```
"""
Nougat by Meta

Good for:
- transcribe Scientific PDFs into an easy to use markdown
format
- Extracting information from PDFs
- Extracting metadata from pdfs

"""

import re

import torch
from PIL import Image
from transformers import NougatProcessor, VisionEncoderDecoderModel


class Nougat:
    """
    Nougat

    Args:
        model_name_or_path: str, default="facebook/nougat-base"
        min_length: int, default=1
        max_new_tokens: int, default=30
```

```
Usage:
>>> from swarm_models.nougat import Nougat
>>> nougat = Nougat()
>>> nougat("path/to/image.png")



"""


def __init__(
    self,
    model_name_or_path="facebook/nougat-base",
    min_length: int = 1,
    max_new_tokens: int = 5000,
):
    self.model_name_or_path = model_name_or_path
    self.min_length = min_length
    self.max_new_tokens = max_new_tokens

    self.processor = NougatProcessor.from_pretrained(
        self.model_name_or_path
    )
    self.model = VisionEncoderDecoderModel.from_pretrained(
        self.model_name_or_path
    )
    self.device = "cuda" if torch.cuda.is_available() else "cpu"
```

```
    self.model.to(self.device)


def get_image(self, img: str):
    """Get an image from a path"""
    img = Image.open(img)

    if img.mode == "L":
        img = img.convert("RGB")
    return img


def __call__(self, img: str, *args, **kwargs):
    """Call the model with an image_path str as an input"""
    image = Image.open(img)
    pixel_values = self.processor(
        image, return_tensors="pt"
    ).pixel_values

    # Generate transcriptions, here we only generate 30 tokens
    outputs = self.model.generate(
        pixel_values.to(self.device),
        min_length=self.min_length,
        max_new_tokens=self.max_new_tokens,
        *args,
        **kwargs,
    )
```

```
    sequence = self.processor.batch_decode(
        outputs, skip_special_tokens=True
    )[0]
    sequence = self.processor.post_process_generation(
        sequence, fix_markdown=False
    )

    out = print(sequence)
    return out

def clean_nougat_output(raw_output):
    """Clean the output from nougat to be more readable"""
    # Define the pattern to extract the relevant data
    daily_balance_pattern = (
        r"\*\*(\d{2}/\d{2}/\d{4})\*\*\n\n\*\*([\d,]+\.\d{2})\*\*"
    )

    # Find all matches of the pattern
    matches = re.findall(daily_balance_pattern, raw_output)

    # Convert the matches to a readable format
    cleaned_data = [
        f"Date: {date}, Amount: {amount.replace(',', '')}"
        for date, amount in matches
    ]
```

```
# Join the cleaned data with new lines for readability

return "\n".join(cleaned_data)
```