**The structure of the folders of the model**

model\_repository/

└── llama3.2-1b/

          ── config.pbtxt

          ── 1/

            ── model.onnx # or the appropriate model file

              └── (other files if required)

**Get the docker container of triton inference server**

Docker pull nvcr.io/nvidia/tritonserver:24.10-py3

**Example of config.pbtxt file**

name: "NuExtract"platform: "onnxruntime\_onnx"max\_batch\_size: 8input [ { name: "input\_ids" data\_type: TYPE\_INT32 dims: [ -1, -1 ] # Dynamic batch size and sequence length }]output [ { name: "logits" data\_type: TYPE\_FP32 dims: [ -1, -1, -1 ] }]

**Script to create a model.onnx file**

import torchfrom transformers import AutoModelForCausalLM, AutoTokenizermodel\_name = "numind/NuExtract-1.5-smol"model = AutoModelForCausalLM.from\_pretrained(model\_name, torch\_dtype=torch.bfloat16, trust\_remote\_code=True)tokenizer = AutoTokenizer.from\_pretrained(model\_name, trust\_remote\_code=True)# Dummy input for exportdummy\_input = tokenizer("Example input text", return\_tensors="pt").input\_ids# Export model to ONNXtorch.onnx.export( model, (dummy\_input,), "model.onnx", opset\_version=13, # Adjust opset version if required input\_names=["input\_ids"], output\_names=["logits"], dynamic\_axes={"input\_ids": {0: "batch\_size", 1: "sequence\_length"}, "logits": {0: "batch\_size", 1: "sequence\_length"}},)print("Model successfully exported to ONNX!")

**Example of a model.pt file**

import os

import json

import numpy as np

import torch

from transformers import AutoModelForCausalLM, AutoTokenizer

import triton\_python\_backend\_utils as pb\_utils

class TritonPythonModel:

def initialize(self, args):

# Set up the cache directory for Hugging Face

os.environ["TRANSFORMERS\_CACHE"] = "/opt/tritonserver/model\_repository/smolLM2/hf-cache"

# Load model configuration from Triton server

self.model\_config = json.loads(args["model\_config"])

self.model\_params = self.model\_config.get("parameters", {})

# Define model path

default\_model = "HuggingFaceTB/SmolLM2-1.7B"

model\_name = self.model\_params.get("huggingface\_model", {}).get("string\_value", default\_model)

# Define max output length with a default value

default\_max\_length = "20"

self.max\_output\_length = int(self.model\_params.get("max\_output\_length", {}).get("string\_value", default\_max\_length))

# Initialize tokenizer and model

self.logger = pb\_utils.Logger

self.logger.log\_info(f"Loading HuggingFace model: {model\_name} with max output length: {self.max\_output\_length}")

# Load tokenizer and model

self.tokenizer = AutoTokenizer.from\_pretrained(model\_name)

self.model = AutoModelForCausalLM.from\_pretrained(model\_name)

self.model.eval() # Set the model to evaluation mode

def execute(self, requests):

responses = []

for request in requests:

# Parse inputs

input\_ids = pb\_utils.get\_input\_tensor\_by\_name(request, "input\_ids").as\_numpy()

attention\_mask = pb\_utils.get\_input\_tensor\_by\_name(request, "attention\_mask").as\_numpy()

# Convert inputs to PyTorch tensors

input\_ids\_tensor = torch.tensor(input\_ids, dtype=torch.long)

attention\_mask\_tensor = torch.tensor(attention\_mask, dtype=torch.long)

# Generate outputs

with torch.no\_grad():

outputs = self.model.generate(

input\_ids=input\_ids\_tensor,

attention\_mask=attention\_mask\_tensor,

max\_length=self.max\_output\_length

)

# Convert outputs to numpy array for Triton

output\_ids = outputs.cpu().numpy()

# Create response output tensor

output\_tensor = pb\_utils.Tensor("output\_ids", output\_ids)

responses.append(pb\_utils.InferenceResponse(output\_tensors=[output\_tensor]))

return responses

def finalize(self):

self.logger.log\_info("Finalizing the SmolLM2 model.")

**Run the container of the server with CPU and allocate the ports**

 docker run -it --rm -p 8000:8000 -p 8001:8001 -p 8002:8002 \

-v ${PWD}/model\_repository:/opt/tritonserver/model\_repository \

triton-transformer-server tritonserver --model-repository=/opt/tritonserver/model\_repository

**Link of Triton Inference Server Documentation**

<https://docs.nvidia.com/deeplearning/triton-inference-server/user-guide/docs/getting_started/quickstart.html>