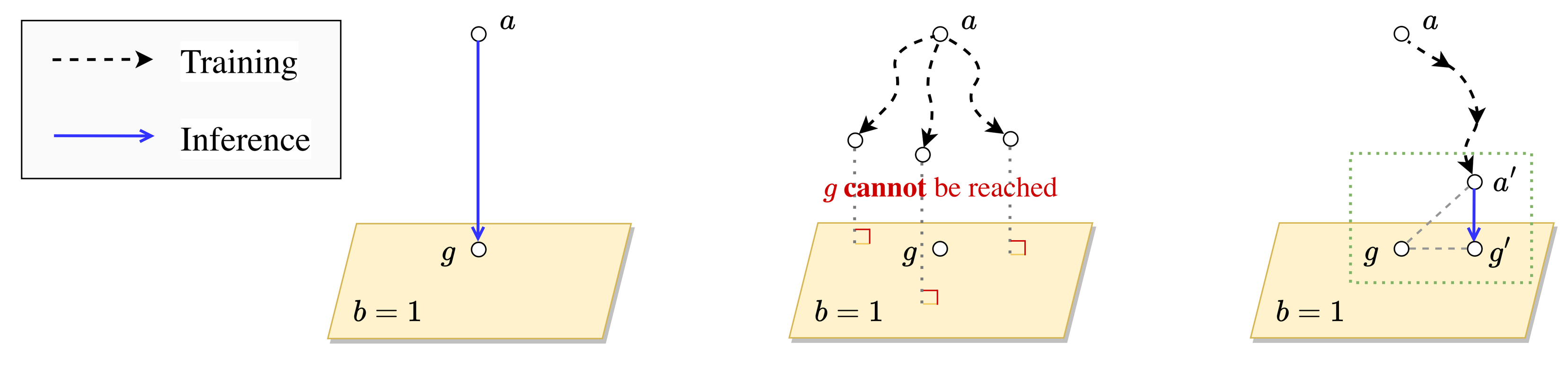
Formalization of Guaranteed Generation

We define Ideal distribution g as

$$g(y) = \begin{cases} 0, & b(y) = 0 \\ \frac{1}{Z_{a,b}} a(y), & b(y) = 1 \end{cases}$$

- g is the **guaranteed** distribution, a conditioned by the fact that y satisfies b, i.e. $g(y) = a(y\,|\,[b(y) = 1])$.
- g is the guaranteed distribution p that minimizes $KL(p\,||\,a)$, g ***preserves the original distribution*** a as much as possible.

Limitation of Autoregressive Models



(1) Rejection sampling (2) Autoregressive training (3) GUARD

Theorem 1. *However, it is impossible to exactly fit $g(y)$ with an autoregressive trained model $a'(y)$.*

Claim 1. To achieve this objective, we need **inference-time** Monte-Carlo (MC) methods that exploits the approximation a' as a *proposal* sampler but only *retains* samples satisfying the constraint b.

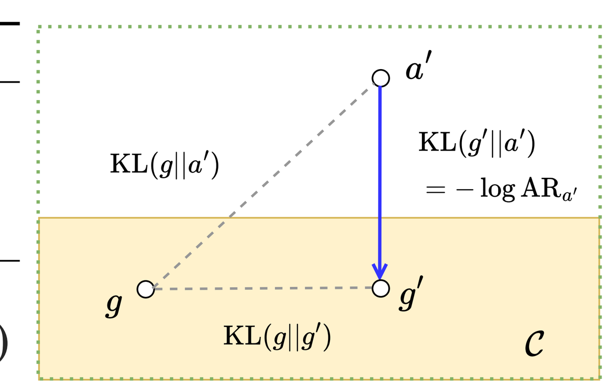
Claim 2. When we apply *rejection sampling* to a, we *obtain* g with significant Inference cost. Even with a efficient proposal a', a significant inference cost must still be incurred to improve the *quality* of samples via an MCMC.

To address this, we introduce 🛡 GUARD to ***amortize*** this inference procedure.

🛡 GUARD Framework and its properties

Algorithm 1 GUARD sampler

```
1: while True do
2:     y ~ a'
3:     if b(y) = 1 return y
```



Theorem 2. $\underbrace{KL(g\,||\,a')}_{objective} = \underbrace{-\log AR_{a'}}_{efficiency} + \underbrace{KL(g\,||\,g')}_{closeness}$

❓ *How can we optimize $KL(g\,||\,a')$?*

CAP: prompting a to be aware of constraint b

SFT: sample a lot of $y \sim a$ and retain only $b(y) = 1$. Then fine-tune those on a
$\nabla_\theta KL(g\,||\,a'_\theta) = -\mathbb{E}_{y\sim a'_\theta}\frac{g(y)}{a'_\theta(y)}\nabla_\theta \log a'_\theta(y)$, which is *distributional* policy gradient loss,

DPG: sample y from **adaptive** proposal a'_θ and update a' with $\nabla_\theta KL(g\,||\,a'_\theta)$.

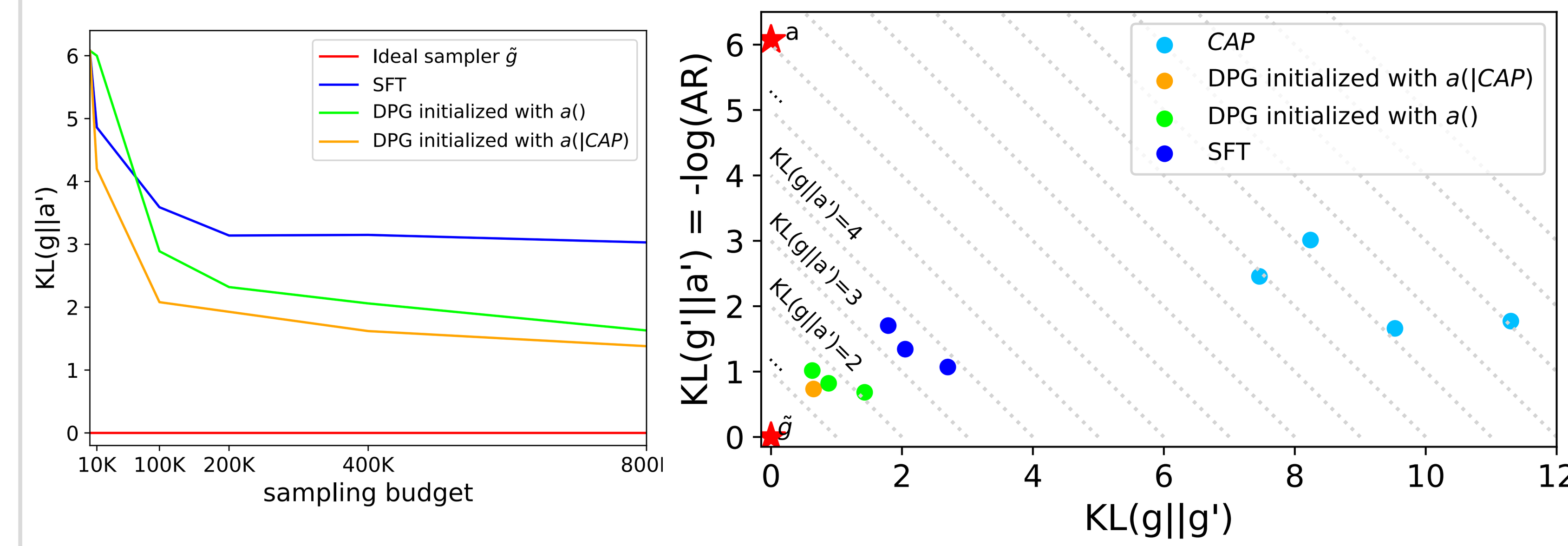
Both SFT and DPG is very slow in the early stage… Can we boost up?

Warm-start DPG: Fine-tune a' with $y \sim a(\cdot\,|\,CAP)$ for ***skipping early stage***.

Experiment & Discussion

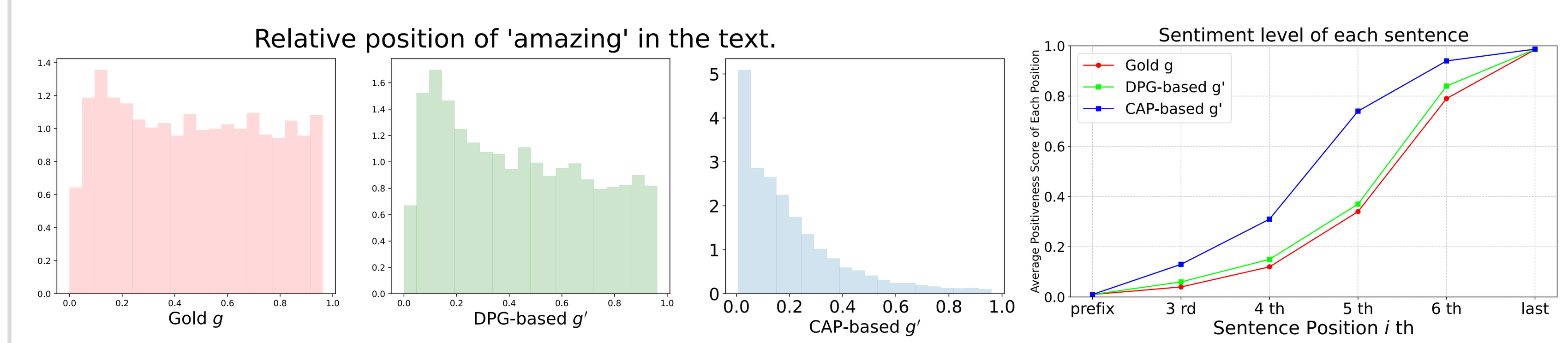
Constraint 1: Lexical constraints **"amazing"** $AR_a = 0.23\,\%$

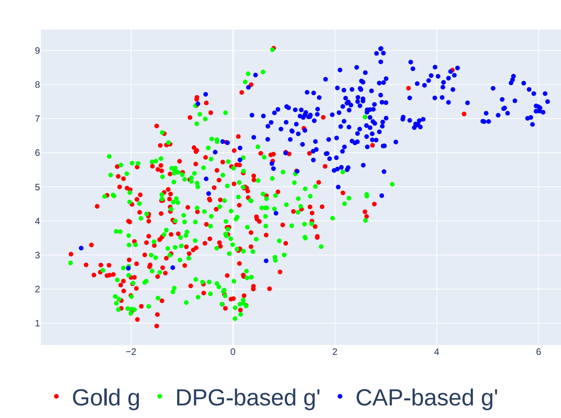
Constraint 2: Story completion from *negative* opening with "**positive**" ending $AR_a = 0.5\,\%$



- DPG's ***adaptive*** proposal excels other baselines.
- ***Warm-start*** helps to skip the inefficient early stage.
- After 🛡 GUARD training,

$AR_{a'}$ of *constraint 1*: $0.23\,\% \rightarrow 0.416\,\%$ **(180x boost-up!)**
$AR_{a'}$ of *constraint 2*: $0.5\,\% \rightarrow 0.306\,\%$ **(60x boost-up!)**
while preserving a high proximity to g





Various analysis show minimal distortion from 🛡 GUARD training

Conclusion. We formalize how to **guarantee** that LLMs perfectly meet specified requirements **without compromising their usefulness.**

GUARD