Decoding Strategies

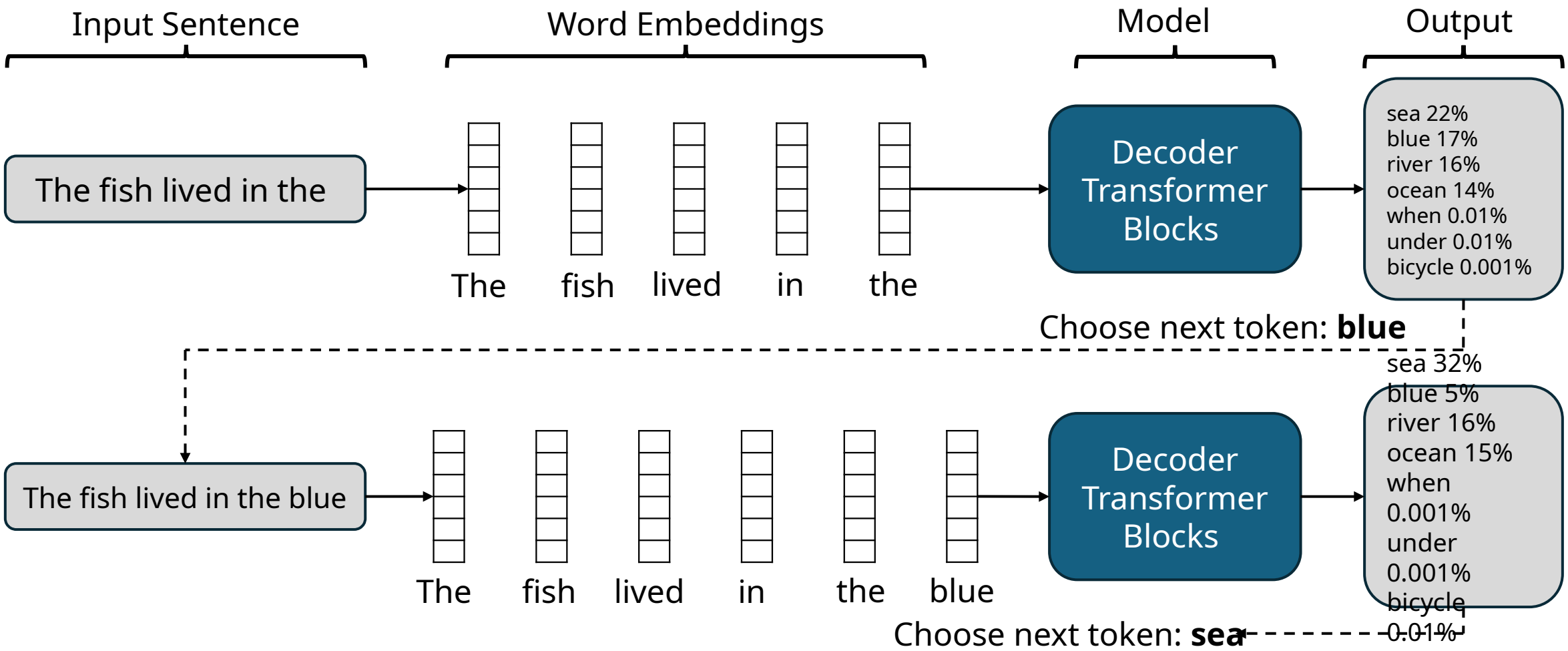
CS XXX: Introduction to Large Language Models

Contents

- Generating Text
- Greedy Decoding
- Beam Search Decoding
- Top k Sampling
- Top p (Nucleus) Sampling
- Generation Parameters

Generating Text

- Generate next token based on the current context



Generating Text

- More concretely, Once language model trained we generate the text in the following way
  - Sample a word in the output from the softmax distribution that results from using the beginning of sentence marker, <s>, as the first input.
  - Use the word embedding for that first word as the input to the network at the next time step, and then sample the next word in the same fashion.
  - Continue generating until the end of sentence marker, </s>, is sampled or a fixed length limit is reached.

- The goal of sequence generation is to produce a sequence of tokens that maximizes the overall probability of the complete sequence given the context.

Generating Text

- The goal of sequence generation is to produce a sequence of tokens that maximizes the overall probability of the complete sequence given the context.
- Consider the following distribution given the context “<s>The mouse”:

$$\mathrm{P}(<\mathrm{s}>, \mathrm{The}, \mathrm{mouse}, \mathrm{ate}, \mathrm{the}, \mathrm{cheese}, </\mathrm{s}>) = 0.02,$$

$$\mathrm{P}(<\mathrm{s}>, \mathrm{The}, \mathrm{cheese}, \mathrm{ate}, \mathrm{cheese}, </\mathrm{s}>) = 0.01,$$

$$\mathrm{P}(<\mathrm{s}>, \mathrm{mouse}, \mathrm{the}, \mathrm{the}, \mathrm{cheese}, \mathrm{ate}, </\mathrm{s}>) = 0.0001.$$

$$\vdots$$

then the generation of the model should result in the sequence with the highest probability

<s>The mouse ate the cheese</s>

Generating Text

- How should the model pick the next word at each time step?
- Consider the distribution after the word “ate” was generated

$$P(\text{cheese} \mid \text{<s>}, \text{The}, \text{mouse}, \text{ate}) = 0.30$$

$$P(\text{icecream} \mid \text{<s>}, \text{The}, \text{mouse}, \text{ate}) = 0.18$$

$$P(\text{the} \mid \text{<s>}, \text{The}, \text{mouse}, \text{ate}) = 0.25$$

$$P(\text{ate} \mid \text{<s>}, \text{The}, \text{mouse}, \text{ate}) = 0.12$$

$$P(\text{mouse} \mid \text{<s>}, \text{The}, \text{mouse}, \text{ate}) = 0.01$$

$$P(\text{The} \mid \text{<s>}, \text{The}, \text{mouse}, \text{ate}) = 0.05$$

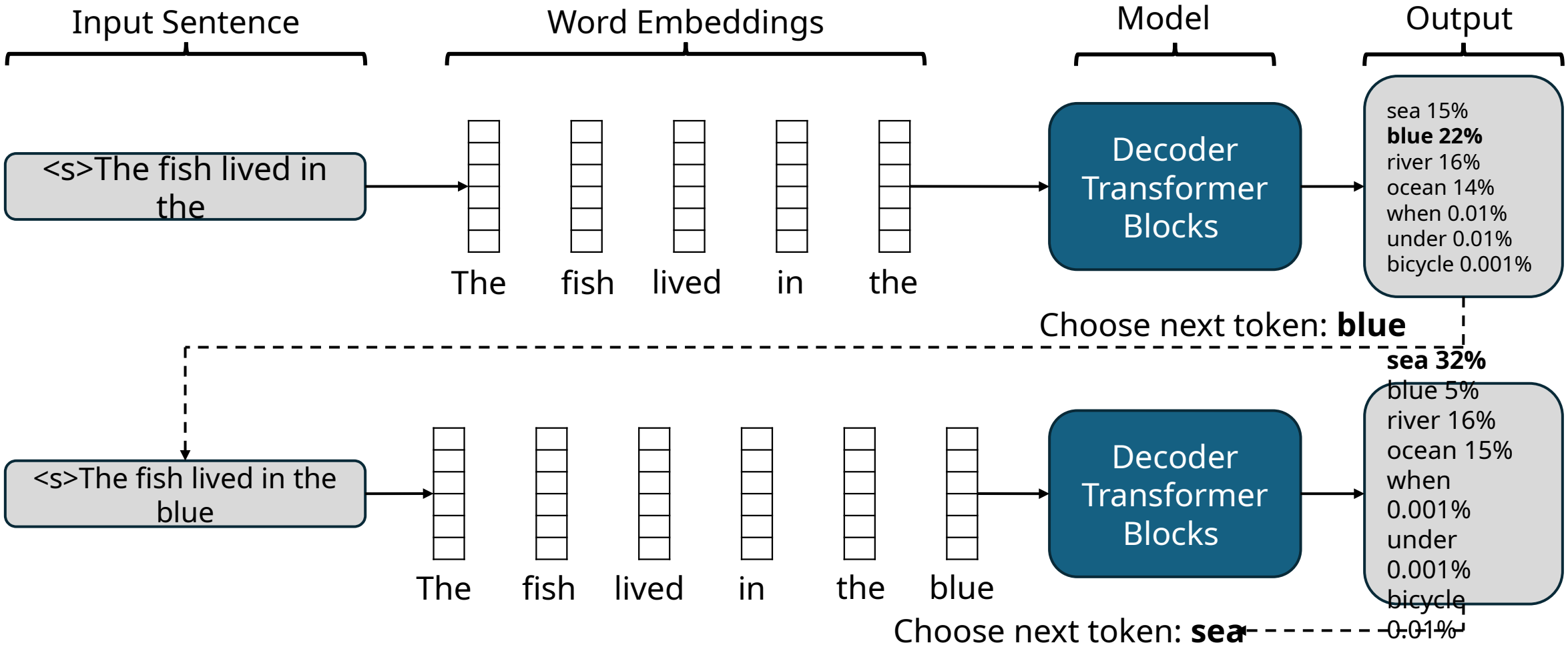
The next word generated should be “the” instead of “cheese” even though “cheese has a higher probability than “the” since predicting “the” will lead to the model generating the highest probability sequence “<s>The mouse ate the cheese</s>”

Decoding Strategies

- Decoding strategies in Large Language Models (LLMs) refer to different methods used to generate text from a model's probability distribution over the vocabulary.
- These strategies determine how the next word (or token) is selected and can significantly impact the quality, coherence, and diversity of generated text.

Greedy Decoding

- **Greedy Decoding**: Choose the token with the greatest probability at each time step.



Greedy Decoding

- **Greedy Decoding**: Choose the token with the greatest probability at each time step.

- We terminate generation once the end of sequence token (</s>) is predicted or when the number of tokens predicted is equal to some max length.

Greedy Decoding

- **Greedy Decoding**: Choose the token with the greatest probability at each time step.

- We terminate generation once the end of sequence token (</s>) is predicted or when the number of tokens predicted is equal to some max length.

- **Problem:** Choosing the token with the highest probability may be locally optimal but not globally since optimal sequence generation requires considering the entire sequence probability, not just picking the most likely word at each step. Since greedy decoding does not explore multiple options, it often produces repetitive text. Greedy decoding always produces the same output for a given input (lack of diversity).

Beam Search

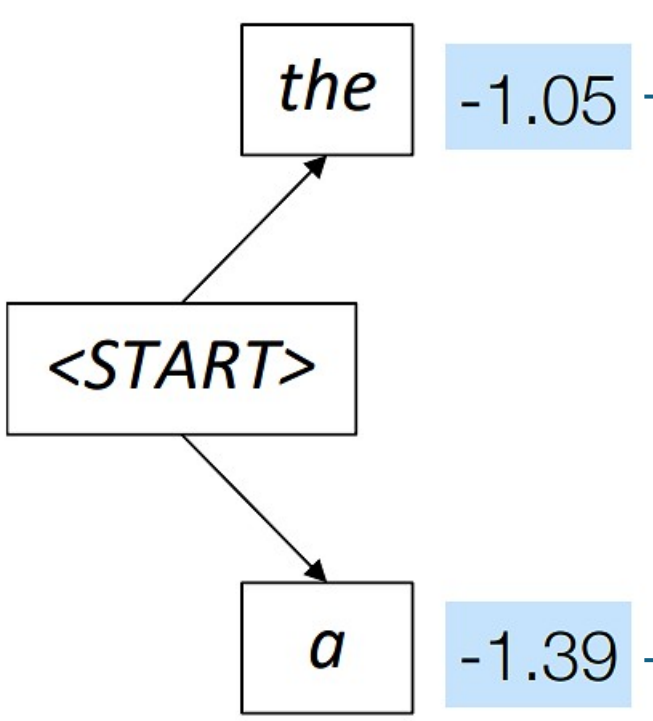
- Keep track of n most probable partial translations at each decoder step instead of just one!
- the beam size n is usually 5-10

Beam Search

- Beam Size = 2

Beam Search

- Beam Size = 2
- Consider only the 2 (beam size) most likely words at the first time step



$P(\text{<START>}, \text{the}) = \log(P(\text{the}|\text{<START>}) \times P(\text{<START>})) = \log(P(\text{the}|\text{<START>})) + \log(P(\text{<START>}))$

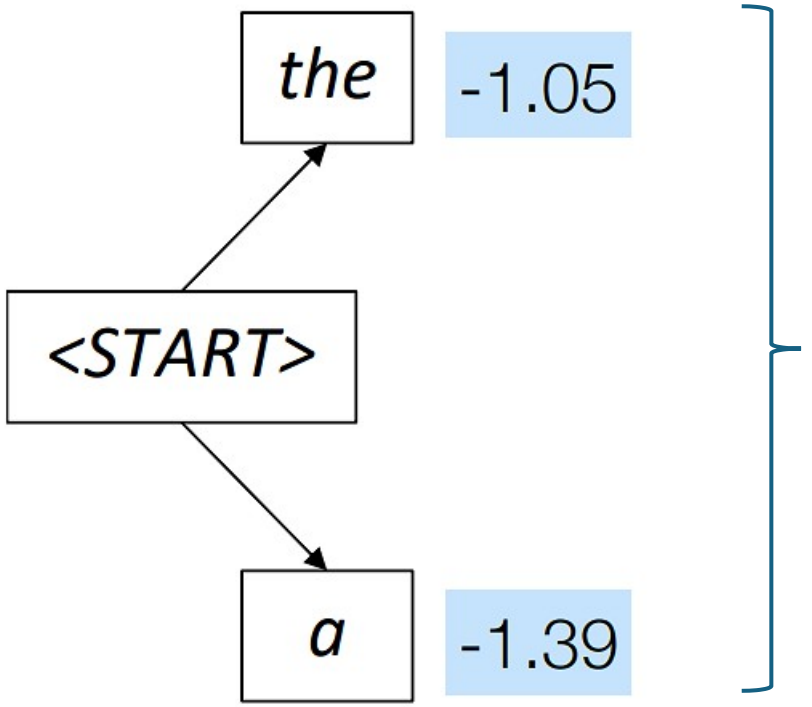
log probabilities: Since probabilities are small numbers (between 0 and 1), multiplying them can lead to numerical underflow. To avoid this, we typically use log probabilities, converting the product into a sum

$P(\text{<START>}, \text{a}) = \log(P(\text{a}|\text{<START>}) \times P(\text{<START>})) = \log(P(\text{a}|\text{<START>})) + \log(P(\text{<START>}))$

Note that $P(\text{<START>}) = 1, \ \log(1) = 0$

Beam Search

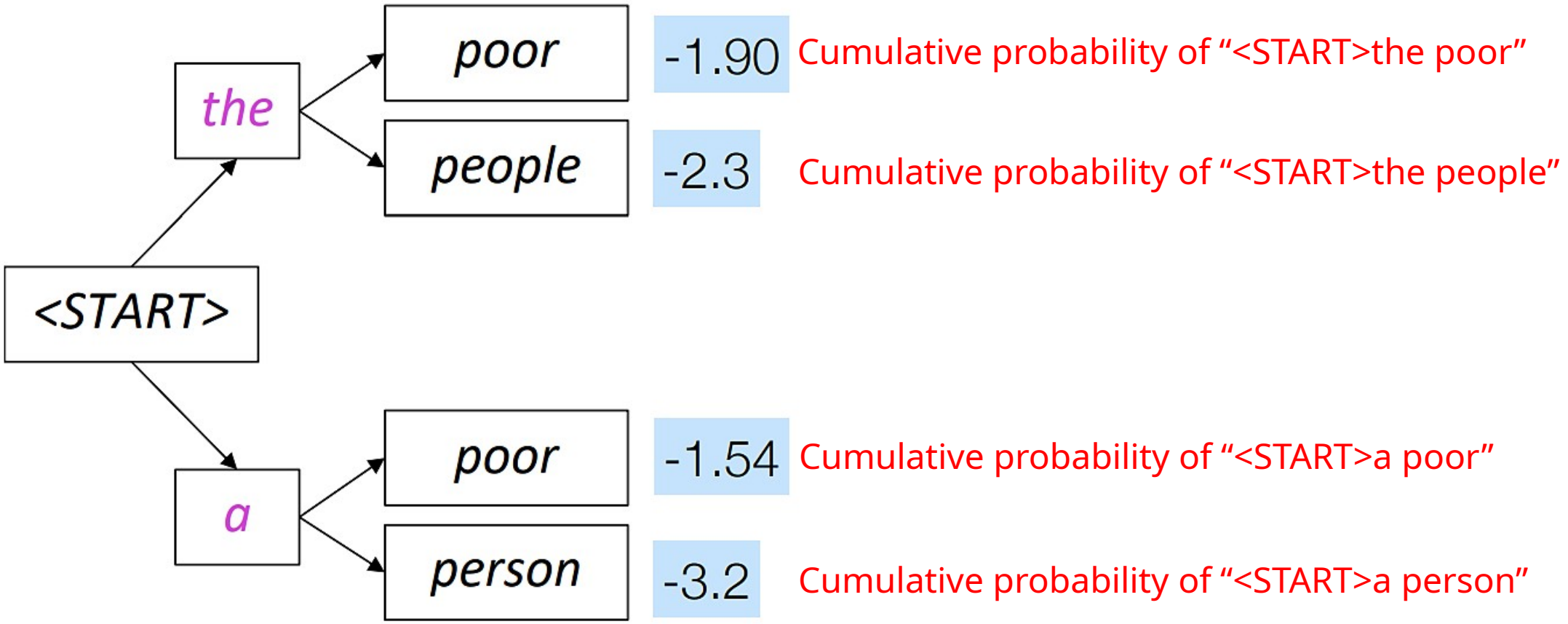
- Beam Size = 2
- Consider only the 2 (beam size) most likely words at the first time step



Expand the current branches (2) by considering the next 2 most likely tokens for each branch

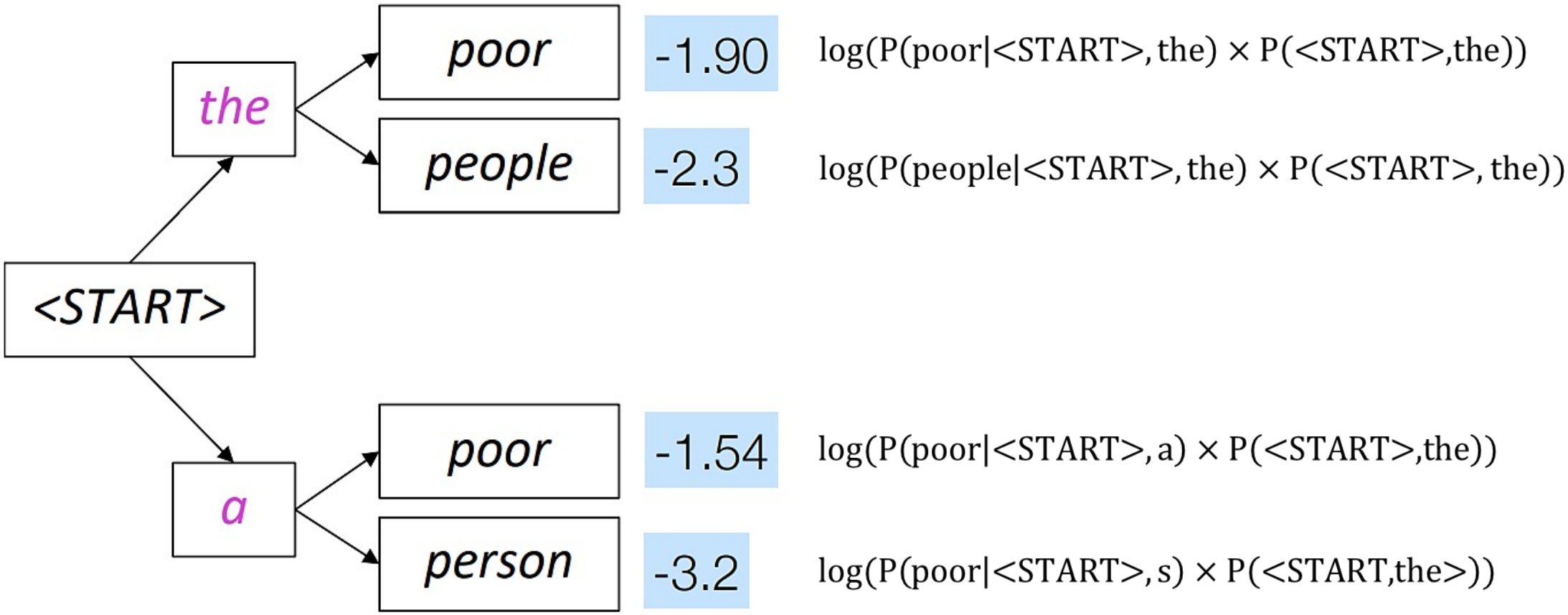
Beam Search

- Beam Size = 2



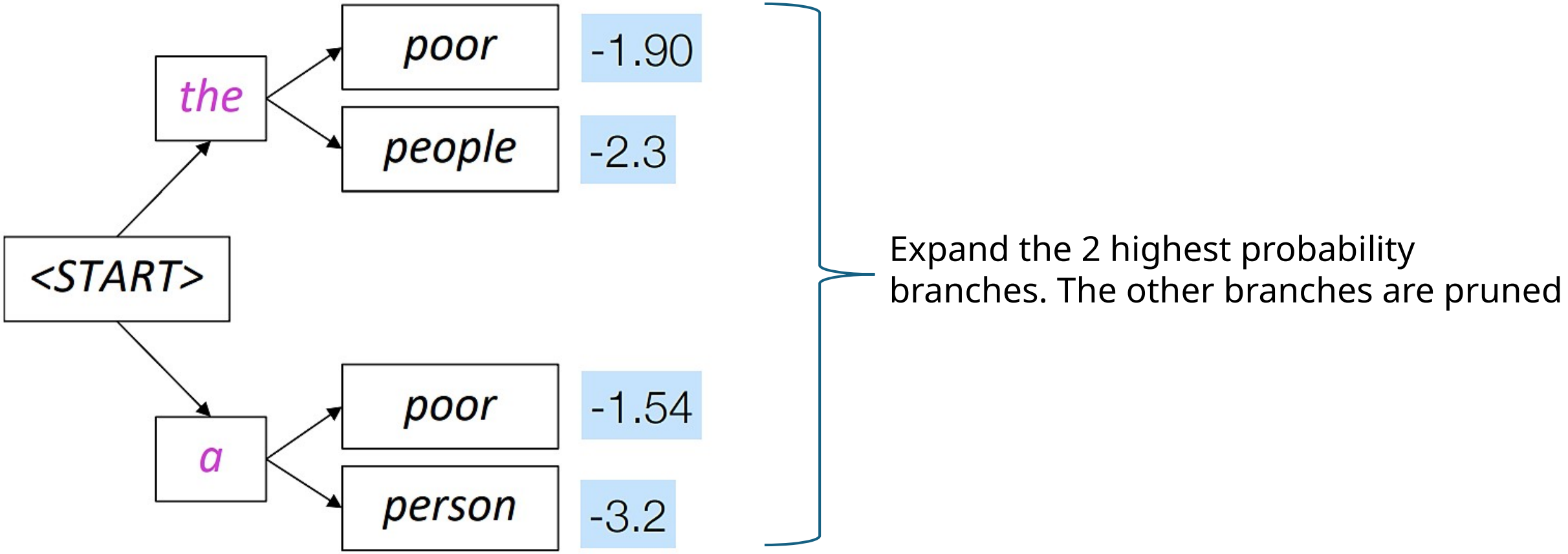
Beam Search

- Beam Size = 2



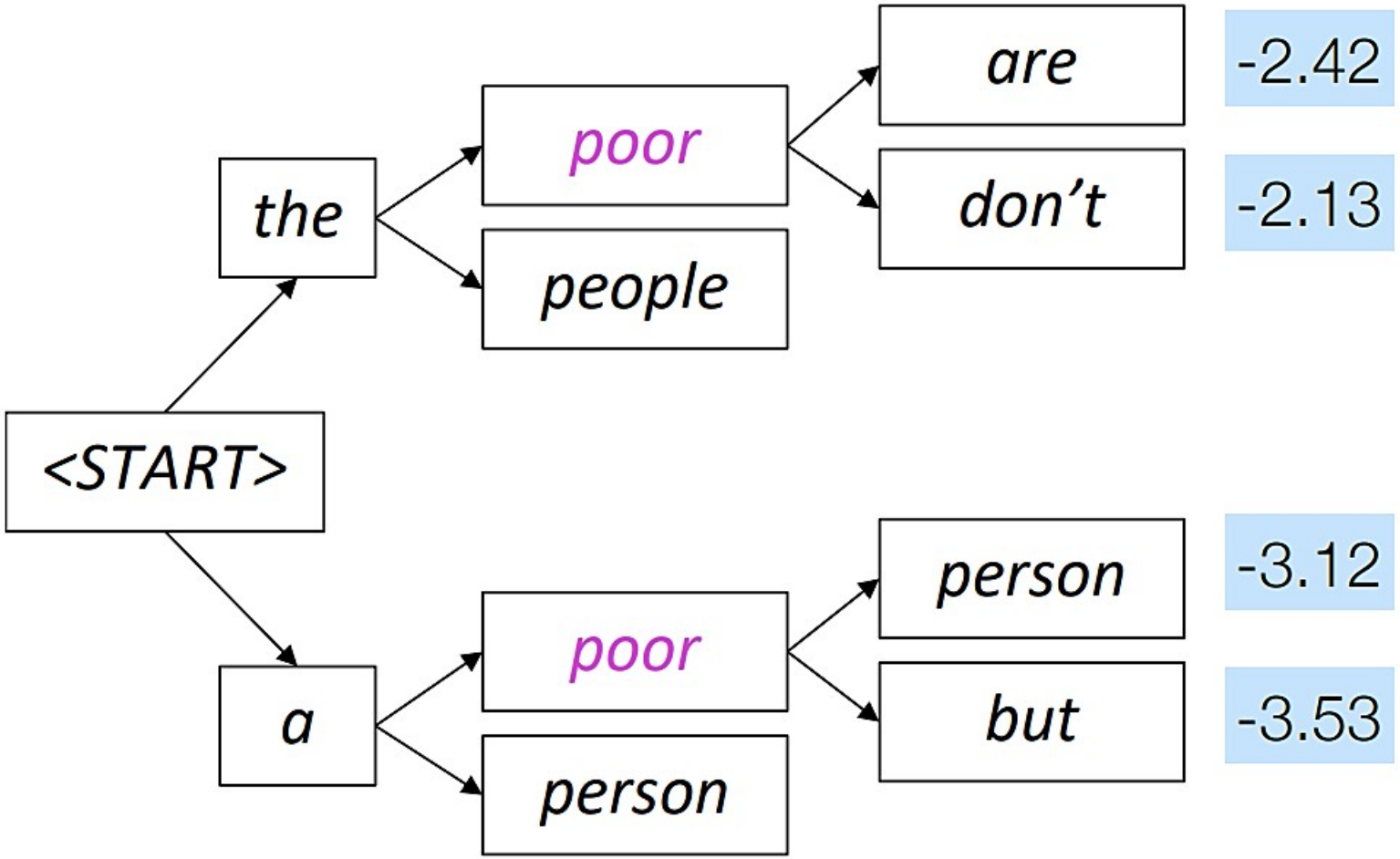
Beam Search

- Beam Size = 2

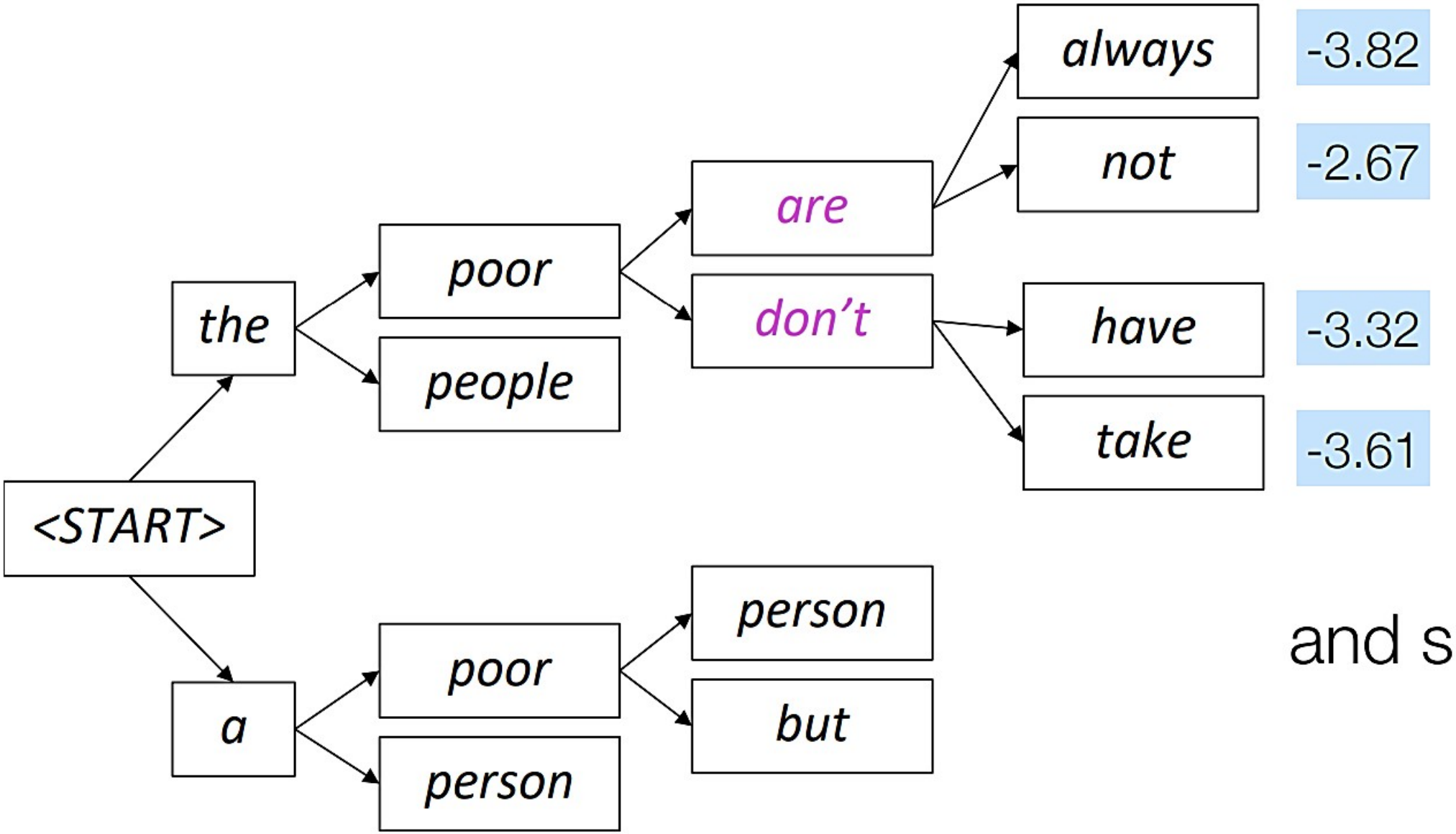


Beam Search

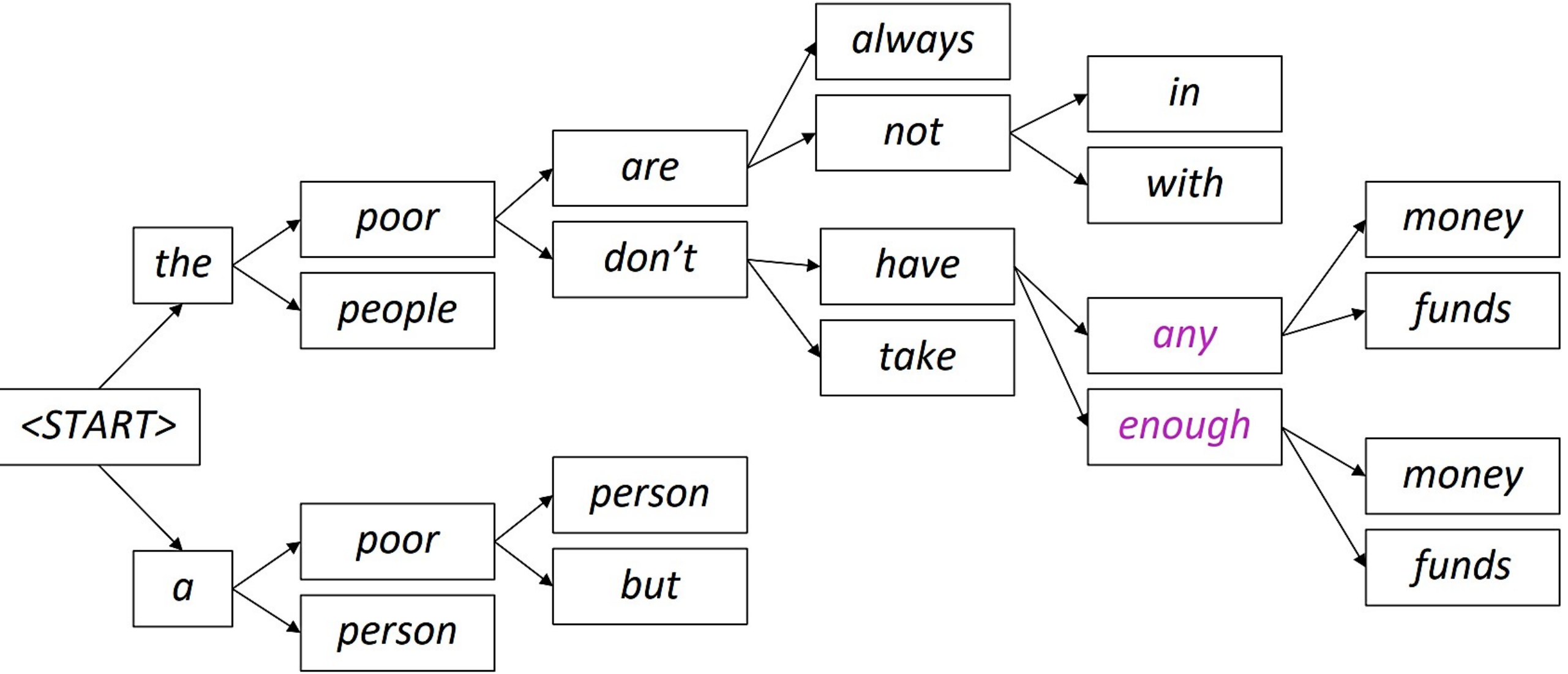
- Beam Size = 2



Beam Search


<START>
the
a
poor
people
poor
person
are
don't
poor
but
always
not
have
take
person
-3.82
-2.67
-3.32
-3.61
and so on…

Beam Search

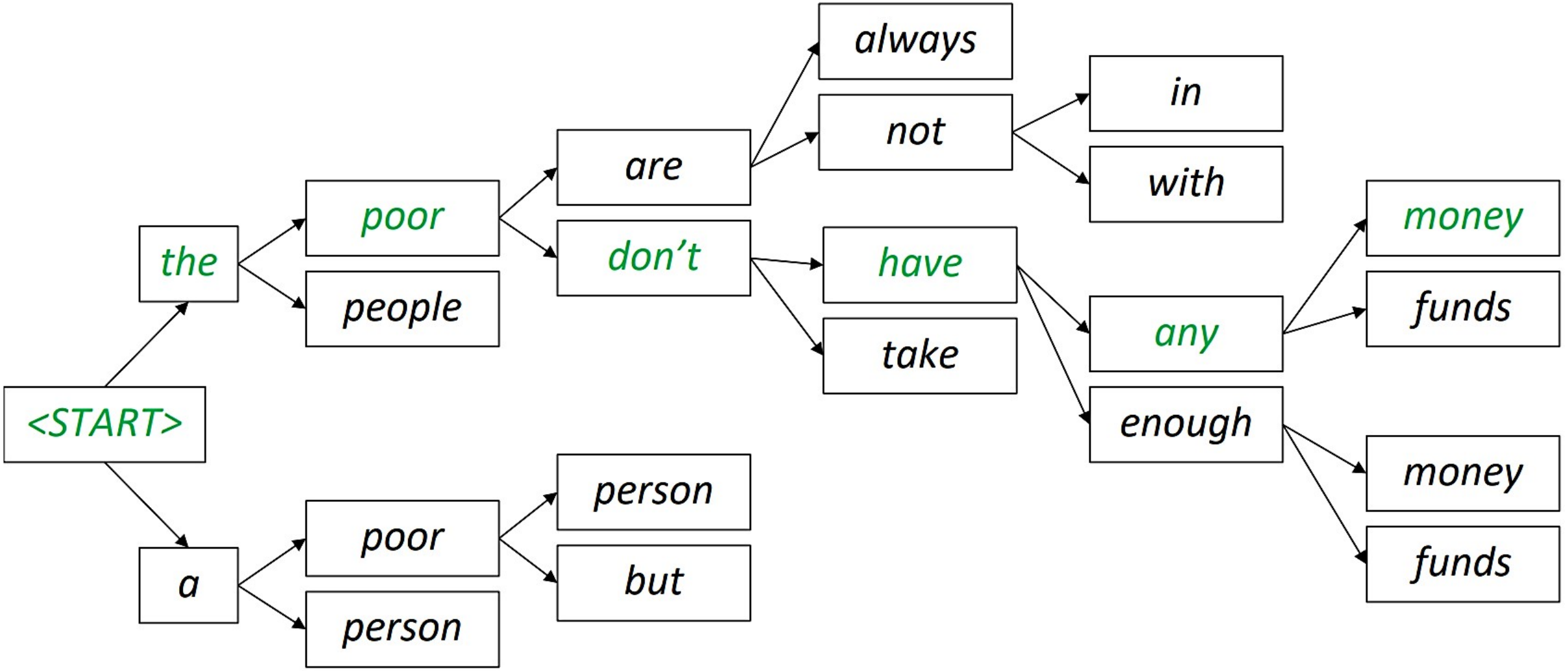

<START>
the
a
poor
people
poor
person
are
don't
person
but
always
not
have
take
in
with
any
enough
money
funds
money
funds

Beam Search

- Generation stops when all N (beam size) branches have either predicted the end-of-sequence (EOS e.g. </s>) token or reached the maximum allowed length.
- Once generation terminates, the sequence with the highest cumulative probability is selected as the final output.

Beam Search

- Once generation terminates, the sequence with the highest cumulative probability is selected as the final output.



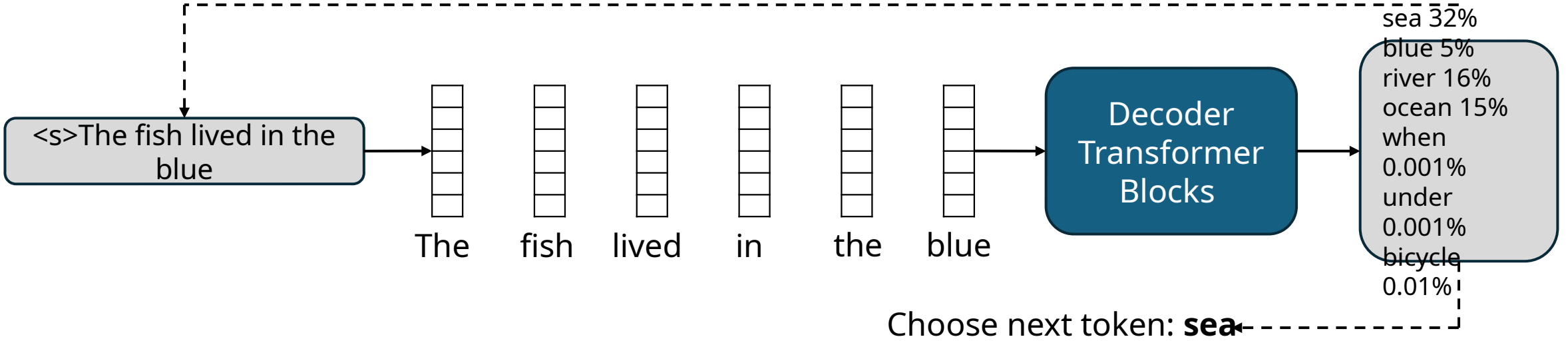
Beam Search

- Effect of beam size N
  - Small N means consider less hypotheses. The resulting generated sequence is likely to be ungrammatical, unnatural, nonsensical, or incorrect.
  - Large N means consider more hypotheses. The resulting generated sequence is likely to be more grammatical and natural, however larger N is more computationally expensive. A larger N may also make the generation more generic and less relevant.

Top k Sampling

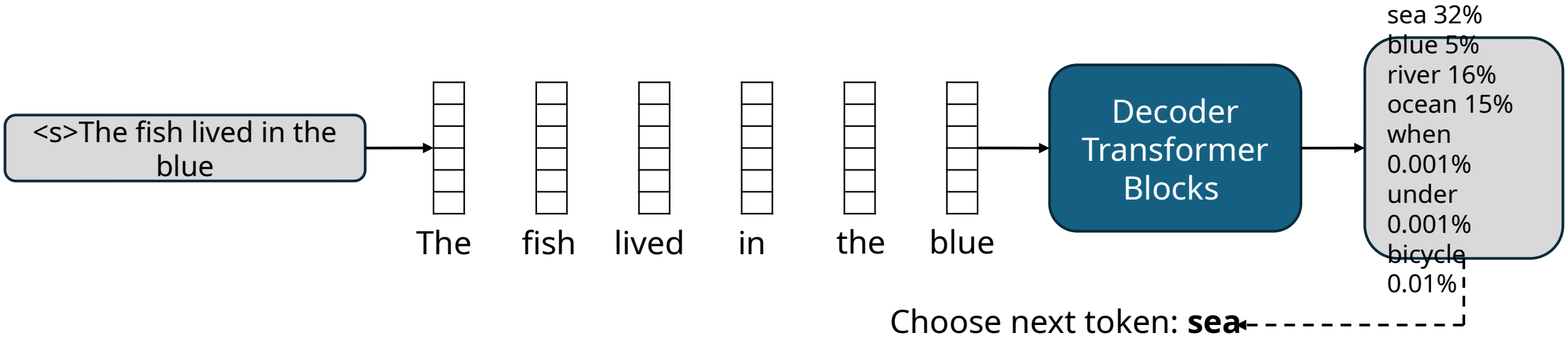
- On each step t, randomly sample from P_t,restricted to just the top-k most probable words

- Consider the probability distribution over the vocabulary in the following example



Top K Sampling

- Consider the probability distribution over the vocabulary in the following example



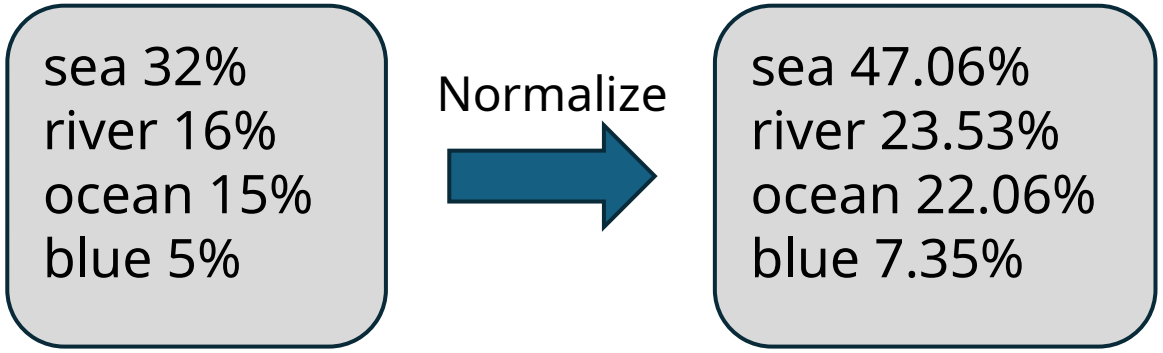
- If k = 4 the distribution is truncated to

sea 32%
river 16%
ocean 15%
blue 5%

Note that this truncation is done at each time step

Top K Sampling

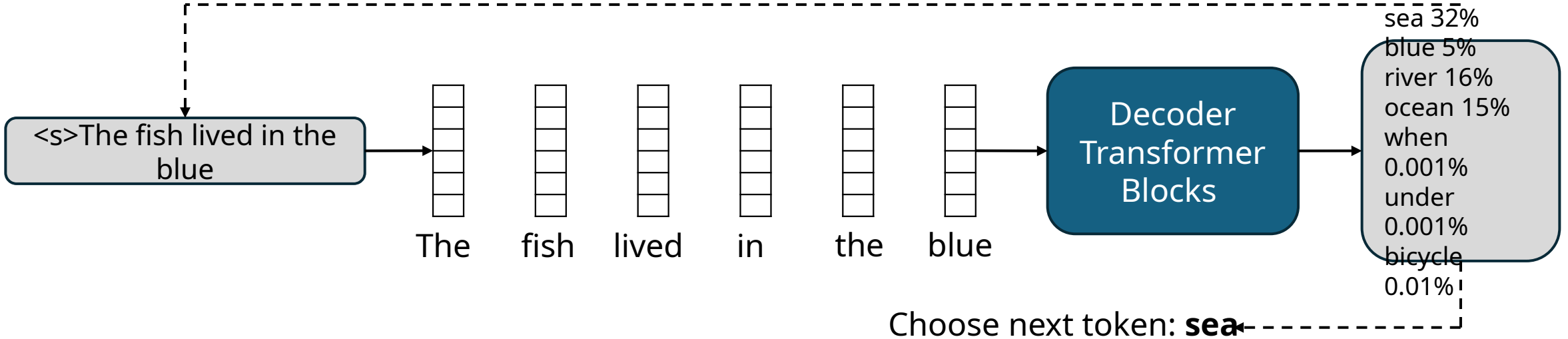
- If k = 4 we truncate the distribution to



- Now we randomly sample from the distribution. That is, 47.06% of the time we choose “sea” to be the next token, 23.53% of the time we choose “river” to be the next token, 22.06% of the time we choose “ocean” to be the next token, and 7.35% of the time we choose “blue” to be the next token.

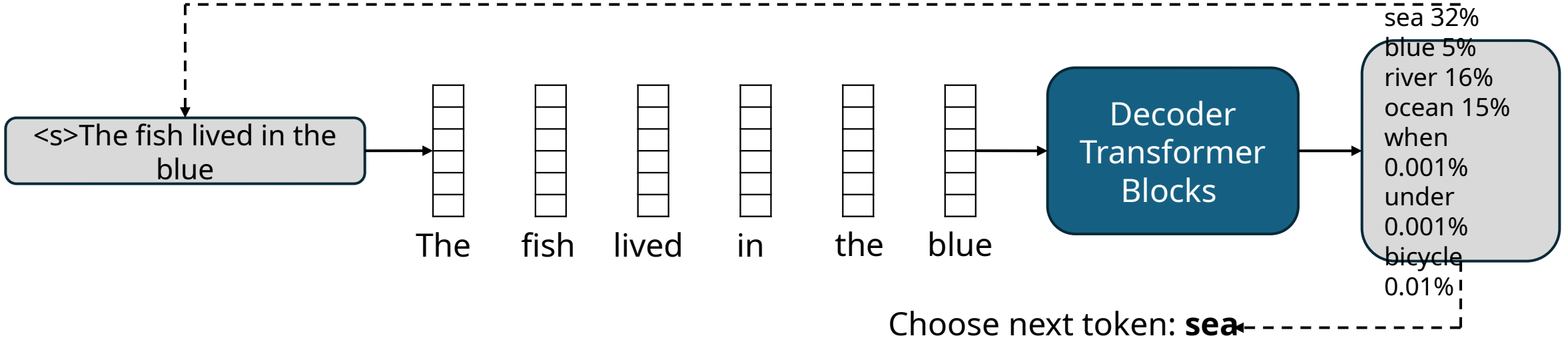
Top p (Nucleus) Sampling

- Truncate the distribution P_t, at every time step t, to the smallest set of words for which the cumulative probability just exceeds p
- Consider the probability distribution over the vocabulary in the following example



Top p (Nucleus) Sampling

- Consider the probability distribution over the vocabulary in the following example

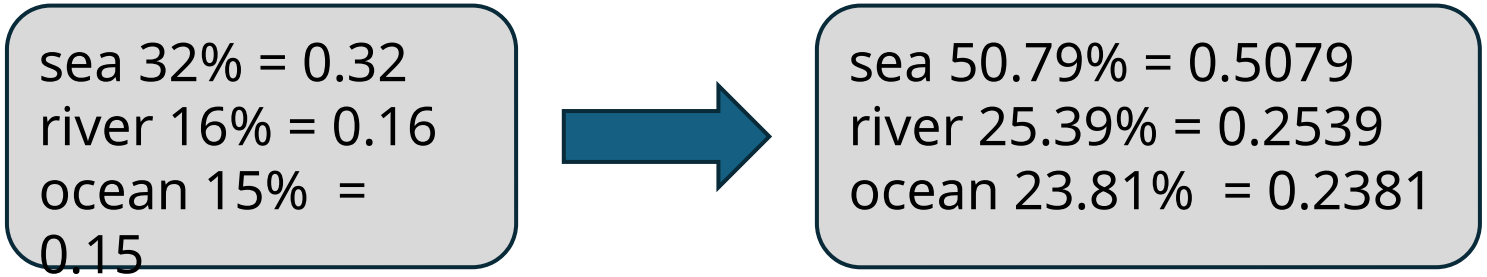


- If p = 0.6 the distribution is truncated to

sea 32% = 0.32
river 16% = 0.16
ocean 15% = 0.15

Note that this truncation is done at each time step

Top p (Nucleus) Sampling



- Now we randomly sample from the distribution. That is, 50.79% of the time we choose “sea” to be the next token, 25.39% of the time we choose “river” to be the next token, and 23.81% of the time we choose “ocean” to be the next token.

Generation parameters

```
from transformers import pipeline

# Initialize the pipeline with a real model
gen = pipeline("text-generation", model="gpt2")  # Using GPT-2 as an example model

# Define generation parameters
generation_params = {
    "max_length": 50,   # Maximum length of the output
    "top_p": 0.9,       # Nucleus sampling (top_p)
    "top_k": 50,        # Top-k sampling
    "num_beams": 5,     # Beam search (set to 1 for greedy decoding)
    "early_stopping": True,  # Stop when at least `num_beams` sentences are finished
    "temperature": 0.7  # Controls randomness (lower = more deterministic)
}

# Use the pipeline
text = "The fish lived in"
output = gen(text, **generation_params)

# Print the output
```

References

- Iyyer, M. (Instructor). (2022). Advanced Natural Language Processing (CS 685) UMass Amherst