Report on : Image Caption Generator

By- Biswajeet Mohanty

Contents

Abstract :

The image captioning model is a deep learning model that generates captions for images. The model uses a pre-trained ViT-GPT2 encoder-decoder architecture to generate image captions. The model takes an input image and generates a text-based caption for the image. Additionally, the model uses a Pegasus paraphrasing model to improve the generated captions. The image captioning model is trained on large datasets of images and their associated captions, which allows the model to generate accurate and relevant captions for a wide variety of images. The model is implemented using Python and several deep learning frameworks, including PyTorch and Transformers.

The Flask server has been used with two pre-trained models to generate paraphrases for an input image. First, the Salesforce image captioning model is used to generate a textual description of the image. Then, the Pegasus paraphrasing model is used to generate multiple paraphrases for the generated text. The paraphrases are returned as a JSON object in response to a POST request containing the URL of an image. The server is hosted on localhost at port 5000 and can be accessed through a web API. Also for frontend React.JS has been used.

Model builinding in Jupyter Notebook

Code:-

```
from transformers import VisionEncoderDecoderModel, ViTImageProcessor, AutoTokenizer
import torch
from PIL import Image


model = VisionEncoderDecoderModel.from_pretrained("nlpconnect/vit-gpt2-image-captioning")
feature_extractor = ViTImageProcessor.from_pretrained("nlpconnect/vit-gpt2-image-captioning")
tokenizer = AutoTokenizer.from_pretrained("nlpconnect/vit-gpt2-image-captioning")


device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model.to(device)


max_length = 16
num_beams = 4
gen_kwargs = {"max_length": max_length, "num_beams": num_beams}
```

```
import requests
from PIL import Image
from transformers import BlipProcessor, BlipForConditionalGeneration

processor = BlipProcessor.from_pretrained("Salesforce/blip-image-captioning-large")
model = BlipForConditionalGeneration.from_pretrained("Salesforce/blip-image-captioning-large")

img_url = 'https://imgv3.fotor.com/images/blog-richtext-image/part-blurry-image.jpg'
raw_image = Image.open(requests.get(img_url, stream=True).raw).convert('RGB')

# unconditional image captioning
inputs = processor(raw_image, return_tensors="pt")

out = model.generate(**inputs)
modelOutput = processor.decode(out[0], skip_special_tokens=True)
```

```
import torch
from transformers import PegasusForConditionalGeneration, PegasusTokenizer
model_name = 'tuner007/pegasus_paraphrase'
torch_device = 'cuda' if torch.cuda.is_available() else 'cpu'
tokenizer = PegasusTokenizer.from_pretrained(model_name)
model = PegasusForConditionalGeneration.from_pretrained(model_name).to(torch_device)

def get_response(input_text,num_return_sequences,num_beams):
  batch = tokenizer([input_text],truncation=True,padding='longest',max_length=60, return_tensors="pt").to(torch_device)
  translated = model.generate(**batch,max_length=60,num_beams=num_beams, num_return_sequences=num_return_sequences, temperature=1.5)
  tgt_text = tokenizer.batch_decode(translated, skip_special_tokens=True)
  return tgt_text
num_beams = 10
num_return_sequences = 10
context = modelOutput
get_response(context,num_return_sequences,num_beams)
```

Code Summary

- The code defines a PyTorch model for image captioning using a VisionEncoderDecoderModel from the Hugging Face Transformers library. The model is pretrained on a large dataset of image-caption pairs using a combination of ViT (Vision Transformer) and GPT-2 (Generative Pretrained Transformer 2) architectures. The script also loads a pre-trained image processor and tokenizer to preprocess the images and text inputs, respectively.

- After setting up the image captioning model, the script downloads an image from a URL, passes it through a pre-trained Salesforce image captioning model to obtain a textual description of the image, and then generates paraphrases of the image description using another pre-trained Pegasus model.

- The generated paraphrases are returned as a JSON object to the user. The script uses the Flask web framework to provide an HTTP API for image captioning and paraphrasing. The API endpoint receives an image URL as input and returns a list of paraphrases of the image description.

Flask (Backend)

Code:-

```
from flask import Flask, request, jsonify
from transformers import PegasusForConditionalGeneration, PegasusTokenizer
import torch
import requests
from PIL import Image
from flask_cors import CORS

app = Flask(__name__)
CORS(app)

model_name = 'tuner007/pegasus_paraphrase'
torch_device = 'cuda' if torch.cuda.is_available() else 'cpu'
tokenizer = PegasusTokenizer.from_pretrained(model_name)
model = PegasusForConditionalGeneration.from_pretrained(model_name).to(torch_device)
```

```
@app.route('/', methods=['POST'])
def generate_paraphrases():

    print(request.get_json())
    link = request.get_json()['img_url']

    # Download and open the image
    raw_image = Image.open(requests.get(link, stream=True).raw).convert('RGB')

    # Get the image caption using the Salesforce model
    from transformers import BlipProcessor, BlipForConditionalGeneration
    processor = BlipProcessor.from_pretrained("Salesforce/blip-image-captioning-large")
    model = BlipForConditionalGeneration.from_pretrained("Salesforce/blip-image-captioning-large")
    inputs = processor(raw_image, return_tensors="pt")
    out = model.generate(**inputs)
    context = processor.decode(out[0], skip_special_tokens=True)
```

```
# Generate paraphrases using the Pegasus model
    num_beams = 10
    num_return_sequences = 10
    paraphrases = get_response(context,num_return_sequences,num_beams)

    # Return the paraphrases as a JSON object
    response = jsonify(paraphrases)
    response.headers.add('Access-Control-Allow-Origin', '*')
    response.headers.add('Access-Control-Allow-Headers', 'Content-Type,Authorization')
    response.headers.add('Access-Control-Allow-Methods', 'GET,PUT,POST,DELETE,OPTIONS')
    return response


def get_response(input_text,num_return_sequences,num_beams):
    batch = tokenizer([input_text],truncation=True,padding='longest',max_length=60, return_tensors="pt").to(torch_device)
    lated = model.generate(**batch,max_length=60,num_beams=num_beams, num_return_sequences=num_return_sequences,
temperature=1.5)
```

```
 tgt_text = tokenizer.batch_decode(translated, skip_special_tokens=True)
    return tgt_text


if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0', port=5000)
```

Code Summary

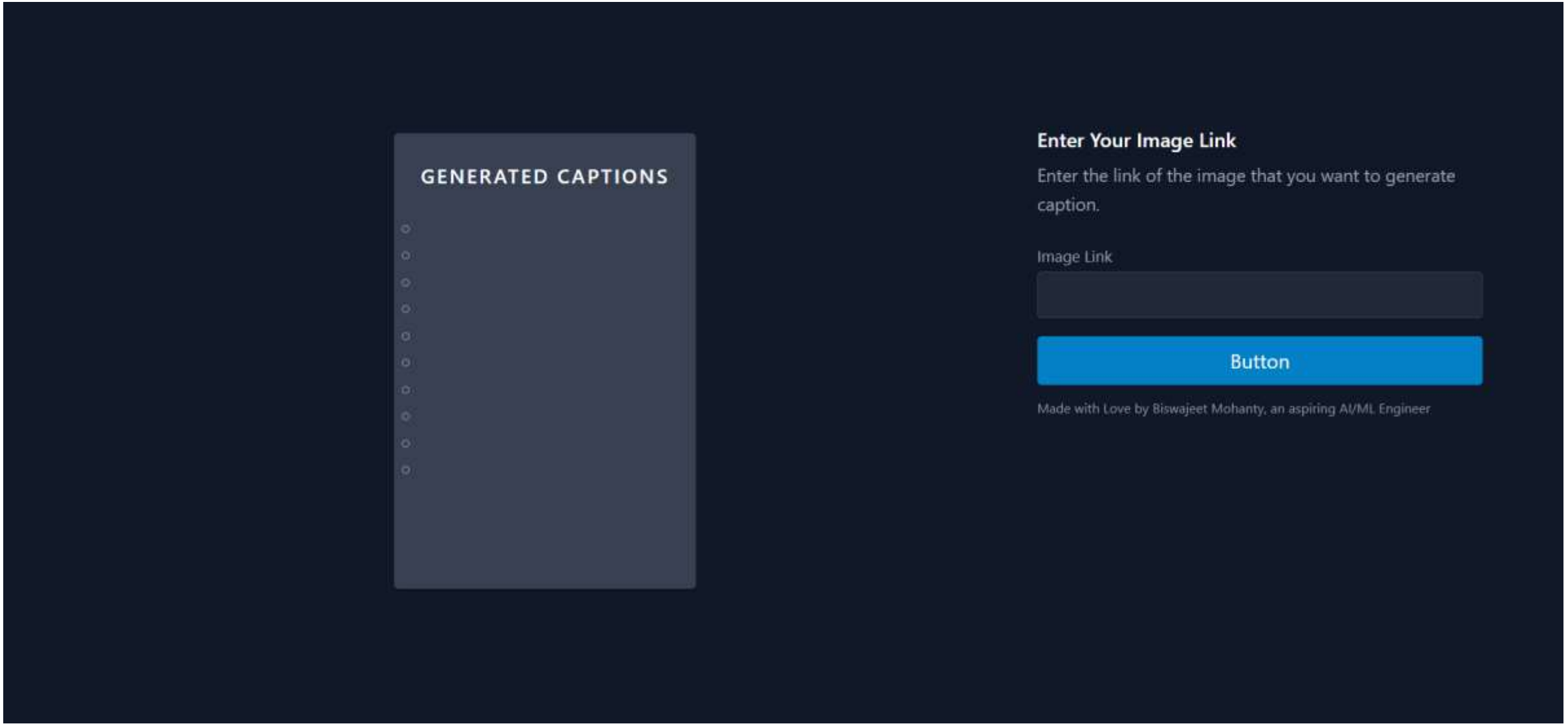
- This is a Python Flask web application that takes an image URL as input, generates a caption for the image using the Salesforce Blip image captioning model, and then generates paraphrases for the caption using the Pegasus model.

- The app runs on the localhost on port 5000 and has an API endpoint defined as '/'. The endpoint takes a POST request with a JSON object containing an image URL, and returns a JSON object containing a list of paraphrases for the image caption.

- The code imports necessary libraries such as Flask, transformers, PIL, and requests. The Pegasus model is loaded and set up along with the Pegasus tokenizer. The Flask app is created and initialized with the necessary settings, and a POST route is defined for the API endpoint.

- Within the route function, the input image is downloaded and converted to RGB. The Salesforce Blip image captioning model is used to generate a caption for the image. The Pegasus model is then used to generate paraphrases for the caption. The paraphrases are returned as a JSON object in the HTTP response.

Front End (React JS)



Project Explanation

https://drive.google.com/file/d/1NCsmVpOW79RKnY0g5HH5_NC4zwbWc-Qx/view?usp=share_link

Conclusion

The model in my project uses a pre-trained Salesforce image captioning model to generate a caption for an input image, and then uses a pre-trained Pegasus paraphrasing model to generate multiple paraphrases of the generated caption. The model is implemented using Flask, a Python web framework, and can be accessed through a POST request to a specified endpoint. The output of the model is a JSON object containing the generated paraphrases and the is deployed in a local host.

Overall, this model can be useful in applications that require generating diverse paraphrases of image captions, such as in natural language generation tasks or in data augmentation for training other models.

THANK YOU