```
from typing import Any, List


from PIL import Image

from transformers import AutoTokenizer, PreTrainedTokenizer




def count_tokens_hf(

    texts: List[str], tokenizer: PreTrainedTokenizer, model: str

) -> int:

    """

    Counts the total number of tokens in a list of texts using a tokenizer.


    Args:

        texts (List[str]): A list of texts to count tokens from.

        tokenizer (PreTrainedTokenizer): The tokenizer to use for tokenization.

        model (str): The name or path of the pre-trained model to use.


    Returns:

        int: The total number of tokens in the texts.

    """

    try:

        tokenizer = AutoTokenizer.from_pretrained(model)

        total_tokens = 0

        for text in texts:

            tokens = tokenizer.encode(text, add_special_tokens=True)

            total_tokens += len(tokens)
        return total_tokens
    except Exception as e:
        return e



# Function to calculate tokens and pricing
def calculate_pricing(
    texts: List[str] = None,
    tokenizer: PreTrainedTokenizer = None,
    images: List[str] = None,
    rate_per_million: float = 0.01,
    img_model: Any = None,
    rate_img: float = 0.003,
    return_cost: bool = True,
    return_tokens: bool = False,
) -> float:
    """
      Calculate containtaining for otal number of  texts based on the number of tokens, sentences,
words, characters, and paragraphs.

    Args:
        texts (list): A list of texts to calculate pricing for.
        tokenizer (PreTrainedTokenizer): A pre-trained tokenizer object used to tokenize the texts.
        rate_per_million (float, optional): The rate per million tokens used to calculate the cost. Defaults
to 0.01.
```

```
    Returns:
        tuple: A tuple containing the total number of tokens, sentences, words, characters, paragraphs,
and the calculated cost.

    Example usage:
    >>> tokenizer = AutoTokenizer.from_pretrained("gpt2")
    >>> texts = ["This is the first example text.", "This is the second example text."]
    >>> total_tokens, total_sentences, total_words, total_characters, total_paragraphs, cost =
calculate_pricing(texts, tokenizer)
    >>> print(f"Total tokens processed: {total_tokens}")
    >>> print(f"Total cost: ${cost:.5f}")

    """
    total_tokens = 0
    total_images_processed = 0
    image_processing_cost = 0

    for text in texts:
        # Tokenize the text and count tokens
        tokens = tokenizer.encode(text, add_special_tokens=True)
        total_tokens += len(tokens)

    if images and img_model:
        for img_path in images:
            # Load the image
            Image.open(img_path)
            # Process the image

            image_processing_cost += rate_img

            total_images_processed += 1


        # Calculate the image processing cost

        total_images_processed + rate_img


    # Calculate total cost with high precision

    cost = (total_tokens / 1_000_000) * rate_per_million

    print(f"Total cost: ${float(cost):.10f}")


    if return_cost and return_tokens:

        return total_tokens, cost


    if return_cost:

        return cost


    if return_tokens:

        return total_tokens


    return cost
```