Captionize

Overview

IgCaptionGenerator is a web application designed to generate Instagram-worthy captions for your images. It uses a combination of machine learning models. The application features a FastAPI-based backend and a React-based frontend.

Backend

The backend is built with FastAPI, which manages image uploads and generates captions.

Endpoints:

- POST /predict: Accepts an image file and returns generated captions.

Steps to Run the Backend:

1. Navigate to the backend directory:
   Example command: cd app/backend
2. Install the necessary dependencies:
   Example command: pip install -r requirements.txt
3. Launch the FastAPI server:
   Example command: uvicorn main:app --reload

Frontend

The frontend is developed with React and provides users with an interface to upload images and view the generated captions.

Steps to Run the Frontend:

1. Navigate to the frontend directory:
   Example command: cd app/frontend
2. Install the necessary dependencies:
   Example command: npm install
3. Start the React development server:
   Example command: npm start

Models

This project integrates multiple machine learning models for caption generation:

- BLIP model for initial caption generation.
- A custom-trained model for caption refinement.
- Qwen model to synthesize captions into a catchy, final version.

Training and Fine-Tuning

Training and fine-tuning processes are documented in the following Jupyter notebooks:

- model3.ipynb
- fine_tune.ipynb

Data
The project uses a dataset of Instagram captions for training. The dataset is preprocessed and tokenized as demonstrated in the Jupyter notebooks.

Model3.ipynb

Model3.ipynb is used for training a custom caption generation model with data from Kaggle.

Instructions:

1. **Install Dependencies**:

```
pip install nltk spacy pandas numpy tensorflow scikit-learn

python -m spacy download en_core_web_sm
```

Detailed Steps:

1. **Data Loading and Preprocessing**:
    - Load the dataset containing image paths and captions.
    - Preprocess the captions by tokenizing and encoding them.
    - Extract image features using the InceptionV3 model.
2. **Tokenization**:
    - Use NLTK and SpaCy for tokenizing the captions.
    - Create a tokenizer to convert words to integer indices.
    - Save the tokenizer for later use.
3. **Feature Extraction**:
    - Load the InceptionV3 model pre-trained on ImageNet.
    - Remove the top layer to use it as a feature extractor.
    - Preprocess images and extract features.
4. **Model Definition**:
    - Define the architecture of the caption generation model using TensorFlow/Keras.
    - The model typically consists of an encoder (for image features) and a decoder (for generating captions).
5. **Model Training**:
    - Split the dataset into training and validation sets.
    - Train the model using the training data.
    - Validate the model using the validation data.
    - Save the trained model to a file.
6. **Model Saving**:
    - Save the trained model and tokenizer to files for later use.

Fine_tune.ipynb

Fine_tune.ipynb is used for fine-tuning the custom caption generation model with data from Hugging Face.

Instructions:

1. **Install Dependencies**:

pip install datasets pandas spacy nltk tensorflow scikit-learn

python -m spacy download en_core_web_sm

Detailed Steps:

1. **Data Loading**:
    - Load the dataset from Hugging Face or another source.
    - Convert the dataset to a DataFrame and save it to CSV files.
2. **Data Preprocessing**:
    - Filter and clean the captions.
    - Tokenize the captions using NLTK and SpaCy.
    - Encode the captions using a saved tokenizer.
    - Extract image features using the InceptionV3 model.
3. **Feature Extraction**:
    - Load the InceptionV3 model pre-trained on ImageNet.
    - Remove the top layer to use it as a feature extractor.
    - Preprocess images and extract features.
4. **Model Fine-Tuning**:
    - Load the pre-trained caption generation model.
    - Split the dataset into training and validation sets.
    - Fine-tune the model using the training data.
    - Validate the model using the validation data.
5. **Model Saving**:
    - Save the fine-tuned model to a file for later use.

models_pipeline.ipynb

models_pipeline.ipynb is used for loading pre-trained models and performing inference on image and text data.

Instructions:

1. Install Dependencies: pip install numpy pickle tensorflow pillow transformers torch

Detailed Steps:

1. Imports:
- Import necessary libraries such as NumPy, TensorFlow, PIL, and Transformers.

2. Model Loading:

- Load pre-trained models using TensorFlow and Transformers.
    - Example: load_model('path_to_model') for TensorFlow models.
    - Example: BlipForConditionalGeneration.from_pretrained('model_name') for Transformers models.

3. Data Preprocessing:

- Preprocess input data for the models.
    - For images:
        - Load images using load_img.
        - Convert images to arrays using img_to_array.
        - Preprocess images using preprocess_input.
    - For text:
        - Tokenize and pad sequences using pad_sequences.

4. Inference:

- Perform inference using the loaded models.
    - Example: model.predict(preprocessed_data) for TensorFlow models.
    - Example: model.generate(input_ids) for Transformers models.

5. Post-Processing:

- Process the model outputs to obtain the final results.
    - Example: Decode generated sequences to text.

6. Save Results:

- Save the results to files if necessary.
    - Example: Use pickle to save objects.

Models_pipeline.py

Models_pipeline.py contains the logic for loading models, processing images, and generating captions. It integrates multiple models to generate Instagram-worthy captions for images.

Steps:

1. **Import Necessary Libraries**:
    - Import libraries for image processing, model loading, and text generation.
2. **Load Tokenizer**:
    - Define a function to load the tokenizer from a pickle file.
    - Load the tokenizer using the defined function.
3. **Load Caption Generation Model**:
    - Define a function to load the caption generation model.
    - Load the model using the defined function.
4. **Load BLIP Model**:
    - Load the BLIP processor and model for image captioning.
5. **Generate BLIP Caption**:
    - Define a function to generate a caption using the BLIP model.
    - Preprocess the image and generate the caption using the BLIP model.
6. **Load InceptionV3 Model**:

- Load the InceptionV3 model for feature extraction.

7. **Extract Features**:
    - Define a function to extract features from an image using the InceptionV3 model.
    - Preprocess the image and extract features using the InceptionV3 model.
8. **Top-K Sampling**:
    - Define a function to perform top-k sampling for generating captions.
    - Use the function to select the next word in the caption based on the model's predictions.
9. **Generate Caption**:
    - Define a function to generate a caption using the custom model.
    - Use the tokenizer to convert text to sequences.
    - Use the custom model to predict the next word in the caption.
    - Use top-k sampling to select the next word.
    - Continue generating words until the end token is reached or the maximum length is exceeded.
10. **Generate Captionize Caption**:
    - Define a function to generate a caption using the custom model and extracted features.
    - Extract features from the image.
    - Generate the caption using the custom model and tokenizer.
11. **Load Qwen Model**:
    - Load the Qwen model for combining captions.
    - Load the Qwen tokenizer.
12. **Generate Combined Caption**:
    - Define a function to combine captions using the Qwen model.
    - Clean the original caption.
    - Create a prompt to instruct the Qwen model to generate a combined caption.
    - Use the Qwen tokenizer to preprocess the prompt.
    - Generate the combined caption using the Qwen model.
    - Decode the generated caption.
13. **Process Image and Generate Caption**:
    - Define a function to put the entire pipeline together.
    - Generate a caption using the BLIP model.
    - Generate a caption using the custom model.
    - Combine the captions using the Qwen model.
    - Return the generated captions.