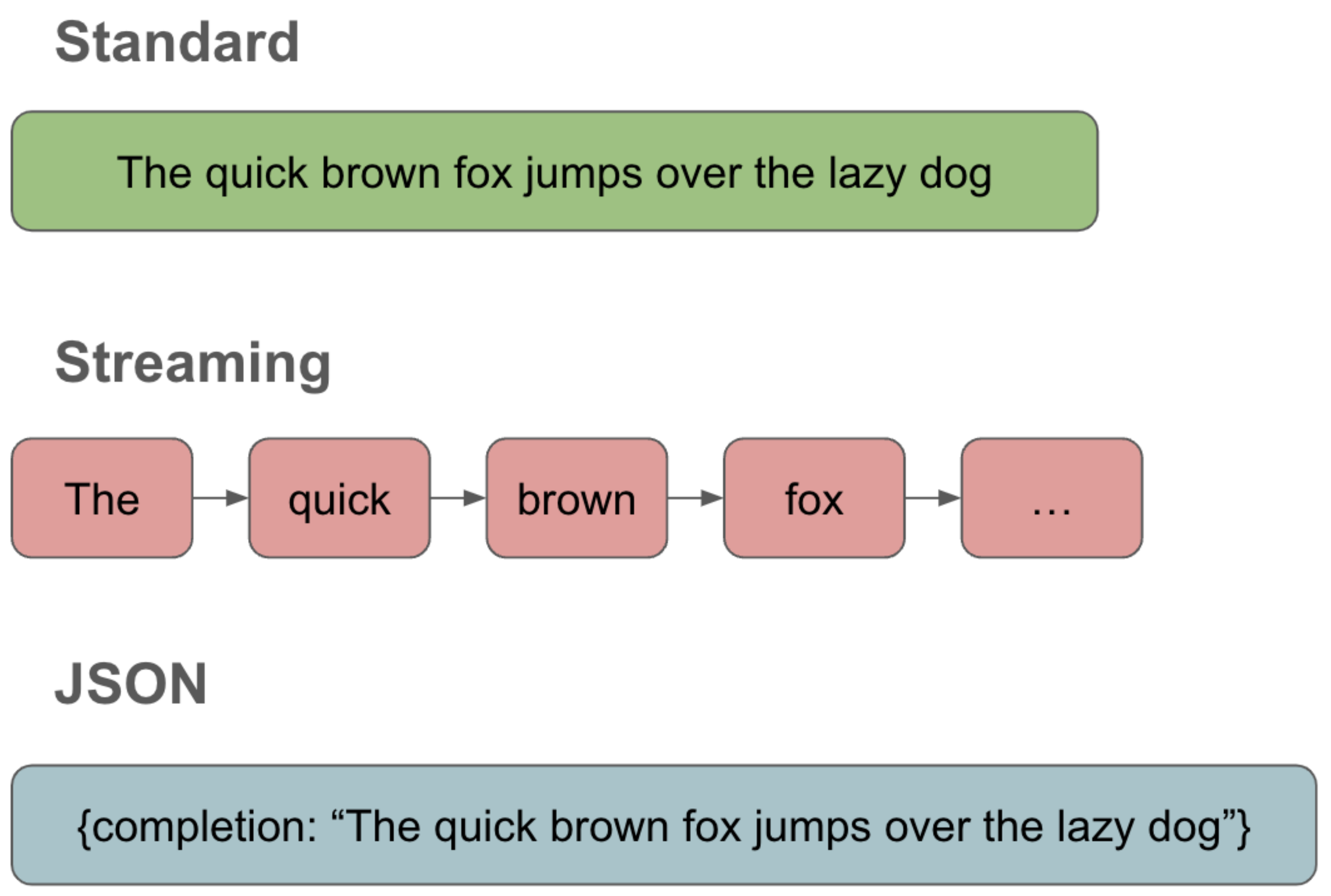
Performing inference with Llama

WORKING WITH LLAMA 3

Imtihan Ahmed
Machine Learning Engineer

Types of completions with Llama

- Basic completion
- Streaming completion
- Chat completions
  - JSON mode
  - JSON schema mode
- and more...


Standard
The quick brown fox jumps over the lazy dog
Streaming
The
quick
brown
fox
...
JSON
{completion: "The quick brown fox jumps over the lazy dog"}

[1] https://github.com/abetlen/llama-cpp-python

Setting up a basic completion

```python
from llama_cpp import Llama
llm = Llama(model_path="./models/8B/Meta-Llama-3-8B-Instruct-IQ3_M.gguf")
output = llm(
      "Q: Which galaxy is the closest to us? A: ", # Prompt
      max_tokens=32, # Max number of tokens to generate
      stop=["Q:", "\n"], # Stop when these tokens are generated
)
print(output['choices'][0]['text'])
```

```
' Milky Way!'
```

Streaming completions

```
output = llm(
        "Q: Which galaxy is the closest to us? A: ", # Prompt
        max_tokens=32, # Max number of tokens to generate
        stop=["Q:", "\n"], # Stop when these tokens are generated
        stream=True,
)
for token in output:
    print(token['choices'][0]['text'], end='')
```

```
"The Milky Way's closest neighbor is Andromeda Galaxy (also known as M31),
which is approximately 2.5 million light-years away."
```

Chat completions in JSON format

```
output = llm.create_chat_completion(
        messages=[
            {"role": "system", "content": "You are a helpful assistant that outputs
             in JSON.",},
            {"role": "user", "content": "Who won the 2020 Nobel prize in physics?
             Provide a list in 'names'"},
        ],
        response_format={"type": "json_object",}
    )
print(output['choices'][0]['message']['content'])
```

```
'{\n"names": [\n"Roger Penrose",\n"Reinhard Genzel",\n"Andrea Ghez"\n]\n}'
```

Chat completions with defined JSON schema

```
output = llm.create_chat_completion(
        messages=[...],
        response_format={
            "type": "json_object",
            "schema": {
                "type": "object",
                "properties": {"prize_type": {"type": "string"},
                               "name": {"type": "string"}},
                "required": ["prize_type", "name"]}
    )
print(output['choices'][0]['message']['content'])
```

```
'{"prize_type": "Nobel Prize in Physics",
  "names": "Andrea Ghezi, Sir Michael H. Hopkins, and Charles L. Bennett"}'
```

Let's practice!

WORKING WITH LLAMA 3

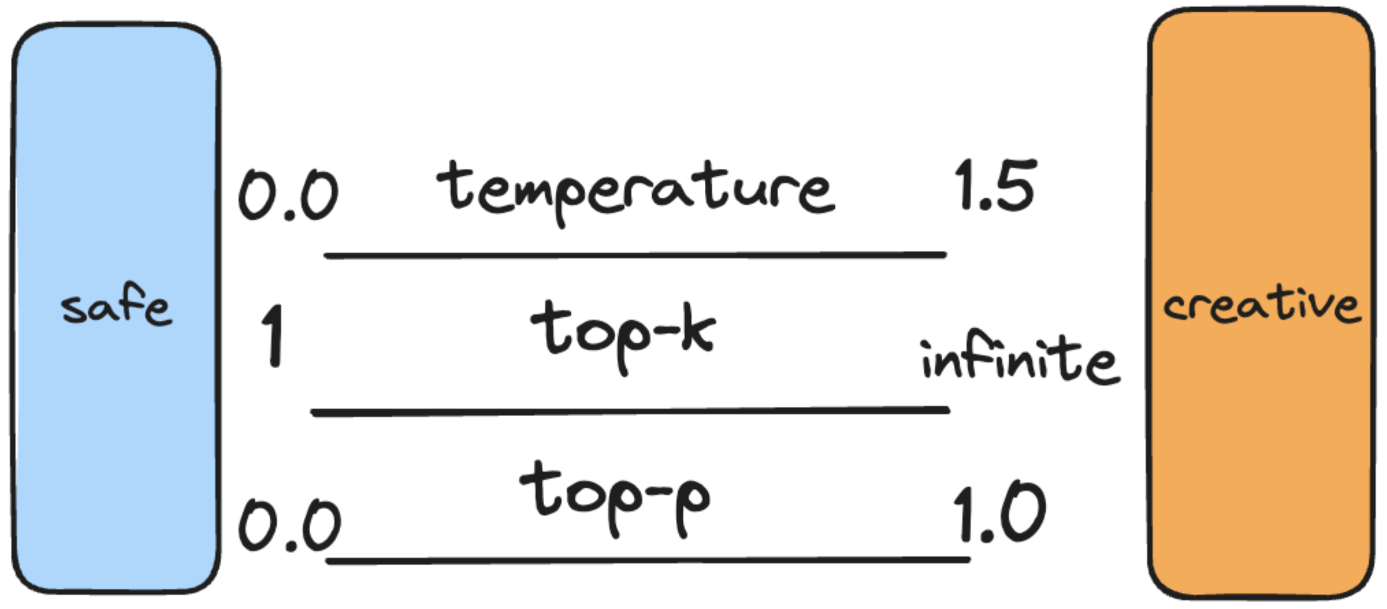
Tuning inference parameters

WORKING WITH LLAMA 3

Imtihan Ahmed
Machine Learning Engineer

Decoding with temperature

- Controls how output tokens are selected
- Three main knobs:
  - **Temperature**
  - **Top-K**
  - **Top-P**



Controlling randomness with temperature

- **High Temperature** (e.g., 1.5 or more):

```
In the bustling library of imagination, books dance with ideas under the moonlight
of creativity.
```

- **Low Temperature** (e.g., close to 0):

```
In the library, books sit quietly, filled with ideas waiting to be explored.
```

- **Zero Temperature** Temperature:

```
Books in the library contain ideas waiting to be discovered.
```

Limiting token choices with top-k

High k Value:

```
Beneath the sun's warmth, she found freedom in the water's embrace, each stroke a
graceful dance in the cerulean depths.
```

Low k Value:

```
She enjoyed swimming under the sun, feeling free in the water's embrace with each
stroke.
```

k = 1:

```
She swam under the sun, feeling free with each stroke.
```

Probabilistic filtering with top-p

- **High p Value** (e.g., 1):

```
Tall trees whispered secrets to the wind, their canopies painting dappled patterns
below.
```

- **Low p Value** (e.g., close to 0):

```
Tall trees swayed gently in the breeze, casting shadows on the forest floor.
```

- **p = 0:**

```
Tall trees stood still in the breeze, casting shadows on the forest floor.
```

Understanding decoding parameters in Llama

```python
from llama_cpp import Llama
llm = Llama(model_path="./models/8B/Meta-Llama-3-8B-Instruct-IQ3_M.gguf")
output = llm(
        "Describe the universe.",
        max_tokens=20,
        temperature=1.5,
)
print(output['choices'][0]['text'])
```

```
' In its entirety, what does it look like? How did it come to be this way? How'
```

Low vs. high temperature

- Temperature = 1.5

```
' In its entirety, what does it look like?
How did it come to be this way?'
```

- Temperature = 0.5

```
' The universe is vast and contains countless galaxies,
stars, planets, and other celestial bodies.'
```

- Temperature = 0.0

```
' The universe is vast and complex,
with an estimated 100-400 billion stars in the Milky Way'
```

Low vs. high top-p

```
output = llm(
        "Describe the universe.",
        max_tokens=20,
        top_p=0.8)
print(output['choices'][0]['text'])
```

- top_p = 0.8

```
' (2019, August 31). In Britannica Kids Homework Help.
  Retrieved from https://kids'
```

- top_p = 0.2

```
' The universe is vast and complex, with an estimated 100-400 billion stars in the Milky Way'
```

Low vs. high top-k

```python
output = llm(
        "Describe the universe.",
        max_tokens=20,
        top_k=50)
print(output['choices'][0]['text'])
```

- top_k = 50

```
" Describe everything. This is an incredibly ambitious task,
  but I'll try to give you a comprehensive overview"
```

- top_k = 5

```
' The universe, also known as the cosmos or all of existence,
  is everything that exists, including space'
```

Let's practice!

WORKING WITH LLAMA 3

datacamp

Creating an LLM inference class

WORKING WITH LLAMA 3

Imtihan Ahmed
Machine Learning Engineer

Starting from create_chat_completion

```python
from llama_cpp import Llama
llm = Llama(model_path="./Llama3-gguf-unsloth.Q4_K_M.gguf")
output = llm.create_chat_completion(
      messages = [
          {"role": "system",
           "content": "You are an assistant who speaks like Shakespeare"},
          {
             "role": "user",
             "content": "Describe the Eiffel tower."
          },
          {
            {"role": "assistant", "content": "Ti's a beautiful building."}
          }
      ]
)
```

Creating the instance attributes

```python
class Agent:
    def __init__(self, llm: Llama, system_prompt='': str, history=[]: list):
        self.llm=llm
        self.system_prompt=system_prompt
        self.history=[{"role": "system", "content": self.system_prompt}] + history
```

Custom completion method

```python
class Agent:
    def __init__(self, llm, system_prompt='', history=[]):
        ...
    def create_completion(self, user_prompt='', max_tokens=20):
        self.history += [{"role": "user", "content": user_prompt},]
        output = llm.create_chat_completion(messages=self.history, max_tokens=max_tokens)
        agent_result = output['choices'][0]['message']
        self.history += [agent_result]
        return agent_result['content']
```

Creating an agent instance

```python
from llama_cpp import Llama
llm = Llama(model_path="./Llama3-gguf-unsloth.Q4_K_M.gguf")

class Agent:
  ...

agent = Agent(llm,
              system_prompt="You only speak in the voice of Shakespeare")
res = ag1.create_completion('Describe the eiffel tower')
print(res)
```

```
'O, fair Eiffel Tower, thou majestic spire,
A marvel of engineering, a wonder'
```

Continuing the conversation

```
res2 = ag1.create_completion('Describe the eiffel tower')
print(res)
```

```
'O, fair Eiffel Tower, thou majestic spire,
A marvel of engineering, a wonder'
```

```
res2 = ag1.create_completion('Tell me more.')
print(res)
```

```
'Thou art not alone, but part of a pair, Twin towers that reach for the sky.'
```

Let's practice!

WORKING WITH LLAMA 3