Large Language Models

Part 5: Generation strategies

David Samuel

Language Technology Group, University of Oslo



Table of contents



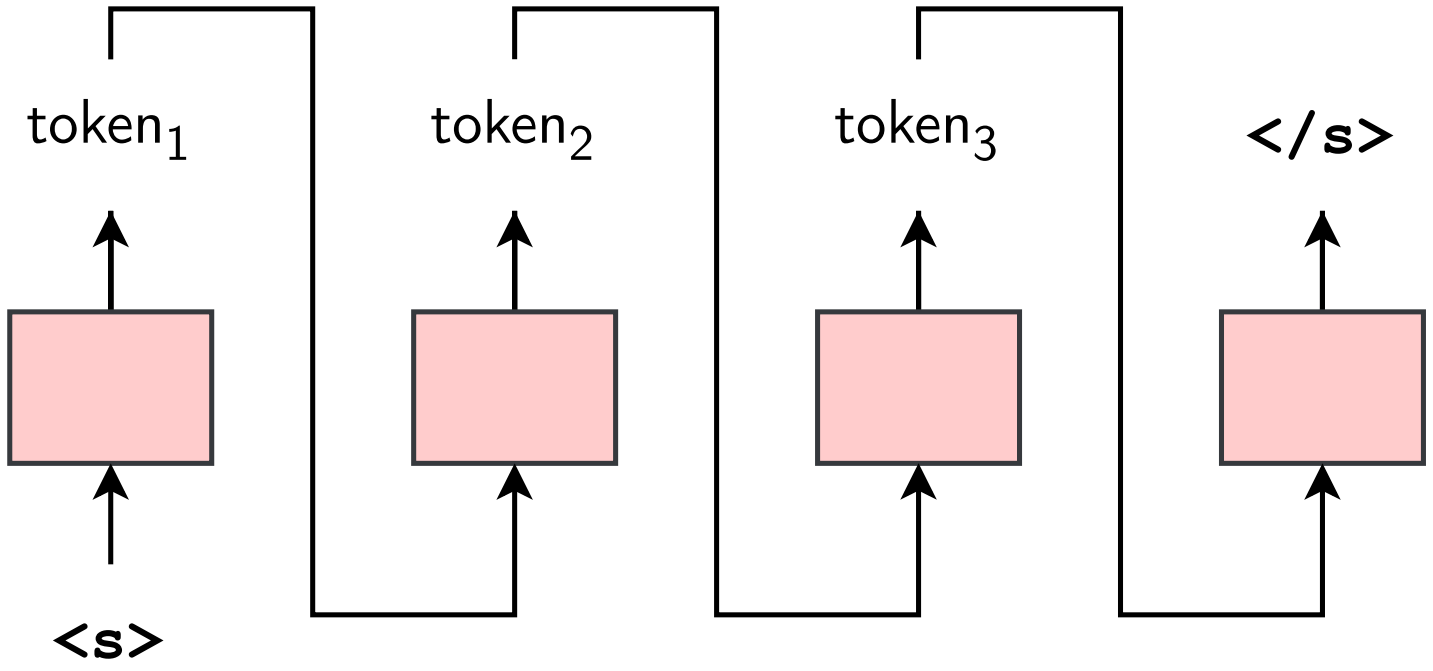
1. Autoregressive generation

2. Searching for the most probable sequence

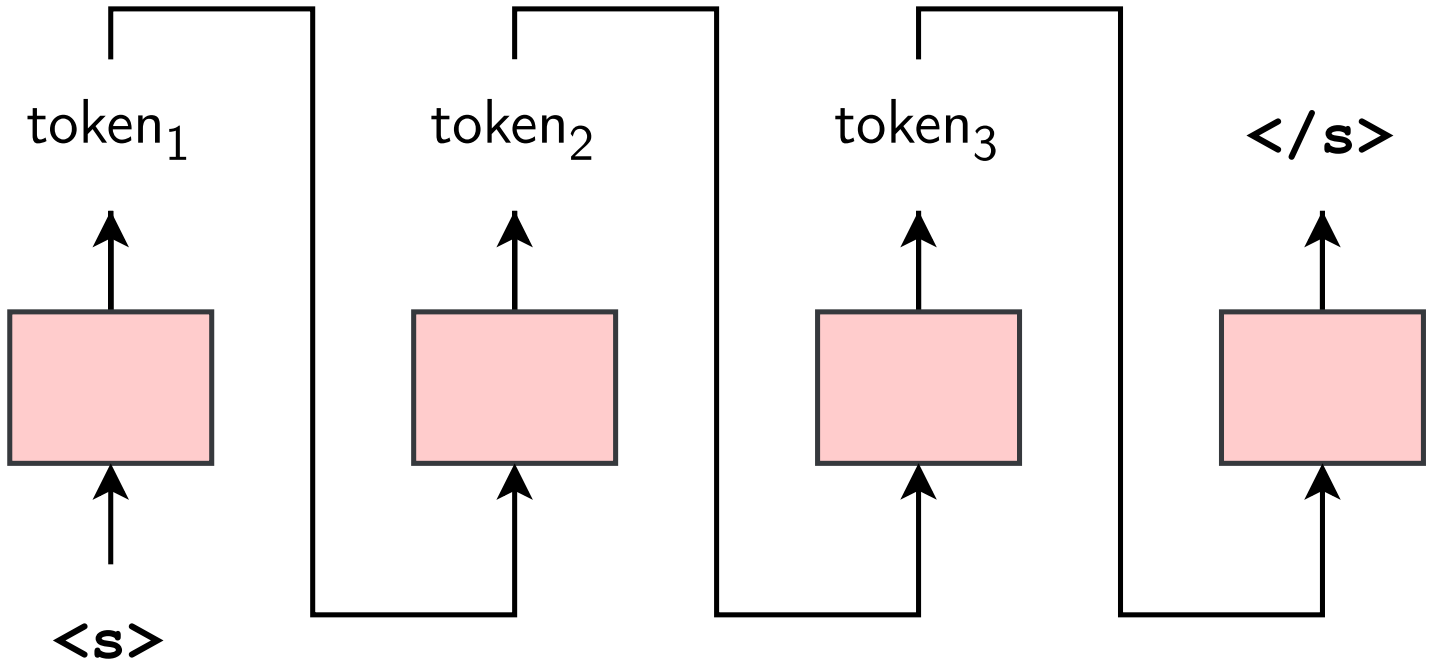
3. Random sampling

4. References

Autoregressive generation



Autoregressive generation



How to choose **token**$_i$?

Table of contents

1. Autoregressive generation

2. Searching for the most probable sequence

3. Random sampling

4. References

Searching for the most probable sequence

Probability of a token sequence

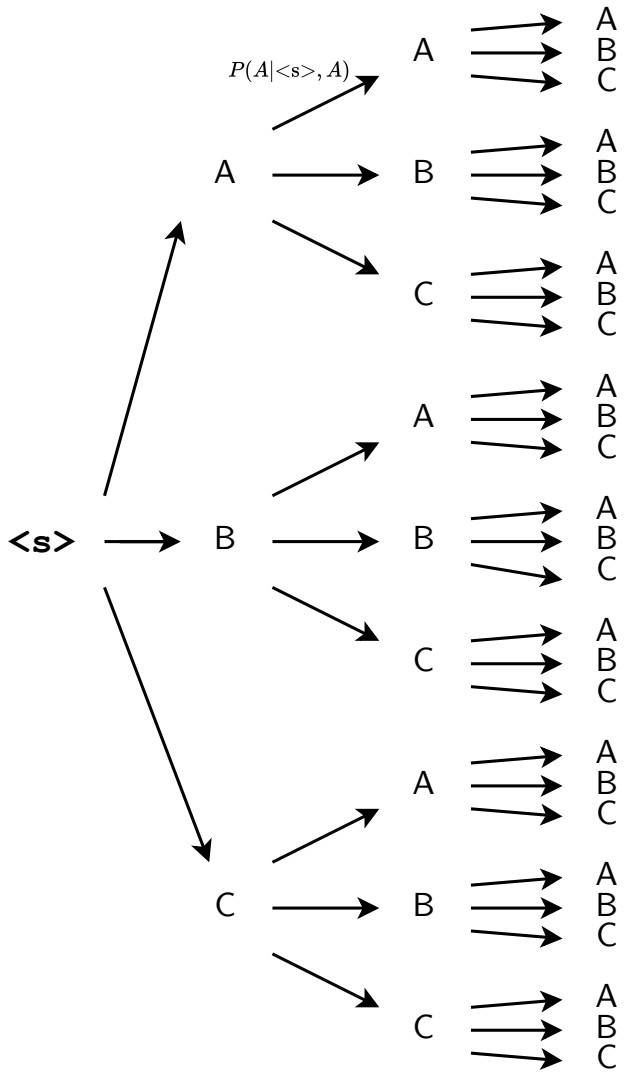
- Remember that the causal language models are trained to estimate the probability of a sequence of tokens:

$$P(w_{0:n}) = \prod_{i=0}^{n} P(w_i|w_{0:i-1}) = P(w_0) \cdot P(w_1|w_0) \cdot P(w_2|w_0, w_1) \dots$$

- But how can we output the most probable $w_{0:n}$?
- If we have vocabulary of size $|V|$ and max sequence length t, we have to go through $O(|V|^t)$ sequences!

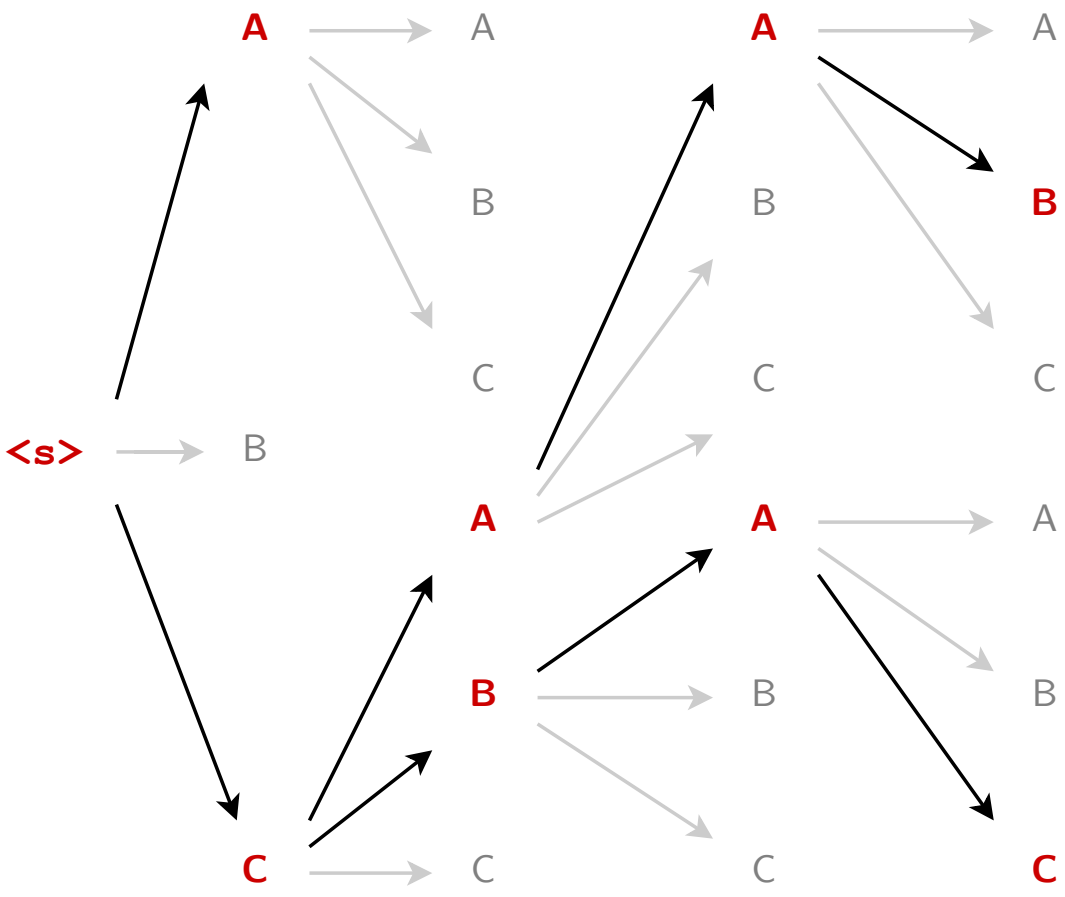
Searching for the most probable sequence

Naive search in the exponentially large space



Searching for the most probable sequence

Pruning of the search-space with beam search (beam size 2)

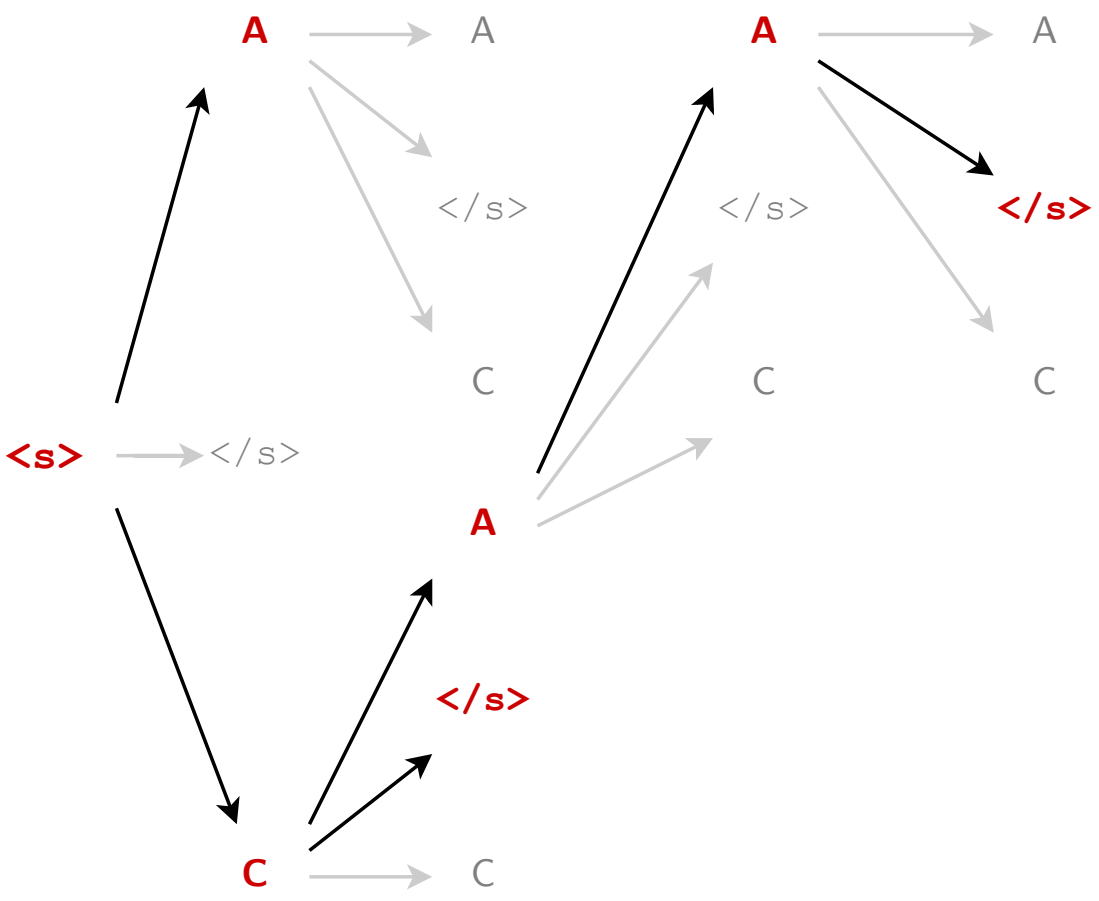


Two candidates:

- <s>CAAB
- <s>CBAC

Searching for the most probable sequence

Beam search with a stop condition



Two candidates:

- <s>CAA</s>
- <s>C</s>



Beam search

- The most popular search method when we want to estimate the most likely output
- An essential part of machine translation
  - Note that seq2seq problems use a slightly different formulation:

$$P(w_{0:n}|\text{source_seq}) = \prod_{i=0}^{n} P(w_i|w_{0:i-1}, \text{source_seq})$$

- We take into account only the k most probably candidates seen so far ("beam size" hyperparameter)
- The complexity drops from $O(|V|^t)$ to $O(k \cdot t)$

Searching for the most probable sequence

Greedy search

- A special case of beam search with beam size of 1
- Simpler to implement than the general beam search and usually substentially faster (no overhead)
- The performance drop is usually not large with a good model

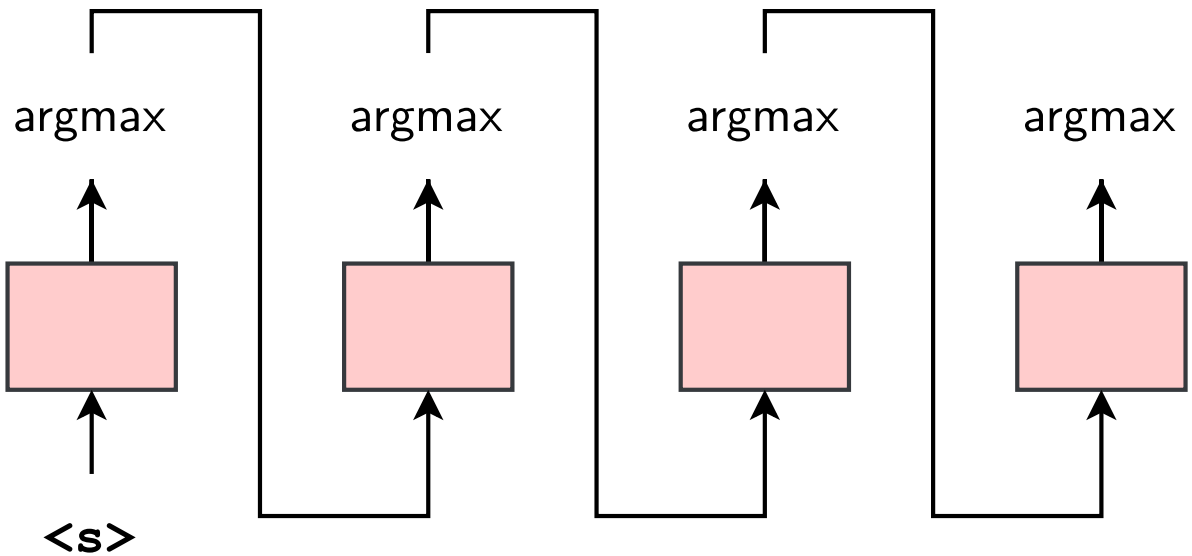


Table of contents



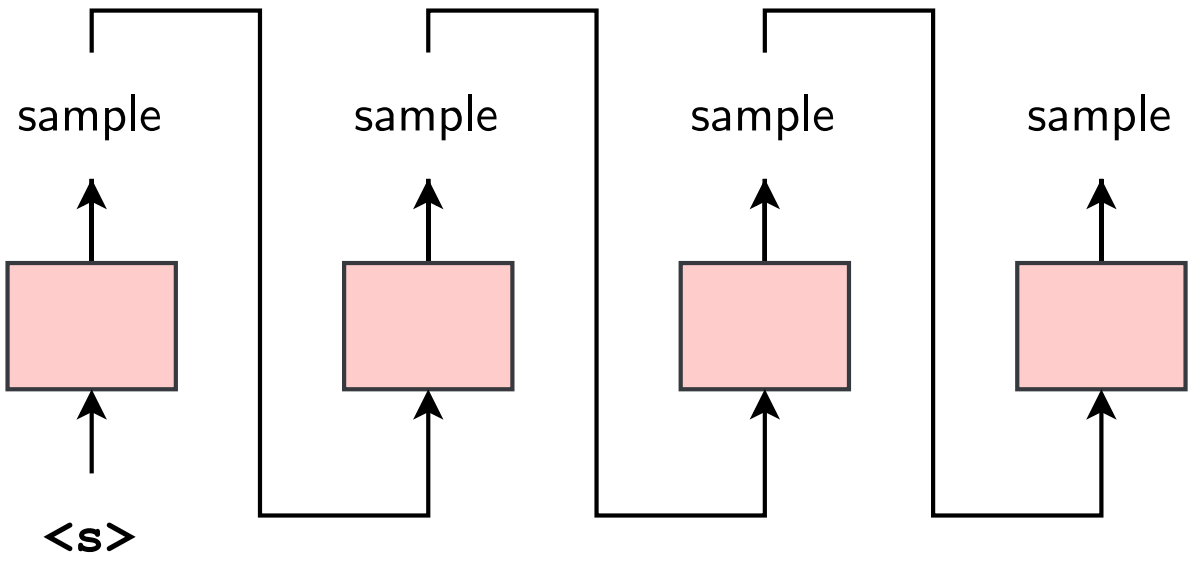
1. Autoregressive generation

2. Searching for the most probable sequence

3. Random sampling

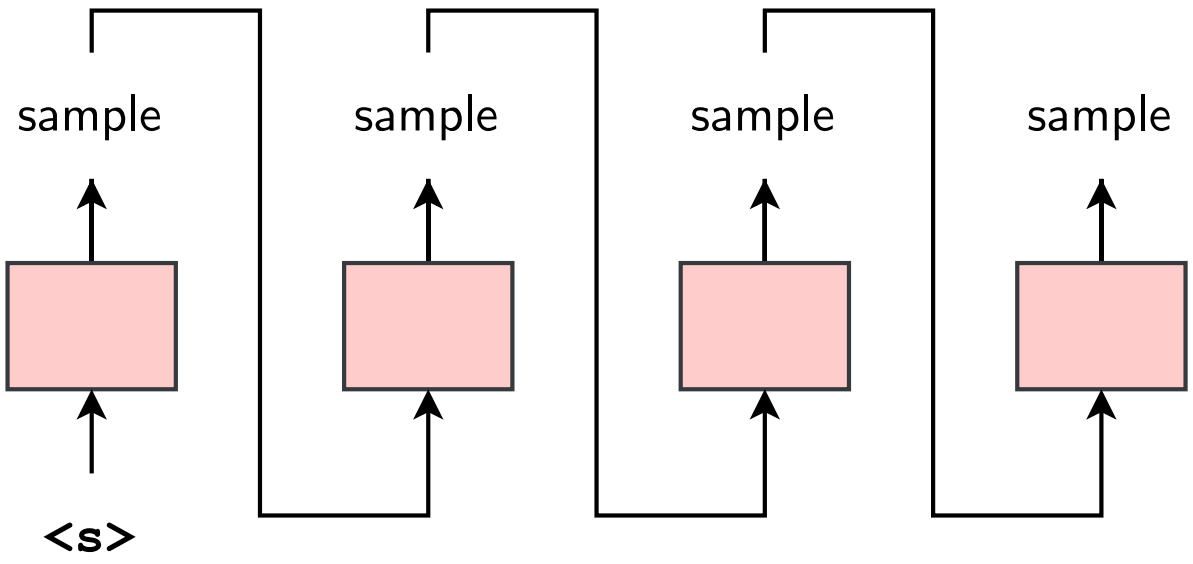
4. References

Random sampling



How to **sample**?

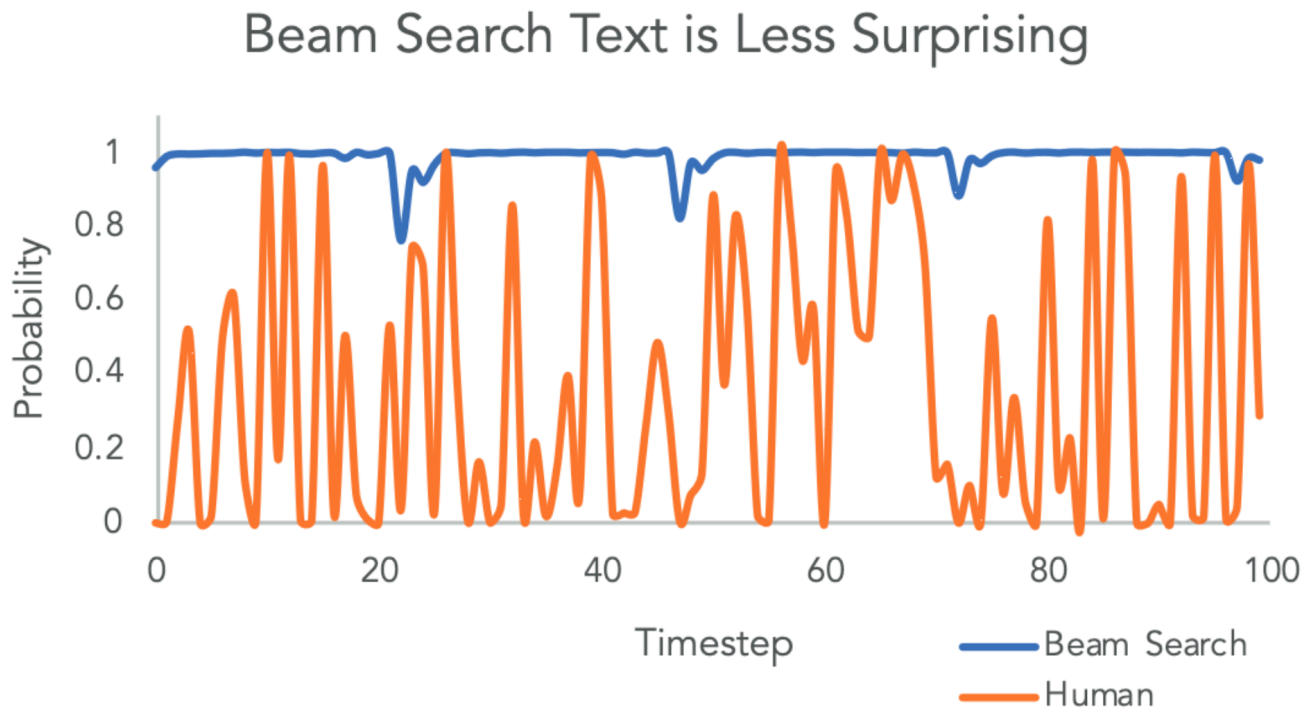
Random sampling



How to **sample**? And why?

Random sampling

"Natural language does not maximaze probability" (Holtzman et al., 2020)



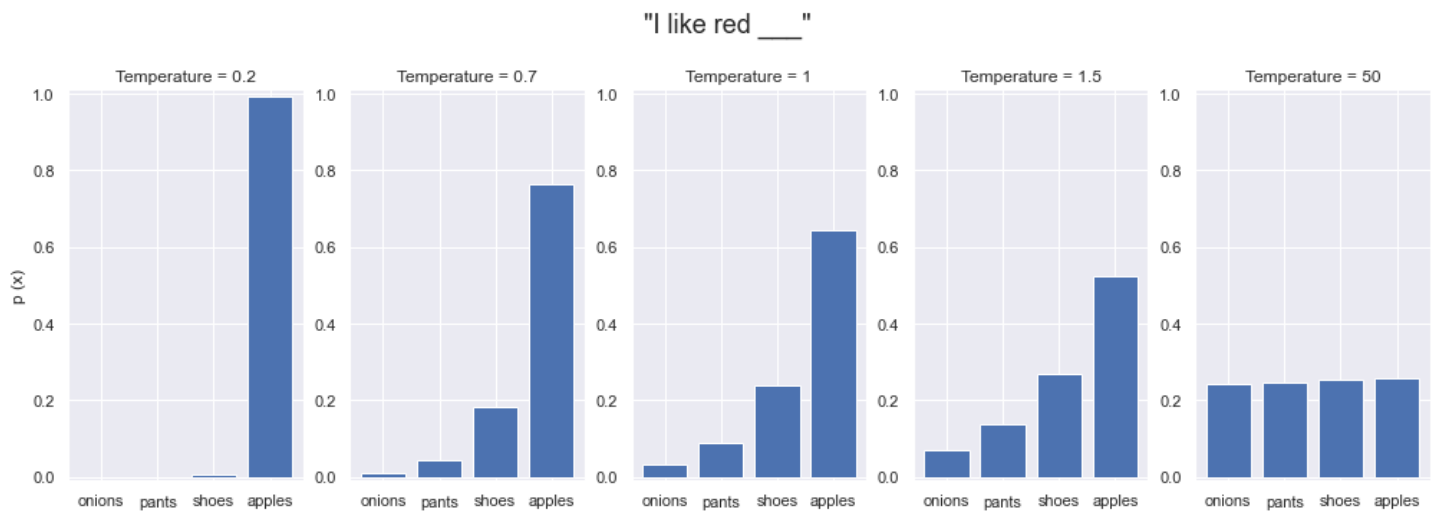
Sampling vs. beam search

- The outputs from beam/greedy search can sound stiff, generic, repetitive or simply boring
- That's exactly what we want from machine translation (most of the time) – to exactly translate the source text without any creative variation
- But it's not what we expect from chatbots, for example
- We want them to get "creative" and use less probable tokens from time to time
- Also, a practical issue of beam/greedy search is that it leads to outputs with infinitely repeated n-grams

Sampling

- Randomly picks a token from the conditional distribution $P(w_i|w_{0:i-1})$
- We can control the amount of "randomness" (entropy) with the temperature hyperparameter (T)

$$p_i = \frac{e^{x_i/T}}{\sum_j e^{x_j/T}}$$



https://medium.com/@imisri1/how-to-set-sampling-temperature-for-gpt-models-762887df1fac

Sampling

- Randomly pick a token from the conditional distribution $P(w_i|w_{0:i-1})$
- We can control the amount of "randomness" (entropy) with the temperature hyperparameter (T)

$$p_i = \frac{e^{x_i/T}}{\sum_j e^{x_j/T}}$$

- NB: when $T \to 0$, we get greedy search; when $T \to \infty$, we sample uniformly
- When unlucky, this method can sample even the least probable token, how to prevent that?

Random sampling

Context: In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

Beam Search, *b*=32:
"The study, published in the Proceedings of the National Academy of Sciences of the United States of America (PNAS), was conducted by researchers from the Universidad Nacional Autónoma de México (UNAM) and the Universidad Nacional Autónoma de México (UNAM/Universidad Nacional Autónoma de México/Universidad Nacional Autónoma de México/Universidad Nacional Autónoma de México/Universidad Nacional Autónoma de ..."

Pure Sampling:
They were cattle called Bolivian Cavalleros; they live in a remote desert uninterrupted by town, and they speak huge, beautiful, paradisiacal Bolivian linguistic thing. They say, 'Lunch, marge.' They don't tell what the lunch is," director Professor Chuperas Omwell told Sky News. "They've only been talking to scientists, like we're being interviewed by TV reporters. We don't even stick around to be interviewed by TV reporters. Maybe that's how they figured out that they're cosplaying as the Bolivian Cavalleros."

Figure 1: Even with substantial human context and the powerful GPT-2 Large language model, Beam Search (size 32) leads to degenerate repetition (highlighted in blue) while pure sampling leads to incoherent gibberish (highlighted in red). When $b \geq 64$, both GPT-2 Large and XL (774M and 1542M parameters, respectively) prefer to stop generating immediately after the given context.

(Holtzman et al., 2020)

Random sampling



Top-k sampling (Fan et al., 2018)

- We select only the k most probable tokens and sample from those
- Another way to control precision vs. creativity
- Doesn't take into account the probability distribution – one very likely token vs. hundreds of equally likely tokens

Top-p (nucleus) sampling (Holtzman et al., 2020)

- We sample from the smallest set of tokens whose sum of probabilities exceeds p
- The set of sampled tokens dynamically changes according to the probability distribution

Conclusion

- In practice, it's best to combine top-k with top-p sampling and with temperature adjustment
- The quality of generated text is highly dependent on these hyper-parameters
- Degenerate outputs are still an issue and better decoding strategies are an active research area

Table of contents



1. Autoregressive generation

2. Searching for the most probable sequence

3. Random sampling

4. References

References I



Fan, A., Lewis, M., and Dauphin, Y. (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.

Holtzman, A., Buys, J., Du, L., Forbes, M., and Choi, Y. (2020). The curious case of neural text degeneration. In *International Conference on Learning Representations*.