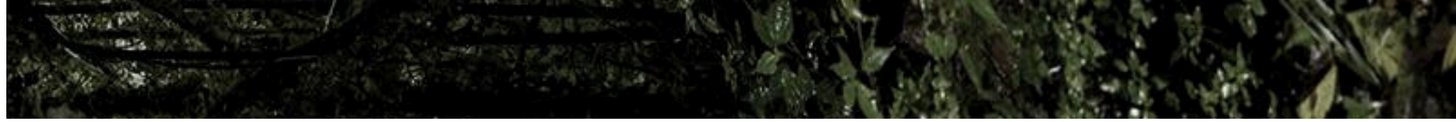
Final Project

Landmark Recognition Project

Shahad Al-Jahdali
Raneen Alharbi

2 Oct. 2024

Table of contents

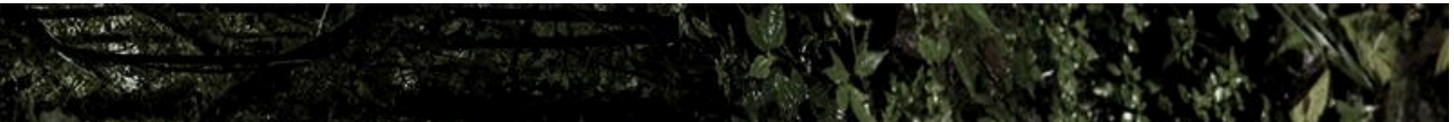


Previous Work
Audio to Summarized Transcript

The project demonstrates a pipeline for converting audio files into text transcripts and then summarizing the text using advanced Hugging Face models

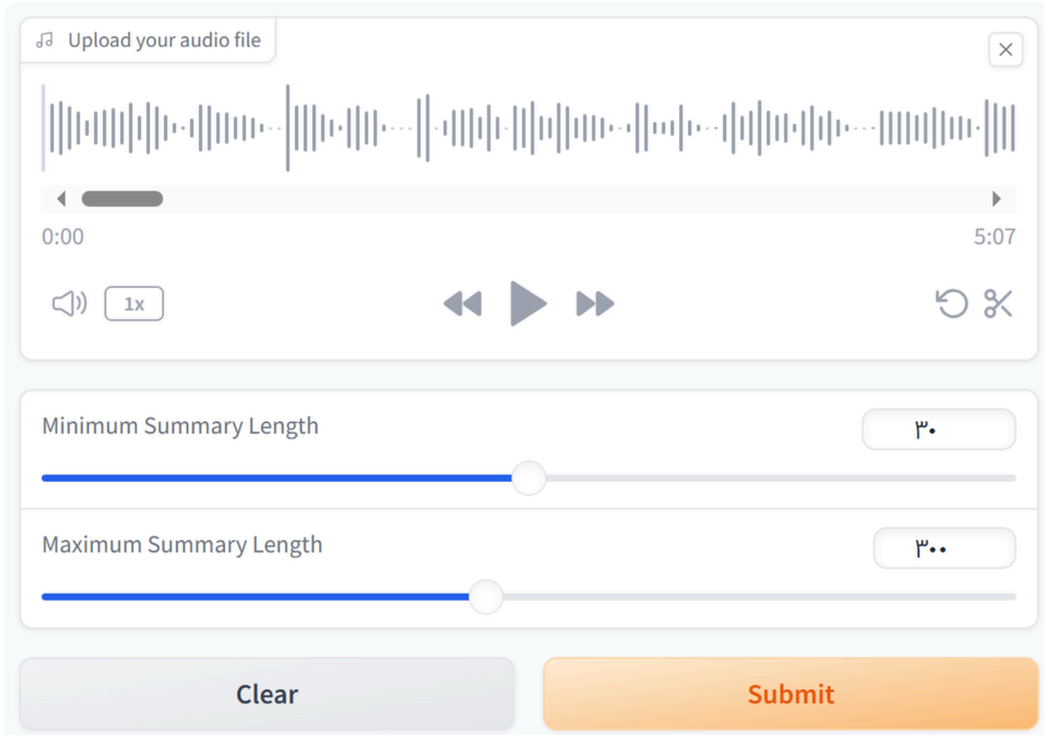
Whisper model converts audio files into text. This is the first step where spoken content is transcribed into written form.

PEGASUS model summarization model takes the transcribed text and generates a concise summary while retaining the key information.



Audio to Summarized Transcript

Upload an audio file and adjust summary length to get both the transcript and summary.



Transcript

Recorded books is pleased to present the Modern Scholar series, where great professors teach you. My name is Richard Davidson, and I'll be your host. Today, we begin a course entitled, Discovering the Philosopher in You, The Big Questions in Philosophy. Your professor is Colin McGinn of Rutgers University. Professor McGinn was educated at Oxford University and has written extensively on philosophy in publications such as The New York Review of Books, The London Review of Books, The New Republic, and The New York Times Book Review. He's written fourteen books, among them the highly praised title, The Making of a Philosopher. As well as works entitled, The Mysterious Flame, The Character of Mind, and Ethics, Evil, and Fiction. Of all the branches of intellectual inquiry, philosophy seems to be the most esoteric and out of grasp. Yet, the basic questions of philosophy, from logic to ethics, from the human mind to God, have been pondered by people around the world for centuries. Perhaps this is because every human being has their own inner philosopher, a voice within that asks the basic philosophical questions. For surely everyone on some level

Summary

Today, we begin a course entitled, Discovering the Philosopher in You, The Big Questions in Philosophy. Yet, the basic questions of philosophy, from logic to ethics, from the human mind to God, have been pondered by people around the world for centuries. For surely everyone on some level wants to know what the ultimate nature of the world is, where ethical truth comes from, what the meaning of life is, and whether or not we can really know anything.

The user interface on the hugging face where the audio is loaded and the summary length is specified to get the output.

Introduction

The 'Landmark Recognition and Description' project aims to create a bilingual AI application for Arabic and English. It identifies landmarks in images using CLIP for classification and BLIP for caption generation.

The system retrieves information from Wikipedia, summarizes it in both languages, and converts the text to speech using Google Text-to-Speech, enhancing user experience.

Objectives

- Generate image captions
- Classify images based on textual descriptions
- Summarize information in both Arabic and English
- Convert text to audio

Pipeline Implementation

Our project consists of a seamless pipeline that processes the uploaded images through multiple stages, leveraging various AI models to achieve our objectives.

Components of the Pipeline:

Image Input

Importing the landmark image.

Caption Generation

Using the BLIP model to generate a caption for the image.

Image Classification

Using the CLIP model to classify the image and identify which landmark is depicted.

Wikipedia Summary

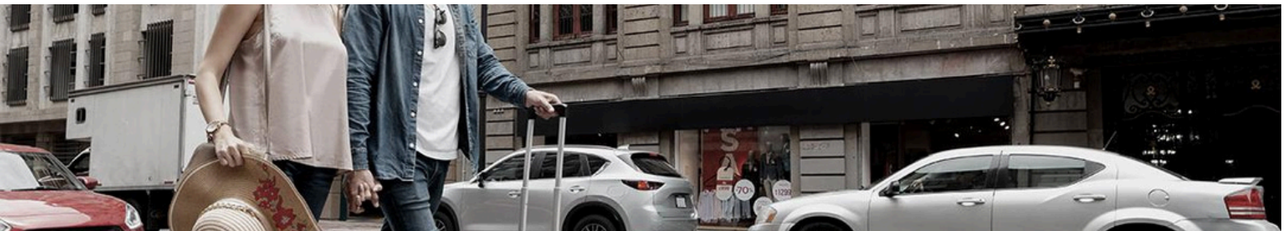
Fetching a summary from Wikipedia about the identified landmark.

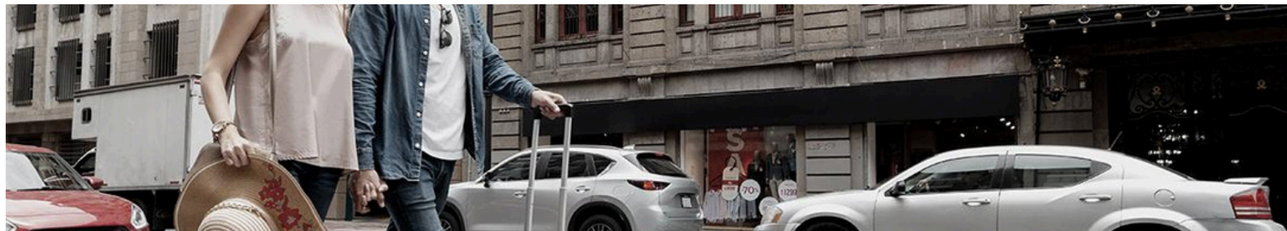
Text Summarization

Summarizing the description using Pegasus models (in English) and Auto-Arabic-Summarization (in Arabic).

Text-to-Speech

Converting the summarized text to speech using TTS models (for both English and Arabic).

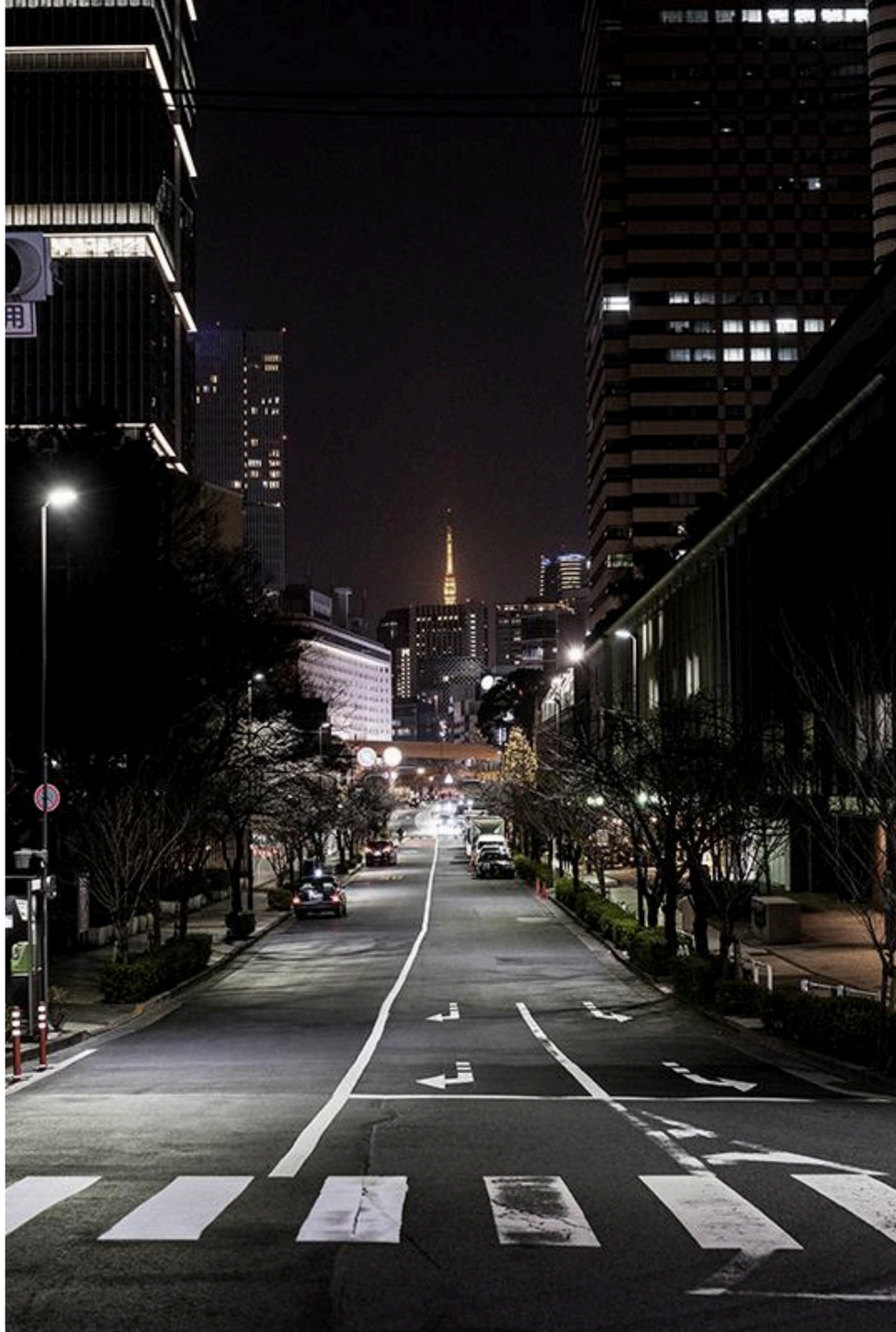


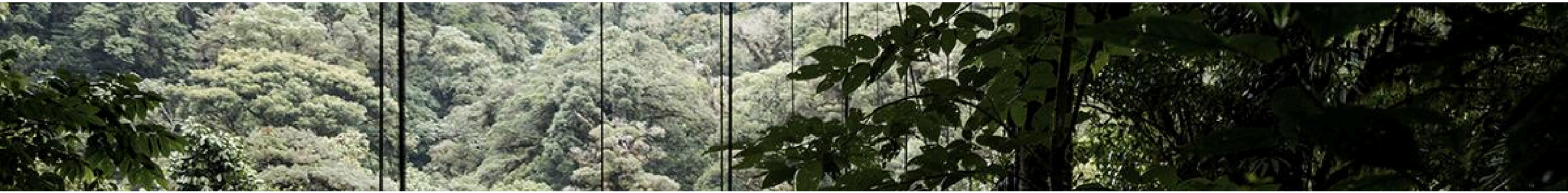


Models Used

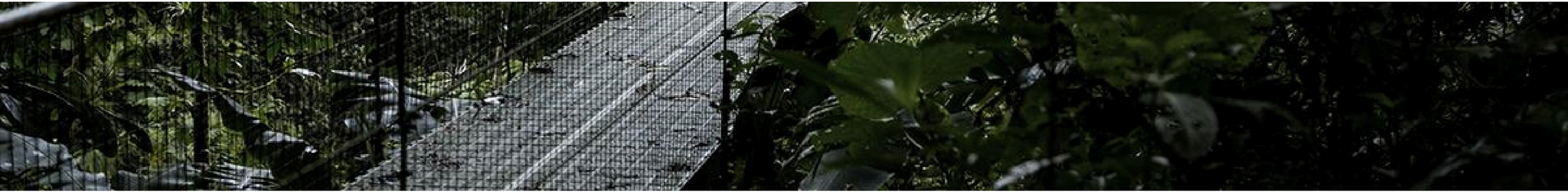
Here’s an overview of the models used and their roles in the project.

- **BLIP (Bootstrapping Language-Image Pretraining): Generates accurate descriptions of images.**
- **CLIP (Contrastive Language–Image Pretraining): Allows for precise landmark recognition.**
- **Pegasus: Condenses English descriptions into useful summaries.**
- **Auto-arabic-summarization: Efficiently summarizes Arabic text for accessibility.**
- **NLLB (No Language Left Behind): Enhances multilingual interactions.**
- **gTTS (Google Text-to-Speech): Provides audio output of the summaries.**





Code explanation



```python
# Load the BLIP model for image captioning
caption_image = pipeline("image-to-text", model="Salesforce/blip-image-captioning-large", device=device)

# Load CLIP model for image classification
clip_model = CLIPModel.from_pretrained("openai/clip-vit-large-patch14").to(device)
clip_processor = CLIPProcessor.from_pretrained("openai/clip-vit-large-patch14")

# Load the English summarization model
summarization_pipeline = pipeline("summarization", model="google/pegasus-xsum")

# Load the Arabic summarization model
arabic_summarization_pipeline = pipeline("summarization", model="abdalrahmanshahrour/auto-arabic-summarization")

# Load the translation model
translation_pipeline = pipeline("translation", model="facebook/nllb-200-distilled-600M")
```

Loading Models

```python
# Function to fetch long texts from Wikipedia
def get_wikipedia_summary(landmark_name, language='en'):
    url = f"https://{language}.wikipedia.org/wiki/{landmark_name.replace(' ', '_')}"  # Construct the URL
    response = requests.get(url)  # Make an HTTP GET request to fetch the page
    soup = BeautifulSoup(response.content, 'html.parser')  # Parse the HTML content with BeautifulSoup

    paragraphs = soup.find_all('p')  # Extract all paragraph elements
    summary_text = ' '.join([para.get_text() for para in paragraphs if para.get_text()])  # Join text from

    return summary_text[:2000]  # Return the first 2000 characters of the summary
```

```python
# Function to load landmarks from an external file
def load_landmarks(filename):
    landmarks = {}
    with open(filename, 'r', encoding='utf-8') as file:  # Open the file in read mode
        for line in file:
            if line.strip():
                english_name, arabic_name = line.strip().split('|')  # Split by the delimiter
                landmarks[english_name] = arabic_name  # Add to the dictionary
    return landmarks  # Return the dictionary of landmarks

# Load landmarks from the file
landmarks_dict = load_landmarks("landmarks.txt")
```

```
Statue of Liberty|تمثال الحرية
Eiffel Tower|برج إيفل
Great Wall of China|سور الصين العظيم
Machu Picchu|ماشوبتشو
Pyramids of Giza|أهرام الجيزة
Colosseum|الكولوسيوم
Taj Mahal|تاج محل
Sydney Opera House|دار الأوبرا في سيدني
Stonehenge|ستونهينج
Christ the Redeemer|المسيح الفادي
Acropolis of Athens|الأكروبوليس في أثينا
Hagia Sophia|آيا صوفيا
Burj Khalifa|برج خليفة
Leaning Tower of Pisa|برج بيزا المائل
Big Ben|بيغ بن
Angkor Wat|أنغكور وات
Neuschwanstein Castle|قلعة نويشفانشتاين
Petra|البتراء
Chichen Itza|تشيتشن إيتزا
Mount Rushmore|جبل راشمور
The Clock Towers|أبراج الساعة
```

Fetching Wikipedia Summary And Loading Landmarks

```python
# Function to convert text to speech
def text_to_speech(text, language='en'):
    tts = gTTS(text=text, lang=language)  # Create a gTTS object for text-to-speech
    audio_file = "summary.mp3"  # Define the audio file name
    tts.save(audio_file)  # Save the audio file
    return audio_file  # Return the path to the audio file
```

```python
# Function to classify the image using the CLIP model
def classify_image(image, labels):
    inputs = clip_processor(text=labels, images=image, return_tensors="pt", padding=True)  # Prepare inputs for CLIP model
    outputs = clip_model(**inputs)  # Get model outputs
    logits_per_image = outputs.logits_per_image  # Get logits for images
    probs = logits_per_image.softmax(dim=1).cpu().detach().numpy()[0]  # Compute probabilities
    top_label = labels[probs.argmax()]  # Get the label with the highest probability
    top_prob = probs.max()  # Get the highest probability value
    return top_label, top_prob  # Return top label and probability
```

Text-to-Speech Function And Image Classification

```python
# Function to summarize the description
def summarize_description(full_description, language):
    if language == 'ar':
        return arabic_summarization_pipeline(full_description, max_length=150, min_length=50, do_sample=False)[0]['summary_text']  # Summarize in Arabic
    else:
        return summarization_pipeline(full_description, max_length=150, min_length=50, do_sample=False)[0]['summary_text']  # Summarize in English
```

```python
# Function to translate the caption and classification result
def translate_results(caption, top_label, top_prob, landmarks_dict, language):
    if language == 'ar':
        caption_translated = translation_pipeline(caption, src_lang='eng_Latn', tgt_lang='arb_Arab')[0]['translation_text']  # Translate caption to Arabic
        classification_result = translation_pipeline(f"أفضل مطابقة: {landmarks_dict[top_label]} باحتمالية {top_prob:.4f}", src_lang='eng_Latn', tgt_lang='arb_
    else:
        caption_translated = caption  # Keep caption in English
        classification_result = f"Best match: {top_label} with probability {top_prob:.4f}"  # Create English classification result

    return caption_translated, classification_result  # Return translated results
```

Description Summarization And Translation Function

```python
# Function to process the image and generate results
def process_image(image, language='en'):
    try:
        # Generate caption for the image
        caption = generate_caption(image)  # Call the caption generation function

        # Classify the image
        top_label, top_prob = classify_image(image, list(landmarks_dict.keys()))  # Use keys for classification

        # Determine the appropriate name to use based on the language
        landmark_name = top_label if language == 'en' else landmarks_dict[top_label]
        full_description = get_wikipedia_summary(landmark_name, language)  # Get the Wikipedia summary for the top label

        # Summarize the full description
        summarized_description = summarize_description(full_description, language)  # Call the summarization function

        # Translate caption and classification result
        caption_translated, classification_result = translate_results(caption, top_label, top_prob, landmarks_dict, language)  # Call the translation

        # Convert the summarized description to speech
        audio_file = text_to_speech(summarized_description, language)  # Convert summary to audio

        # Return results formatted for Arabic
        if language == 'ar':
            return f"<div style='text-align: right;'>{caption_translated}</div>", \
                   f"<div style='text-align: right;'>{classification_result}</div>", \
                   f"<div style='text-align: right;'>{summarized_description}</div>", \
                   audio_file  # Return formatted results for Arabic
        else:
            return caption_translated, classification_result, summarized_description, audio_file  # Return results for English
    except Exception as e:
        return "Error processing the image.", str(e), "", ""  # Return error message if any exception occurs
```

Main Image Processing Function

```python
# Create Gradio interface for English
english_interface = gr.Interface(
    fn=lambda image: process_image(image, language='en'),  # Function to call on image upload
    inputs=gr.Image(type="pil", label="Upload Image"),  # Input field for image upload
    outputs=[  # Define output fields
        gr.Textbox(label="Generated Caption"),  # Output for generated caption
        gr.Textbox(label="Classification Result"),  # Output for classification result
        gr.Textbox(label="Summarized Description", lines=10),  # Output for summarized description
        gr.Audio(label="Summary Audio", type="filepath")  # Output for audio summary
    ],
    title="Landmark Recognition",  # Title of the interface
    description="Upload an image of a landmark, and we will generate a description, classify it, and provide simple information.",
    examples=[  # Examples for user
        ["/content/SOL.jfif"],
        ["/content/OIP.jfif"]
    ]
)
```

```python
# Create Gradio interface for Arabic
arabic_interface = gr.Interface(
    fn=lambda image: process_image(image, language='ar'),  # Function to call on image upload
    inputs=gr.Image(type="pil", label="تحميل صورة"),  # Input field for image upload
    outputs=[  # Define output fields
        gr.HTML(label="التعليق المولد"),  # Output for generated caption
        gr.HTML(label="نتيجة التصنيف"),  # Output for classification result
        gr.HTML(label="الوصف الملخص"),  # Output for summarized description
        gr.Audio(label="صوت الملخص", type="filepath")  # Output for audio summary
    ],
    title="التعرف على المعالم",  # Title of the interface
    description="قم بتحميل صورة لمعلم، وسنعمل على إنشاء وصف له وتصنيفه وتوفير معلومات بسيطة",  # Description of the tool
    examples=[  # Examples for user
        ["/content/SOL.jfif"],
        ["/content/OIP.jfif"]
    ]
)
```

Gradio Interfaces

Result of hugging face

English | العربية

التعرف على المعالم

قم بتحميل صورة لمعلم، وسنعمل على إنشاء وصف له وتصنيفه وتوفير معلومات بسيطة

Clear | Submit

Examples


نظرة منقطع من الجدار الكبير من الصين

أفضل مطابقة: سور الصين العظيم باحتمالية 0.9998

ويعتبر سور الصين العظيم مشروعا دفاعيا عسكريا قديما بارزا ونادرا في التاريخ المعماري البشري، حيث إنه رمز للأمة الصينية، ولم يظهر ذكاء أسلاف الصينيين فحسب، بل يجسد جهدا بذلوا فيه العرق والدماء.
.ويشتهر في العالم بتاريخه العريق وضخامة تحصيناته وقوته وقوته

صوت الملخص

0:00 | 0:31

1x

Result of hugging face

English العربية

Landmark Recognition

Upload an image of a landmark, and we will generate a description, classify it, and provide simple information.

Upload Image

Clear Submit

Generated Caption

arafed view of a section of the great wall of china

Classification Result

Best match: Great Wall of China with probability 0.9998

Summarized Description

In our series of letters from Chinese journalists, Sima Qian looks at the history and architecture of the Great Wall of China, one of the most impressive architectural feats in history, and how it has been used as a transportation and border control corridor.

Summary Audio

0:00 0:18

1x

Thank you for listening ...

<u>You can access the project through:</u>

GitHub

https://github.com/F-ran3Alh4/Landmark_Recognition

https://github.com/shahadFawaz99/Landmark_Recognition

Hugging Face

https://huggingface.co/spaces/RanAlh443/Landmark_Recognition

https://huggingface.co/spaces/ShahadFawaz99/LandmarkRecognition