```
import tiktoken


import concurrent.futures
from typing import List



class TikTokenizer:
    def __init__(
        self,
        model_name: str = "o200k_base",
    ):
        """
        Initializes a TikTokenizer object.

        Args:
            model_name (str, optional): The name of the model to use for tokenization. Defaults to
"gpt-4o".
        """
        try:
            self.model_name = model_name
            self.encoding = tiktoken.get_encoding(model_name)
        except Exception as e:
            raise ValueError(
                f"Failed to initialize tokenizer with model '{model_name}': {str(e)}"
            )
```

```
def encode(self, string: str) -> str:
    """
    Tokenizes a text string.

    Args:
        string (str): The input text string.

    Returns:
        str: The tokenized text string.
    """
    return self.encoding.encode(string)

def decode(self, tokens: List[int]) -> str:
    """
    Detokenizes a text string.

    Args:
        string (str): The input tokenized text string.

    Returns:
        str: The detokenized text string.
    """
    return self.encoding.decode(tokens)

def count_tokens(self, string: str) -> int:
    """
```

```
Returns the number of tokens in a text string.

Args:
    string (str): The input text string.

Returns:
    int: The number of tokens in the text string.
"""
num_tokens = 0

def count_tokens_in_chunk(chunk):
    nonlocal num_tokens
    num_tokens += len(self.encoding.encode(chunk))

# Split the string into chunks for parallel processing
chunks = [
    string[i : i + 1000] for i in range(0, len(string), 1000)
]

# Create a ThreadPoolExecutor with maximum threads
with concurrent.futures.ThreadPoolExecutor(
    max_workers=10
) as executor:
    # Submit each chunk for processing
    futures = [
        executor.submit(count_tokens_in_chunk, chunk)
```

```
            for chunk in chunks
        ]

        # Wait for all futures to complete
        concurrent.futures.wait(futures)

    return num_tokens


# # Path: swarms/models/tiktoken_wrapper.py
# # Example
# # Initialize the TikTokenizer object with the default model
# tokenizer = TikTokenizer()

# # Tokenize a text string

# text = "Hello, how are you doing today?"
# tokens = tokenizer.encode(text)

# print(f"Tokens: {tokens}")

# # Count the number of tokens in the text string
# num_tokens = tokenizer.count_tokens(text)
# print(f"Number of tokens: {num_tokens}")
```