```
from abc import abstractmethod

import torch
from termcolor import colored

from swarm_models.base_llm import BaseLLM
from transformers.pipelines import pipeline


class HuggingfacePipeline(BaseLLM):
    """HuggingfacePipeline

    Args:
        BaseLLM (BaseLLM): [description]
        task (str, optional): [description]. Defaults to "text-generation".
        model_name (str, optional): [description]. Defaults to None.
        use_fp8 (bool, optional): [description]. Defaults to False.
        *args: [description]
        **kwargs: [description]

    Raises:

    """

    def __init__(
        self,
    task_type: str = "text-generation",

    model_name: str = None,

    use_fp8: bool = False,

    *args,

    **kwargs,

):

    super().__init__(*args, **kwargs)

    self.task_type = task_type

    self.model_name = model_name

    self.use_fp8 = use_fp8


    if torch.cuda.is_available():

        self.use_fp8 = True

    else:

        self.use_fp8 = False


    self.pipe = pipeline(

        task_type, model_name, use_fp8=use_fp8 * args, **kwargs

    )


@abstractmethod

def run(self, task: str, *args, **kwargs) -> str:

    """Run the pipeline


    Args:

        task (str): [description]
    *args: [description]
    **kwargs: [description]

Returns:
    _type_: _description_
"""
try:
    out = self.pipeline(task, *args, **kwargs)
    return out
except Exception as error:
    print(
        colored(
            (
                "Error in"
                f" {self.__class__.__name__} pipeline:"
                f" {error}"
            ),
            "red",
        )
    )
```