```
from typing import Callable, Optional


import torch

from termcolor import colored

from transformers import AutoProcessor, IdeficsForVisionText2Text


from swarm_models.base_multimodal_model import BaseMultiModalModel




def autodetect_device():
    """
    Autodetects the device to use for inference.


    Returns
    -------
        str
            The device to use for inference.
    """
    return "cuda" if torch.cuda.is_available() else "cpu"




class Idefics(BaseMultiModalModel):
    """


    A class for multimodal inference using pre-trained models from the Hugging Face Hub.
    Attributes
    ----------
    device : str
        The device to use for inference.
    model_name : str, optional
        The name of the pre-trained model model_name (default is "HuggingFaceM4/idefics-9b-instruct").
    processor : transformers.PreTrainedProcessor
        The pre-trained processor.
    max_length : int
        The maximum length of the generated text.
    chat_history : list
        The chat history.

    Methods
    -------
    infer(prompts, batched_mode=True)
        Generates text based on the provided prompts.
    chat(user_input)
        Engages in a continuous bidirectional conversation based on the user input.
    set_model_name(model_name)
        Changes the model model_name.
    set_device(device)
        Changes the device used for inference.
    set_max_length(max_length)
        Changes the maximum length of the generated text.
```

clear_chat_history()

Clears the chat history.

# Usage

```
from swarm_models import idefics

model = idefics()

user_input = "User: What is in this image? https://upload.wikimedia.org/wikipedia/commons/8/86/Id%C3%A9fix.JPG"
response = model.chat(user_input)
print(response)

user_input = "User: And who is that? https://static.wikia.nocookie.net/asterix/images/2/25/R22b.gif/revision/latest?cb=20110815073052"
response = model.chat(user_input)
print(response)

model.set_model_name("new_model_name")
model.set_device("cpu")
model.set_max_length(200)
model.clear_chat_history()
```

```
"""


def __init__(
    self,
    model_name: Optional[
        str
    ] = "HuggingFaceM4/idefics-9b-instruct",
    device: Callable = autodetect_device,
    torch_dtype=torch.bfloat16,
    max_length: int = 100,
    batched_mode: bool = True,
    *args,
    **kwargs,
):
    # Initialize the parent class
    super().__init__(*args, **kwargs)
    self.model_name = model_name
    self.device = device
    self.max_length = max_length
    self.batched_mode = batched_mode

    self.chat_history = []
    self.device = (
        device
        if device
        else ("cuda" if torch.cuda.is_available() else "cpu")
    )
    self.model = IdeficsForVisionText2Text.from_pretrained(
        model_name, torch_dtype=torch_dtype, *args, **kwargs
    ).to(self.device)

    self.processor = AutoProcessor.from_pretrained(
        model_name, *args, **kwargs
    )

def run(
    self, task: str = None, img: str = None, *args, **kwargs
) -> str:
    """
    Generates text based on the provided prompts.

    Parameters
    ----------
        task : str
            the task to perform
        batched_mode : bool, optional
            Whether to process the prompts in batched mode. If True, all prompts are
            processed together. If False, only the first prompt is processed (default is True).

    Returns
    -------
        list
        A list of generated text strings.
    """
    try:
        inputs = (
            self.processor(
                task,
                add_end_of_utterance_token=False,
                return_tensors="pt",
                *args,
                **kwargs,
            ).to(self.device)
            if self.batched_mode
            else self.processor(task, return_tensors="pt").to(
                self.device
            )
        )

        exit_condition = self.processor.tokenizer(
            "<end_of_utterance>", add_special_tokens=False
        ).input_ids

        bad_words_ids = self.processor.tokenizer(
            ["<image>", "<fake_token_around_image"],
            add_special_tokens=False,
        ).input_ids
        generated_ids = self.model.generate(
            **inputs,
            eos_token_id=exit_condition,
            bad_words_ids=bad_words_ids,
            max_length=self.max_length,
        )
        generated_text = self.processor.batch_decode(
            generated_ids, skip_special_tokens=True
        )
        return generated_text

    except Exception as error:
        print(
            colored(
                (
                    "Error in"
                    f" {self.__class__.__name__} pipeline:"
                    f" {error}"
                ),
                "red",
            )
        )

def set_model_name(self, model_name):
    """
    Changes the model model_name.
```

```
Parameters
----------
    model_name : str
        The name of the new pre-trained model model_name.
"""
self.model = IdeficsForVisionText2Text.from_pretrained(
    model_name, torch_dtype=torch.bfloat16
).to(self.device)
self.processor = AutoProcessor.from_pretrained(model_name)
```