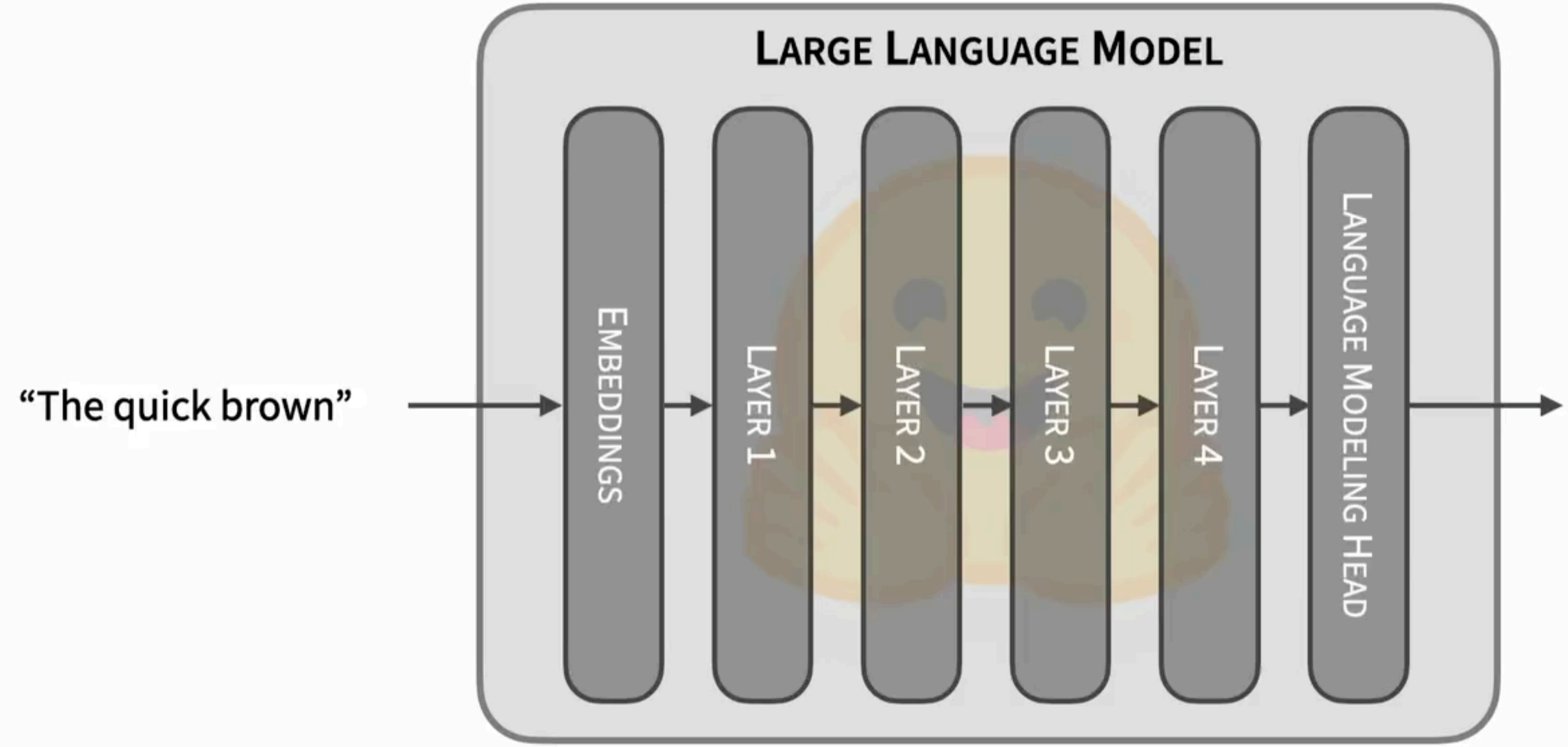
Decoding

NLP: Fall 2023

Anoop Sarkar

Causal Language Models



https://huggingface.co/docs/transformers/llm_tutorial

Causal Language Models

"Autoregressive generation iteratively selects the next token from a probability distribution to generate text"

Causal LMs: Common Pitfalls

- **Generated output is too short/long**: LM may require further tuning, also asking for more tokens can help
- **Incorrect generation mode**: greedy decoding or sampling? Which is better depends on your task
- **Wrong padding side**: you may need to pad the prompt text on the left to ensure that the input is the same size as the training phase of the LM.
- **Wrong prompt**: this is tricky and has produced a whole industry of "prompt engineering"

https://huggingface.co/docs/transformers/llm_tutorial

c.f. for code samples

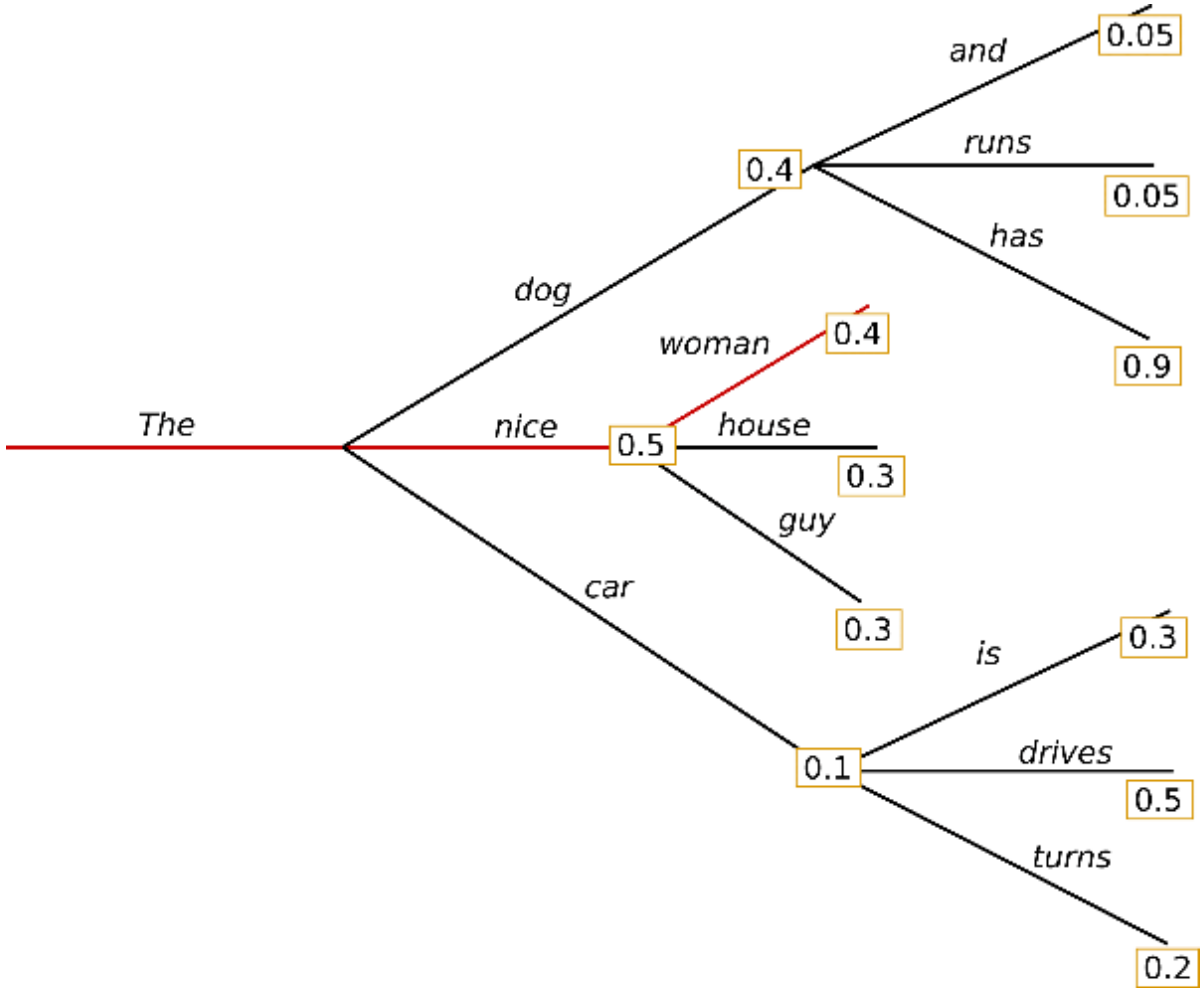
Decoding methods

https://huggingface.co/blog/how-to-generate

$$P(w_{1:T}|W_0) = \prod_{t=1}^{T} P(w_t|w_{1:t-1}, W_0) \text{ ,with } w_{1:0} = \emptyset,$$

- W_0 is the initial context word sequence (aka the "prompt")
- The length T of the word sequence is determined on-the-fly
- T is determined by the generation of the end-of-sentence EOS also known as the <|endoftext|> token
- The EOS token is produced like the other tokens from $P(w_t \mid w_{1:t-1}, W_0)$

Greedy Decoding



("The","nice","woman")
having an overall
probability of
$0.5 \times 0.4 = 0.2$

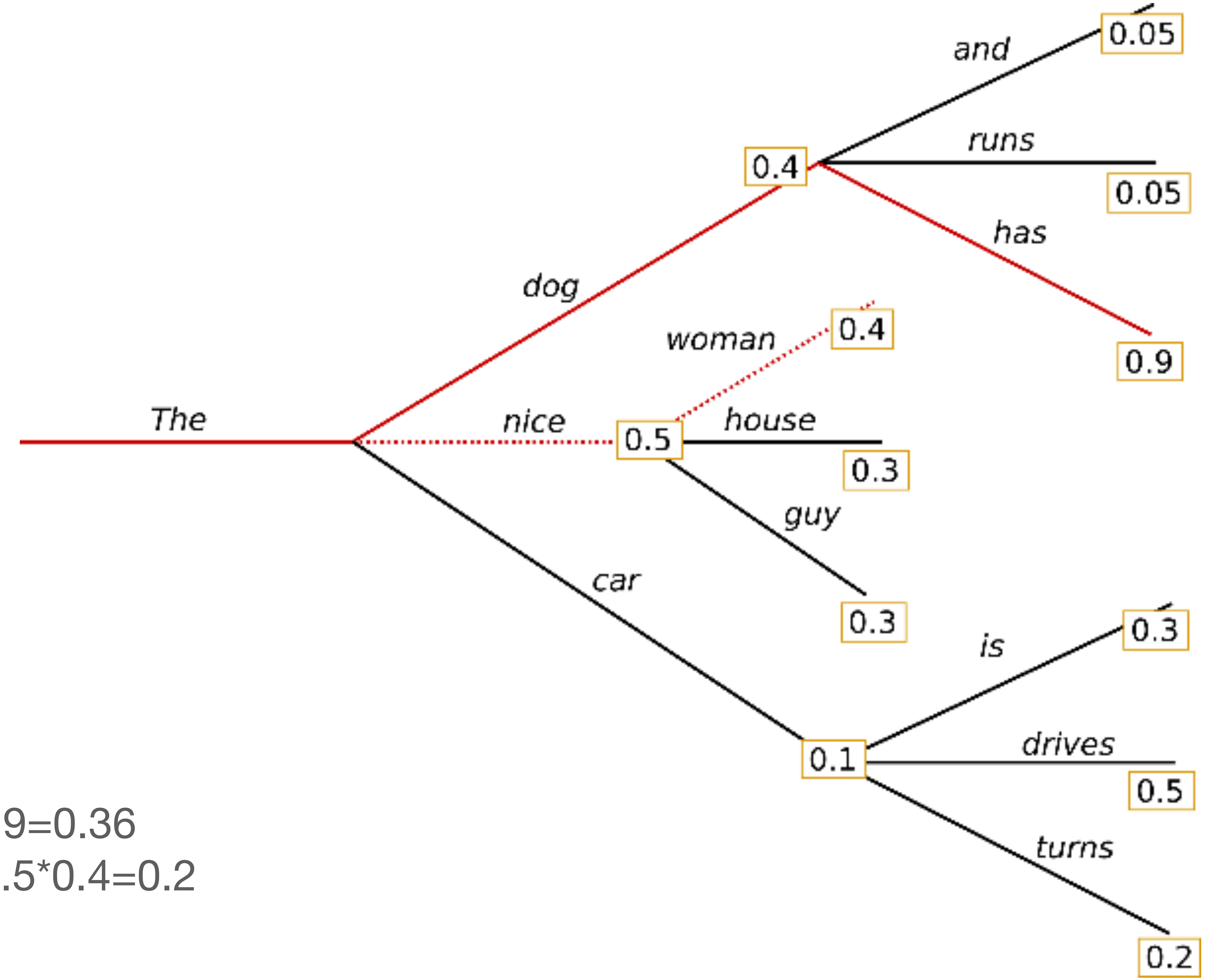
Beam Search

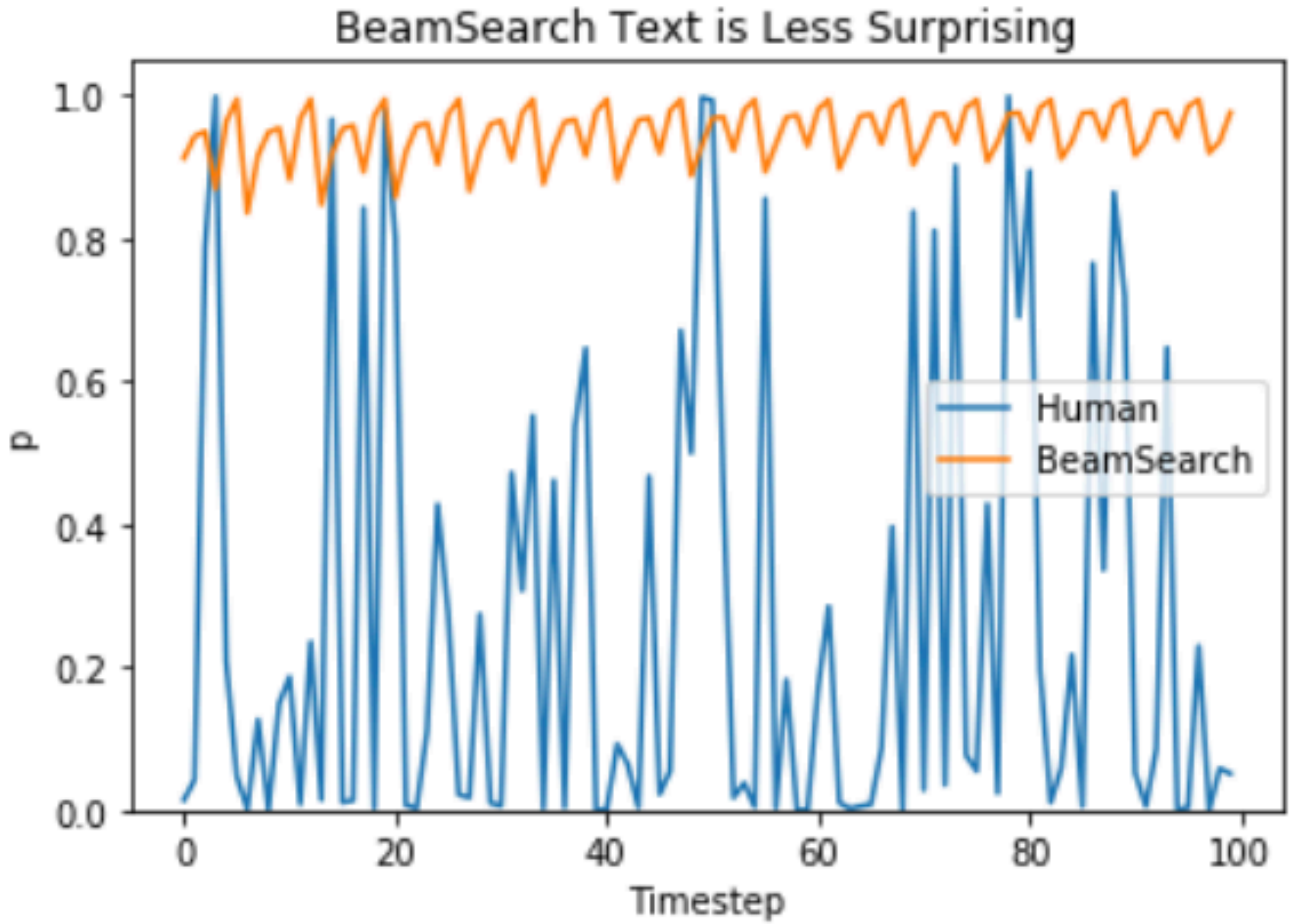
Let us assume a beam size of 2

Keep the 2 best outcomes at each time step

In this example:
("The", "nice") 0.5
("The", "dog") 0.4

Next time step:
("The", "dog", "has") 0.5*0.9=0.36
("The", "nice", "woman") 0.5*0.4=0.2

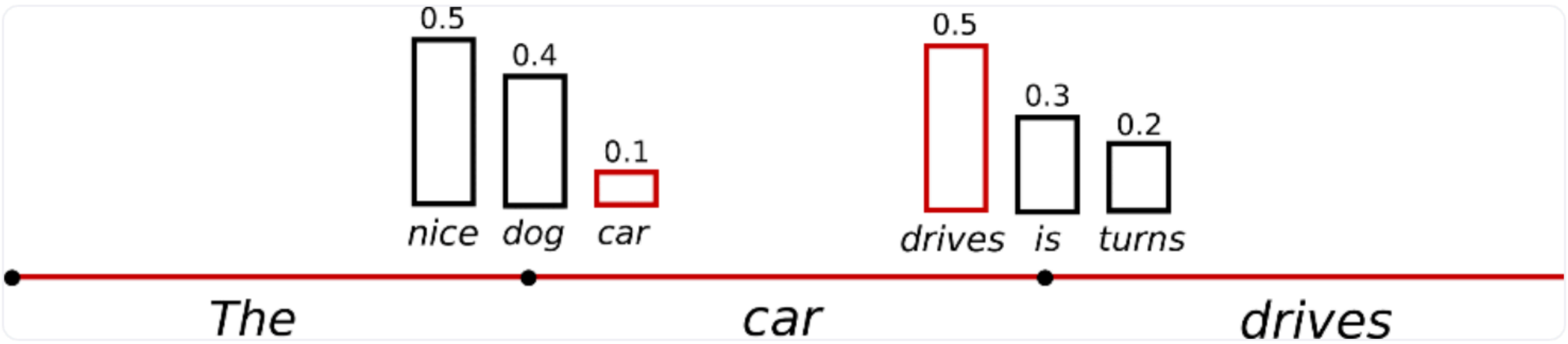




Ari Holtzman et al. (2019) plot probability that a model gives versus an estimate of the probability that a human would give. As humans we want generated text to surprise us and not be boring/predictable (depends on the task).

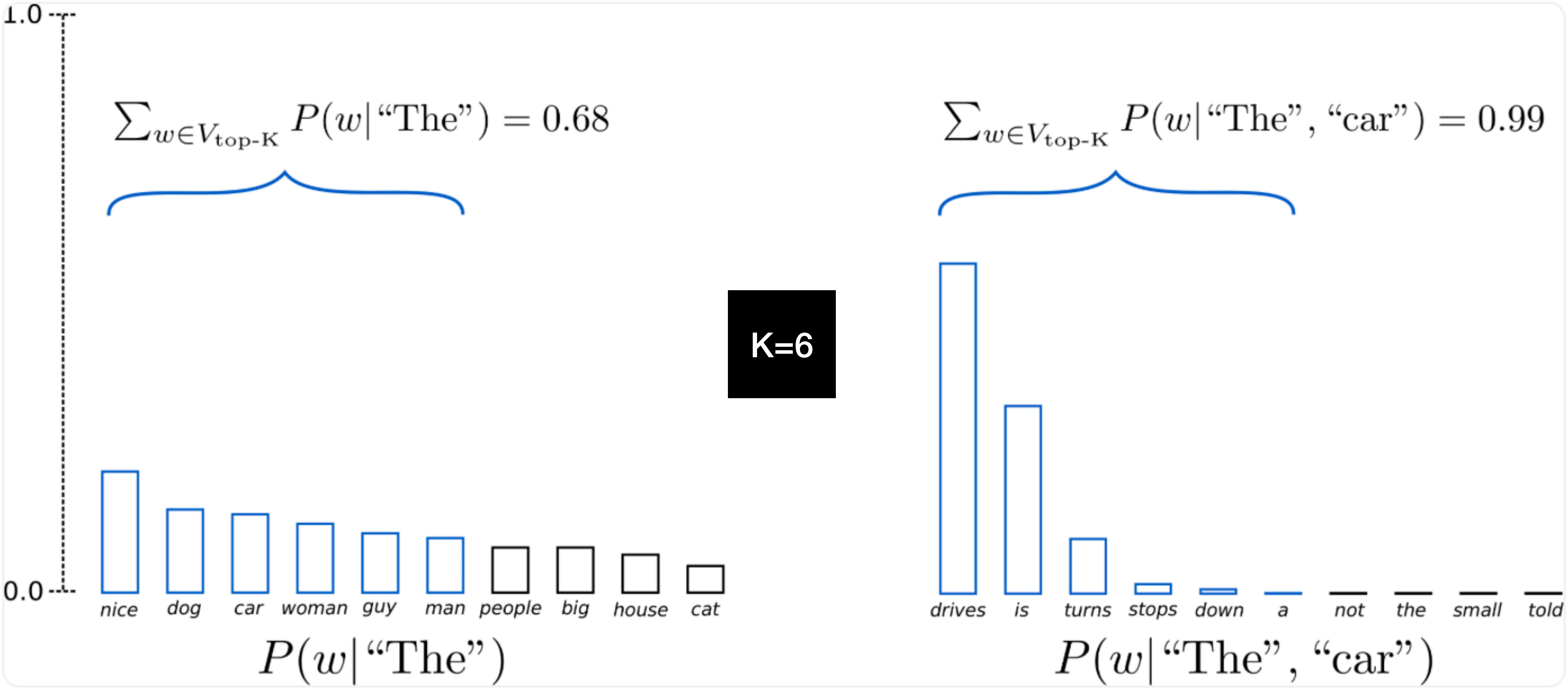
Sampling

- Sampling is represented by the operator $\sim$
- We pick the next word $w_t \sim P(w \mid w_{1:t-1}) = \dfrac{exp(logits(w \mid w_{1:t-1}))}{\sum_{w'} exp(logits(w' \mid w_{1:t-1}))}$
- Generation is no longer *deterministic.*
- Sampling can generate gibberish. Solution: use temperature $\dfrac{exp(logits(w \mid w_{1:t-1})/T)}{\sum_{w'} exp(logits(w' \mid w_{1:t-1})/T)}$



Top-k Sampling

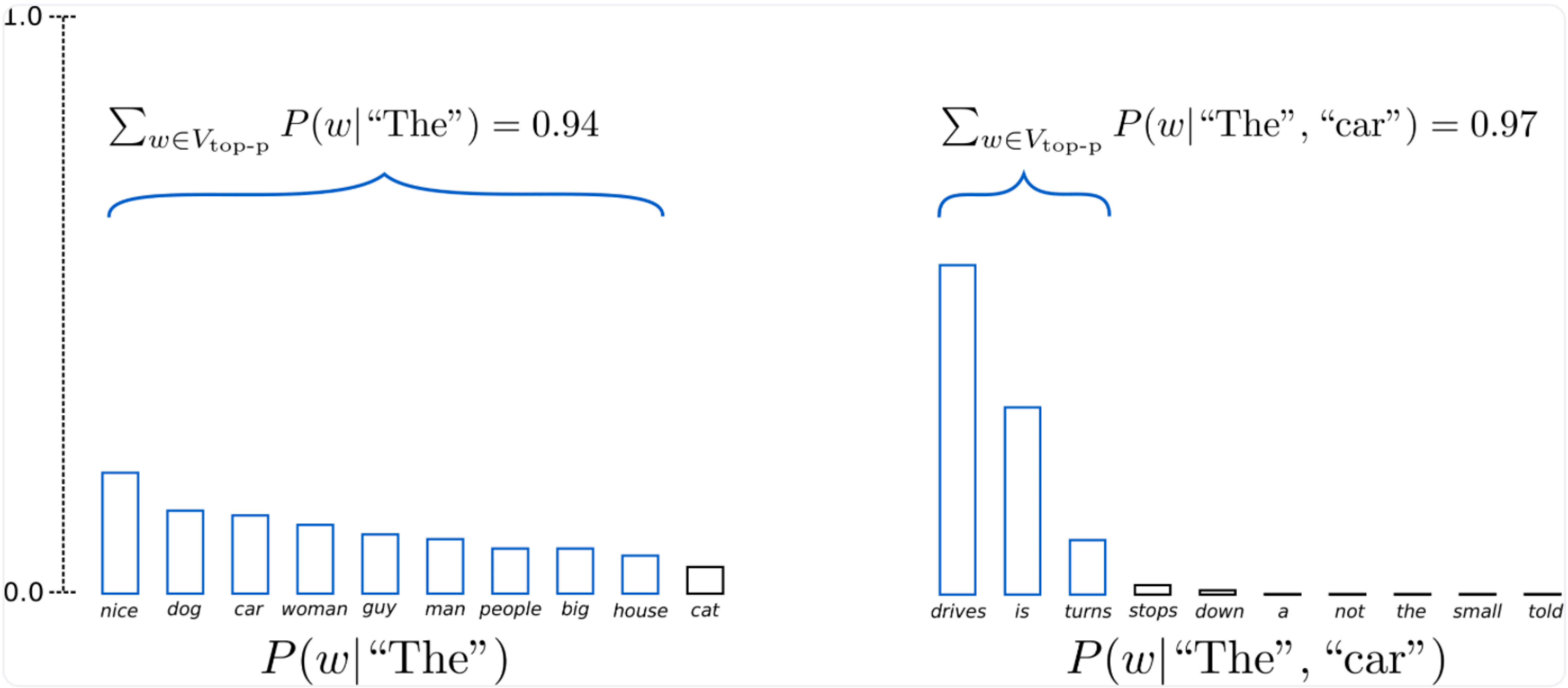
- K most likely next words are filtered and we re-normalize over the K words
- GPT2 showed that this worked better than beam search



Top-p Nucleus Sampling

- Choose the smallest set of words whose cumulative probability exceeds a threshold probability p. The probability mass is redistributed among this set of words.

- The size of the set being sampled from grows and shrinks depending on the probability distribution.



Other problems

- **Unreachable subword problem**: there are some subwords for which under no circumstances is it possible to produce a subword (given any context).
- **Mode collapse**: tuning the LM might cause the model parameters to reach a state where Greedy and Sampling based generation produce the same output.
- **Softmax over very large vocabulary sizes**: Vocabulary sizes have reduced since subword segmentation has become the standard way to set up the vocabulary for LMs; However for very large vocabulary sizes, the compute efficiency for softmax might need careful consideration, e.g. use hierarchical softmax.