`Kosmos` Documentation

Introduction

Welcome to the documentation for Kosmos, a powerful multimodal AI model that can perform various tasks, including multimodal grounding, referring expression comprehension, referring expression generation, grounded visual question answering (VQA), and grounded image captioning. Kosmos is based on the ydshieh/kosmos-2-patch14-224 model and is designed to process both text and images to provide meaningful outputs. In this documentation, you will find a detailed explanation of the Kosmos class, its functions, parameters, and usage examples.

Overview

Kosmos is a state-of-the-art multimodal AI model that combines the power of natural language understanding with image analysis. It can perform several tasks that involve processing both textual prompts and images to provide informative responses. Whether you need to find objects in an image, understand referring expressions, generate descriptions, answer questions, or create captions, Kosmos has you covered.

Class Definition

```python
class Kosmos:
    def __init__(self, model_name="ydshieh/kosmos-2-patch14-224"):
```

Usage

To use Kosmos, follow these steps:

1. Initialize the Kosmos instance:

```python
from swarm_models.kosmos_two import Kosmos

kosmos = Kosmos()
```

2. Perform Multimodal Grounding:

```python
kosmos.multimodal_grounding(
    "Find the red apple in the image.", "https://example.com/apple.jpg"
)
```

Example 1 - Multimodal Grounding

```python
from swarm_models.kosmos_two import Kosmos

kosmos = Kosmos()
```

kosmos.multimodal_grounding(
    "Find the red apple in the image.", "https://example.com/apple.jpg"
)

3. Perform Referring Expression Comprehension:

```python
kosmos.referring_expression_comprehension(
    "Show me the green bottle.", "https://example.com/bottle.jpg"
)
```

Example 2 - Referring Expression Comprehension

```python
from swarm_models.kosmos_two import Kosmos

kosmos = Kosmos()

kosmos.referring_expression_comprehension(
    "Show me the green bottle.", "https://example.com/bottle.jpg"
)
```

4. Generate Referring Expressions:

```python
kosmos.referring_expression_generation(
    "It is on the table.", "https://example.com/table.jpg"
)
```

Example 3 - Referring Expression Generation

```python
from swarm_models.kosmos_two import Kosmos

kosmos = Kosmos()

kosmos.referring_expression_generation(
    "It is on the table.", "https://example.com/table.jpg"
)
```

5. Perform Grounded Visual Question Answering (VQA):

```python
kosmos.grounded_vqa("What is the color of the car?", "https://example.com/car.jpg")
```

Example 4 - Grounded Visual Question Answering

```python
from swarm_models.kosmos_two import Kosmos


kosmos = Kosmos()


kosmos.grounded_vqa("What is the color of the car?", "https://example.com/car.jpg")
```

6. Generate Grounded Image Captions:

```python
kosmos.grounded_image_captioning("https://example.com/beach.jpg")
```

Example 5 - Grounded Image Captioning

```python
from swarm_models.kosmos_two import Kosmos


kosmos = Kosmos()


kosmos.grounded_image_captioning("https://example.com/beach.jpg")
```

7. Generate Detailed Grounded Image Captions:

```python
kosmos.grounded_image_captioning_detailed("https://example.com/beach.jpg")
```

Example 6 - Detailed Grounded Image Captioning

```python
from swarm_models.kosmos_two import Kosmos


kosmos = Kosmos()


kosmos.grounded_image_captioning_detailed("https://example.com/beach.jpg")
```

8. Draw Entity Boxes on Image:

```python
image = kosmos.get_image("https://example.com/image.jpg")
entities = [
    ("apple", (0, 3), [(0.2, 0.3, 0.4, 0.5)]),
    ("banana", (4, 9), [(0.6, 0.2, 0.8, 0.4)]),
]
kosmos.draw_entity_boxes_on_image(image, entities, show=True)
```

Example 7 - Drawing Entity Boxes on Image

```python
from swarm_models.kosmos_two import Kosmos

kosmos = Kosmos()

image = kosmos.get_image("https://example.com/image.jpg")
entities = [
    ("apple", (0, 3), [(0.2, 0.3, 0.4, 0.5)]),
    ("banana", (4, 9), [(0.6, 0.2, 0.8, 0.4)]),
]
kosmos.draw_entity_boxes_on_image(image, entities, show=True)
```

9. Generate Boxes for Entities:

```python
entities = [
    ("apple", (0, 3), [(0.2, 0.3, 0.4, 0.5)]),
    ("banana", (4, 9), [(0.6, 0.2, 0.8, 0.4)]),
]
image = kosmos.generate_boxes(
    "Find the apple and the banana in the image.", "https://example.com/image.jpg"
)
```

Example 8 - Generating Boxes for Entities

```python
from swarm_models.kosmos_two import Kosmos

kosmos = Kosmos()
entities = [
    ("apple", (0, 3), [(0.2, 0.3, 0.4, 0.5)]),
    ("banana", (4, 9), [(0.6, 0.2, 0.8, 0.4)]),
]
image = kosmos.generate_boxes(
    "Find the apple and the banana in the image.", "https://example.com/image.jpg"
)
```

How Kosmos Works

Kosmos is a multimodal AI model that combines text and image processing. It uses the ydshieh/kosmos-2-patch14-224 model for understanding and generating responses. Here's how it works:

1. **Initialization**: When you create a Kosmos instance, it loads the ydshieh/kosmos-2-patch14-224 model for multimodal tasks.

2. **Processing Text and Images**: Kosmos can process both text prompts and images. It takes a textual prompt and an image URL as input.

3. **Task Execution**: Based on the task you specify, Kosmos generates informative responses by combining natural language understanding with image analysis.

4. **Drawing Entity Boxes**: You can use the `draw_entity_boxes_on_image` method to draw bounding boxes around entities in an image.

5. **Generating Boxes for Entities**: The `generate_boxes` method allows you to generate bounding boxes for entities mentioned in a prompt.

Parameters

- `model_name`: The name or path of the Kosmos model to be used. By default, it uses the ydshieh/kosmos-2-patch14-224 model.

Additional Information

- Kosmos can handle various multimodal tasks, making it a versatile tool for understanding and generating content.
- You can provide image URLs for image-based tasks, and Kosmos will automatically retrieve and process the images.
- The `draw_entity_boxes_on_image` method is useful for visualizing the results of multimodal grounding tasks.
- The `generate_boxes` method is handy for generating bounding boxes around entities mentioned

in a textual prompt.

That concludes the documentation for Kosmos. We hope you find this multimodal AI model valuable for your projects. If you have any questions or encounter any issues, please refer to the Kosmos documentation for
further assistance. Enjoy working with Kosmos!