CS846
Machine Learning for Software Engineering

Pengyu Nie

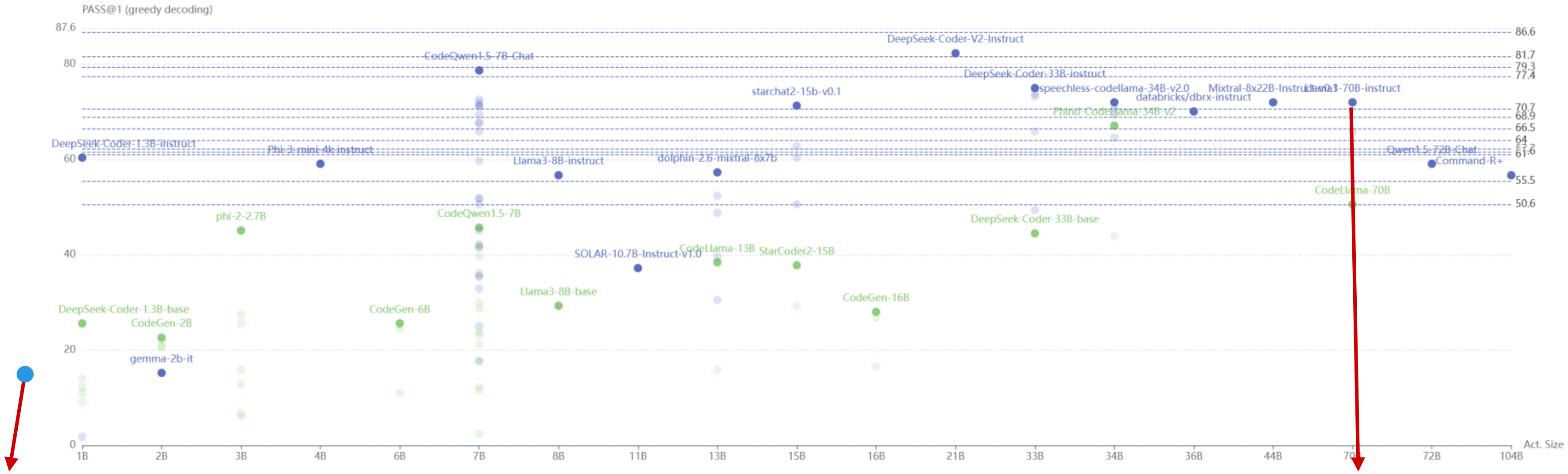
Large Language Models for Code

Training: pre-training, post-training

Inference: greedy/sampling, zero-shot/few-shot

Large Language Models

- Large transformers trained with massive amount of data



CodeT5, 2021-09-02

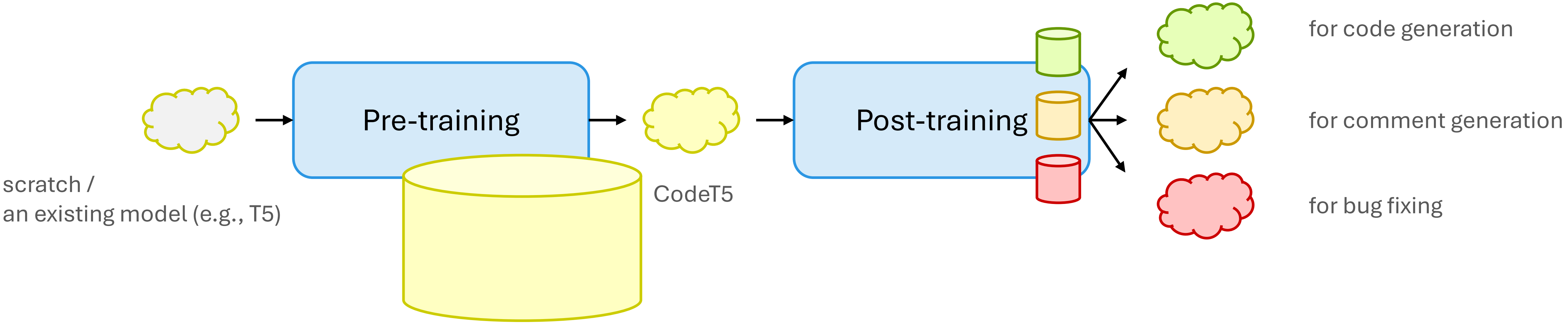
- model size: 220M (2.2e8)
- training #tokens: ~4.1B (4.1e9)
- training cost: 12d on 16 x A100 GPUs
- context window: 512
- vocab size: 32K
- #layers: 12
- model dimension: 768
- attention head:

Llama3-70B-instruct, 2024-04-18

- model size: 70B (7e10)
- training #tokens: 15T (1.5e13)
- training cost: 54d on 16,384 x H100 GPUs
- context window: up to 128K
- vocab size: 128K
- #layers: 80
- model dimension: 8K
- attention head: 64

EvalPlus leaderboard: https://evalplus.github.io/leaderboard; CodeT5: https://arxiv.org/pdf/2109.00859; Llama3: https://arxiv.org/pdf/2407.21783

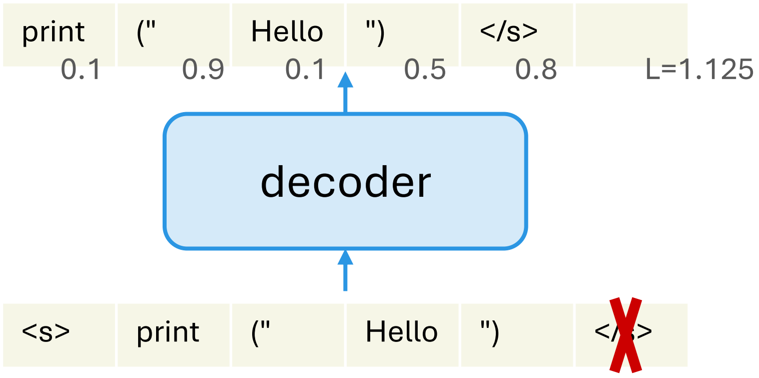
Training Overview



Pre-Training

- Massive dataset, self-supervised task(s)

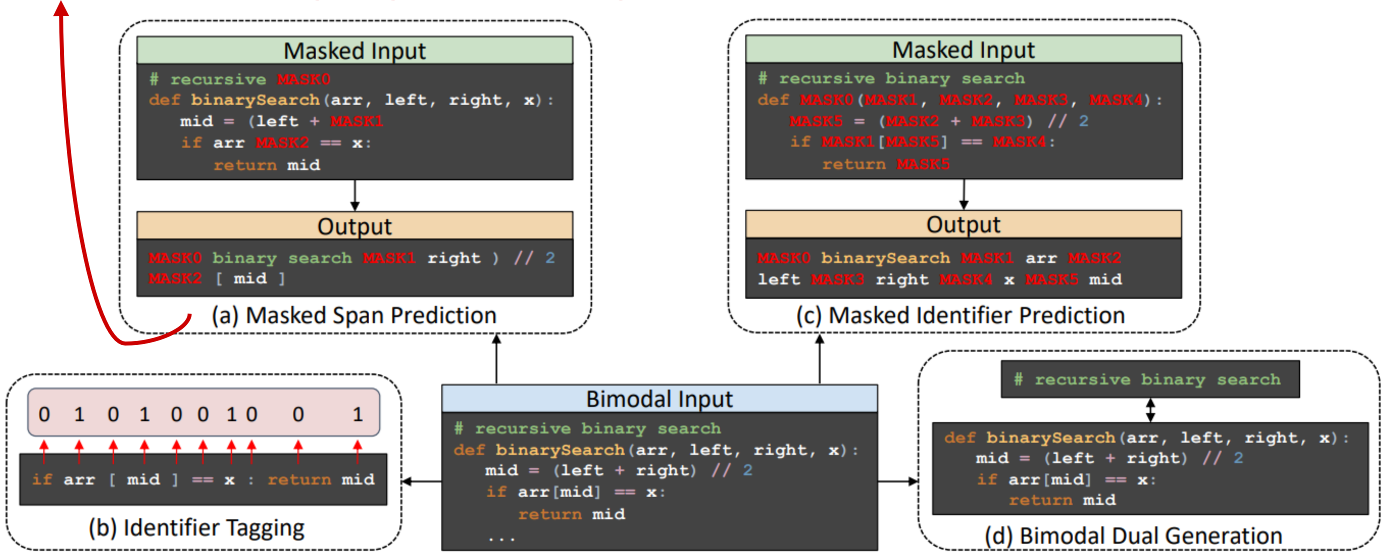
next-token prediction
(aka casual language modeling)



$$L_{CE} = -\frac{1}{n}\sum \log P(y_i)$$

cross-entropy loss
stochastic gradient descent

(aka masked language modeling)



```
# recursive MASK0
def binarySearch(arr, left, right, x):
   mid = (left + MASK1
   if arr MASK2 == x:
      return mid
```

```
MASK0 binary search MASK1 right ) // 2
MASK2 [ mid ]
```

(a) Masked Span Prediction

```
# recursive binary search
def MASK0(MASK1, MASK2, MASK3, MASK4):
   MASK5 = (MASK2 + MASK3) // 2
   if MASK1[MASK5] == MASK4:
      return MASK5
```

```
MASK0 binarySearch MASK1 arr MASK2
left MASK3 right MASK4 x MASK5 mid
```

(c) Masked Identifier Prediction

```
if arr [ mid ] == x : return mid
```

(b) Identifier Tagging

```
# recursive binary search
def binarySearch(arr, left, right, x):
   mid = (left + right) // 2
   if arr[mid] == x:
      return mid
   ...
```

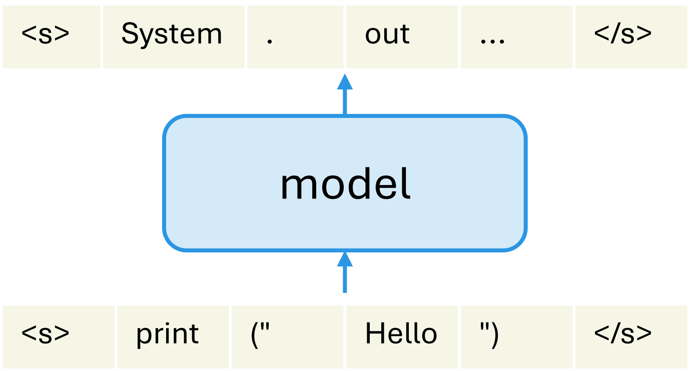
```
# recursive binary search
```

```
def binarySearch(arr, left, right, x):
   mid = (left + right) // 2
   if arr[mid] == x:
      return mid
```

(d) Bimodal Dual Generation

Post-Training / Supervised Finetuning

- Smaller dataset, usually labelled by human
- Cross-entropy loss + stochastic gradient descent (same as pre-training)

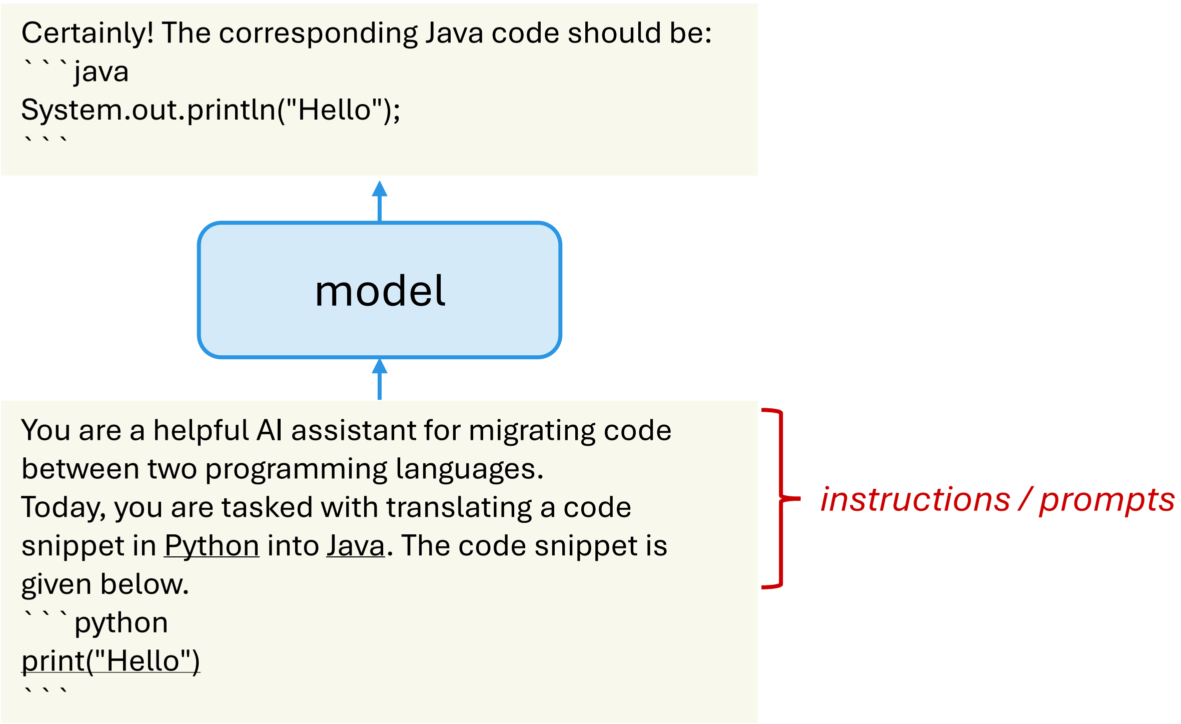


<s>	System	.	out	…	</s>

<s>	print	("	Hello	")	</s>

$$L_{CE} = -\frac{1}{n}\sum \log P(y_i)$$

Variant: instruction finetuning



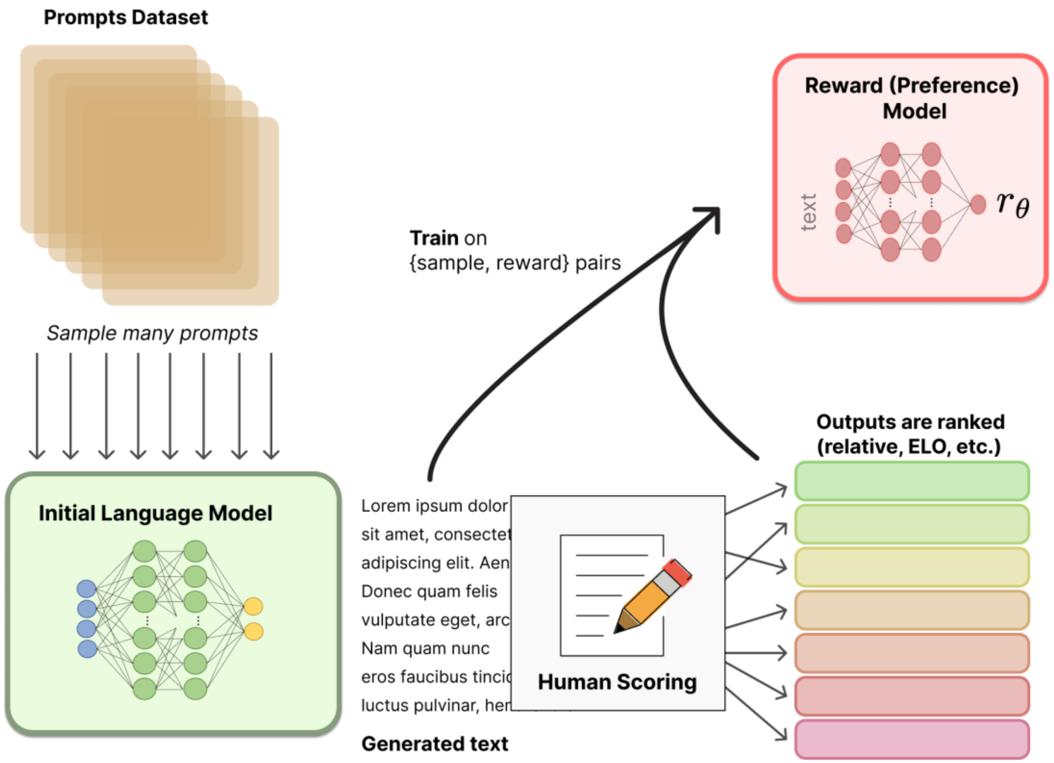
Certainly! The corresponding Java code should be:
` ` `java
System.out.println("Hello");
` ` `

You are a helpful AI assistant for migrating code between two programming languages.
Today, you are tasked with translating a code snippet in Python into Java. The code snippet is given below.
` ` `python
print("Hello")
` ` `

instructions / prompts

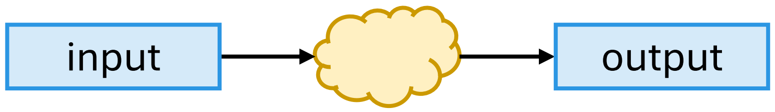
Post-Training / Reinforcement Learning with Human Feedback

- For improving instruction-following capabilities "alignment" with human preferences

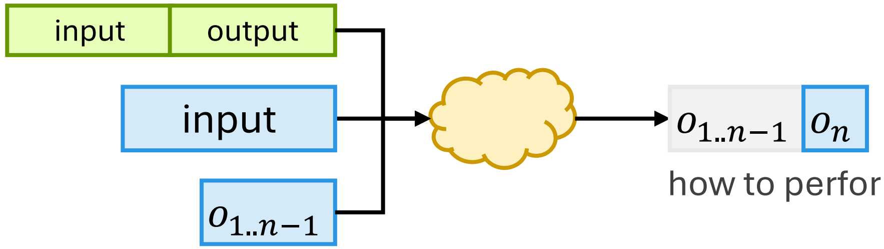

Prompts Dataset
Sample many prompts
Initial Language Model
Train on {sample, reward} pairs
Reward (Preference) Model
text
r_θ
Lorem ipsum dolor sit amet, consectet adipiscing elit. Aen Donec quam felis vulputate eget, arc Nam quam nunc eros faucibus tincid luctus pulvinar, her
Generated text
Human Scoring
Outputs are ranked (relative, ELO, etc.)

https://huggingface.co/blog/rlhf

Inference Overview



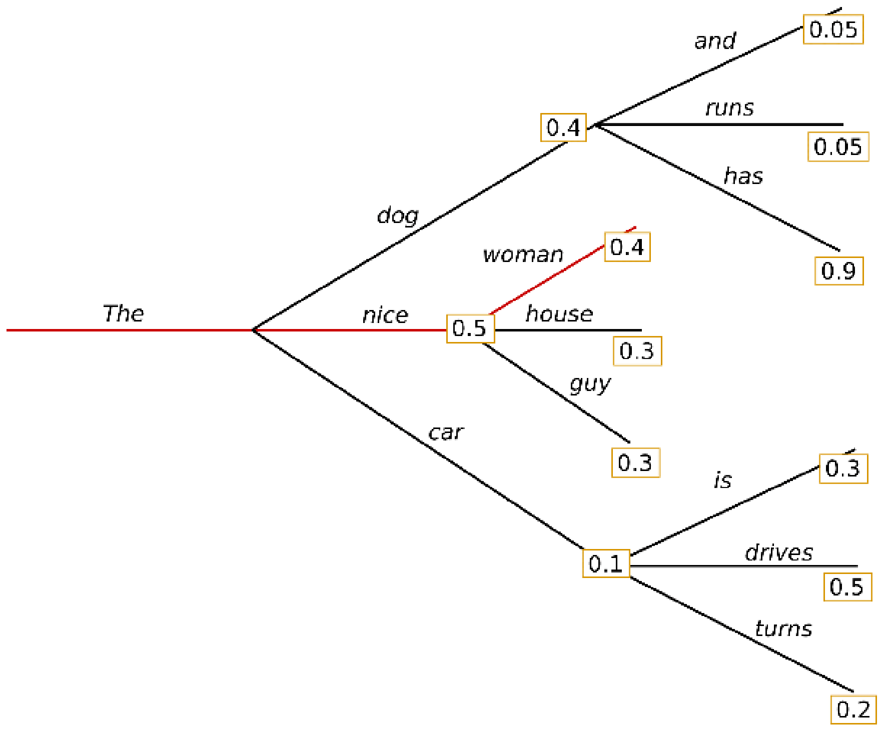
provide input-output examples?



how to perform each decoding step?

Greedy Decoding

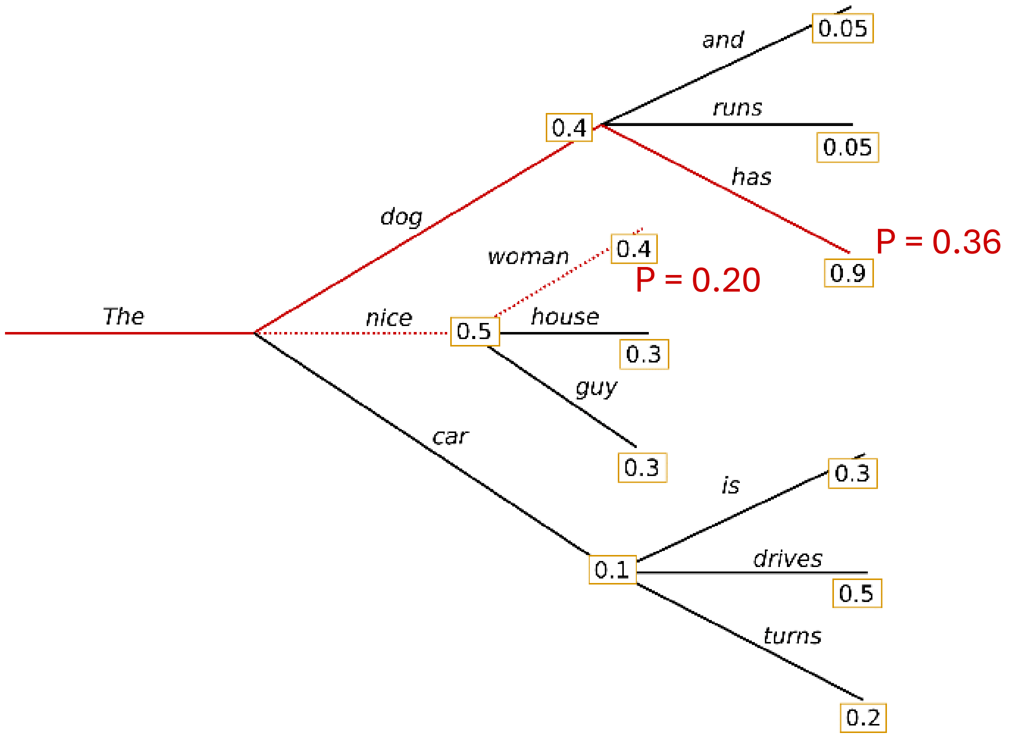
- Choose the token with highest probability at each step
- Local maximum
- Deterministic



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

Beam Search Decoding

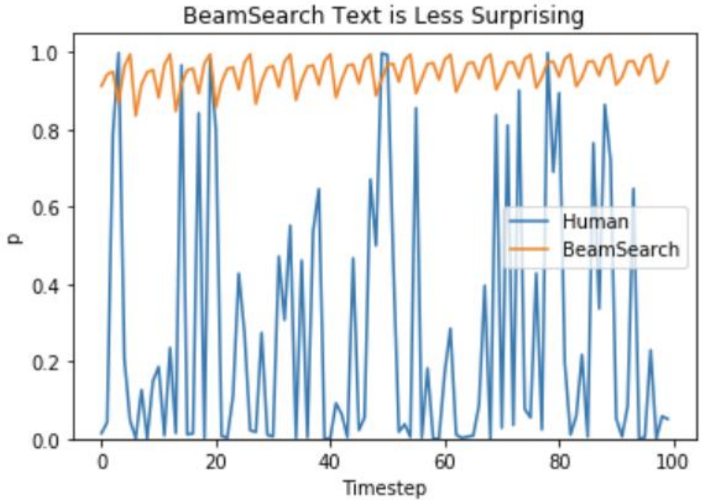
- Search for highest probability sequences, with keeping top-k most likely "hypotheses" at each time step
- Closer to global maximum
- Deterministic

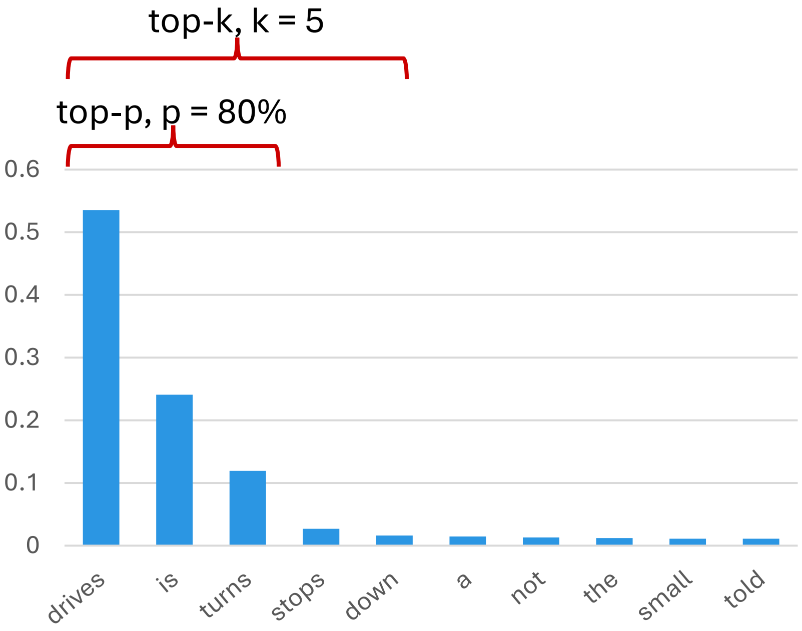


https://huggingface.co/blog/how-to-generate

Top-K / Top-P Sampling Decoding

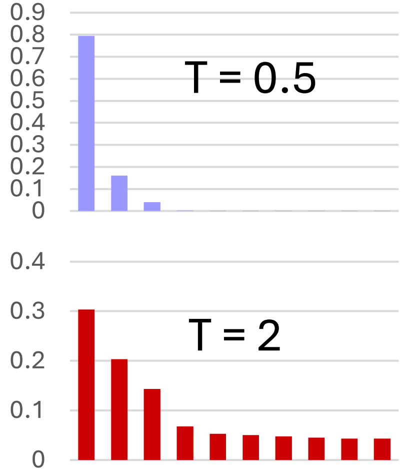
- Beam search generated text/code does not look like human written ones, because they are "boring"
- Sample next token according to probability – constrained by top-k / top-p
- Random – randomness controlled by temperature





P(w | "The", "car")

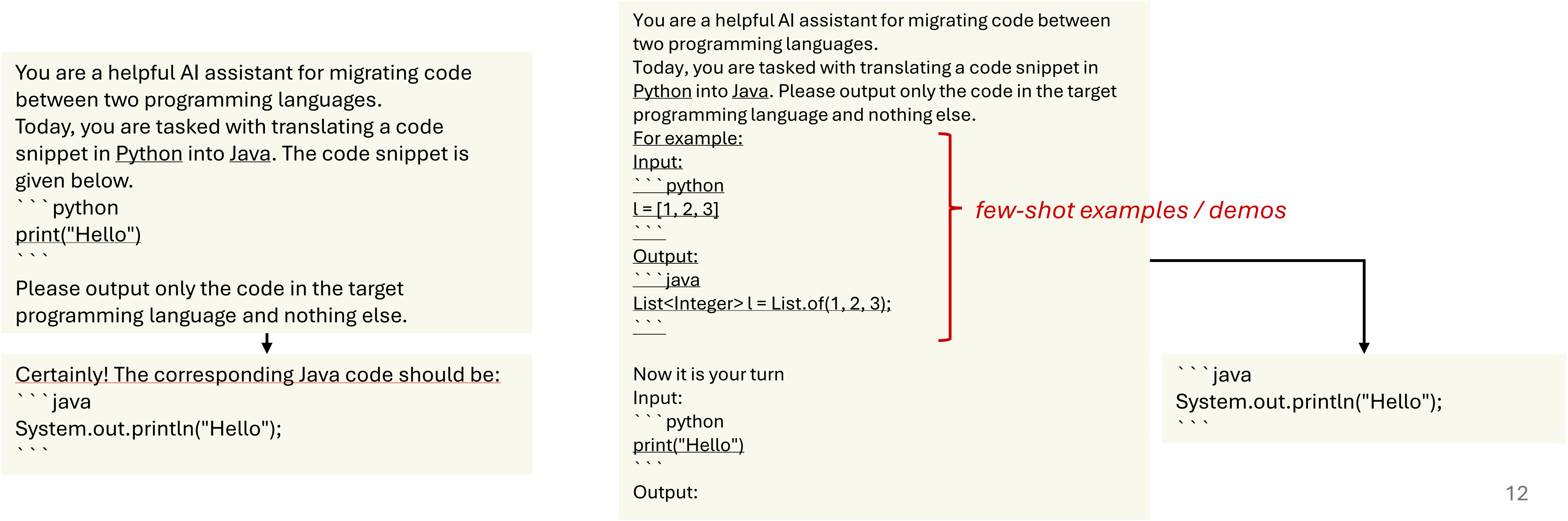
$$p_i = \text{softmax}\left(\frac{o_i}{T}\right) = \frac{\exp\frac{o_i}{T}}{\sum \exp\frac{o_i}{T}}$$



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

Zero-Shot, Few-Shot

- Zero-shot learning:
  ask the model to do something unseen during training or inference
- Few-shot learning (aka in-context learning):
  give the model a few input-output examples during inference



~~~~
You are a helpful AI assistant for migrating code
between two programming languages.
Today, you are tasked with translating a code
snippet in Python into Java. The code snippet is
given below.
```python
print("Hello")
```
Please output only the code in the target
programming language and nothing else.
~~~~

~~~~
Certainly! The corresponding Java code should be:
```java
System.out.println("Hello");
```
~~~~

~~~~
You are a helpful AI assistant for migrating code between
two programming languages.
Today, you are tasked with translating a code snippet in
Python into Java. Please output only the code in the target
programming language and nothing else.
For example:
Input:
```python
l = [1, 2, 3]
```
Output:
```java
List<Integer> l = List.of(1, 2, 3);
```

Now it is your turn
Input:
```python
print("Hello")
```
Output:
~~~~

few-shot examples / demos

~~~~
```java
System.out.println("Hello");
```
~~~~