```
import pytest
from transformers import AutoTokenizer
from swarms_cloud.utils.calculate_pricing import calculate_pricing


# Create a fixture for a valid tokenizer
@pytest.fixture
def valid_tokenizer():
    return AutoTokenizer.from_pretrained("gpt2")


# Create a fixture for valid texts
@pytest.fixture
def valid_texts():
    return ["This is the first example text.", "This is the second example text."]


# Test when the texts are valid
def test_calculate_pricing_valid_texts(valid_tokenizer, valid_texts):
    (
        total_tokens,
        total_sentences,
        total_words,
        total_characters,
        total_paragraphs,
        cost,
```

```
    ) = calculate_pricing(valid_texts, valid_tokenizer)
    assert total_tokens == 20
    assert total_sentences == 2
    assert total_words == 12
    assert total_characters == 68
    assert total_paragraphs == 2
    assert cost == 0.00002


# Test when the texts are empty
def test_calculate_pricing_empty_texts(valid_tokenizer):
    (
        total_tokens,
        total_sentences,
        total_words,
        total_characters,
        total_paragraphs,
        cost,
    ) = calculate_pricing([], valid_tokenizer)
    assert total_tokens == 0
    assert total_sentences == 0
    assert total_words == 0
    assert total_characters == 0
    assert total_paragraphs == 0
    assert cost == 0.0
```

```
# Test when the texts are None
def test_calculate_pricing_none_texts(valid_tokenizer):
    (
        total_tokens,
        total_sentences,
        total_words,
        total_characters,
        total_paragraphs,
        cost,
    ) = calculate_pricing(None, valid_tokenizer)
    assert total_tokens == 0
    assert total_sentences == 0
    assert total_words == 0
    assert total_characters == 0
    assert total_paragraphs == 0
    assert cost == 0.0
```