Art Voice Tour

Virtual Art Museum

Table of Content

1 Project Idea

2 Project objectives

3 Project objectives

4 Results and Outputs

5 Model Justification

6 Special Measures for Arabic Support

7 Previous Work

8 Project Links

PROJECT IDEA

The ArtVoice Tour is a virtual museum where users can explore artworks by uploading an image or entering a description. The system displays the matching artwork, along with a text description and voice narration in English or Arabic, providing an immersive bilingual experience.

PROJECT OBJECTIVES

Image Search Allow users to upload an image and find the closest matching artwork.

Text Search Enable users to enter a description and retrieve matching artwork based on semantic similarity.

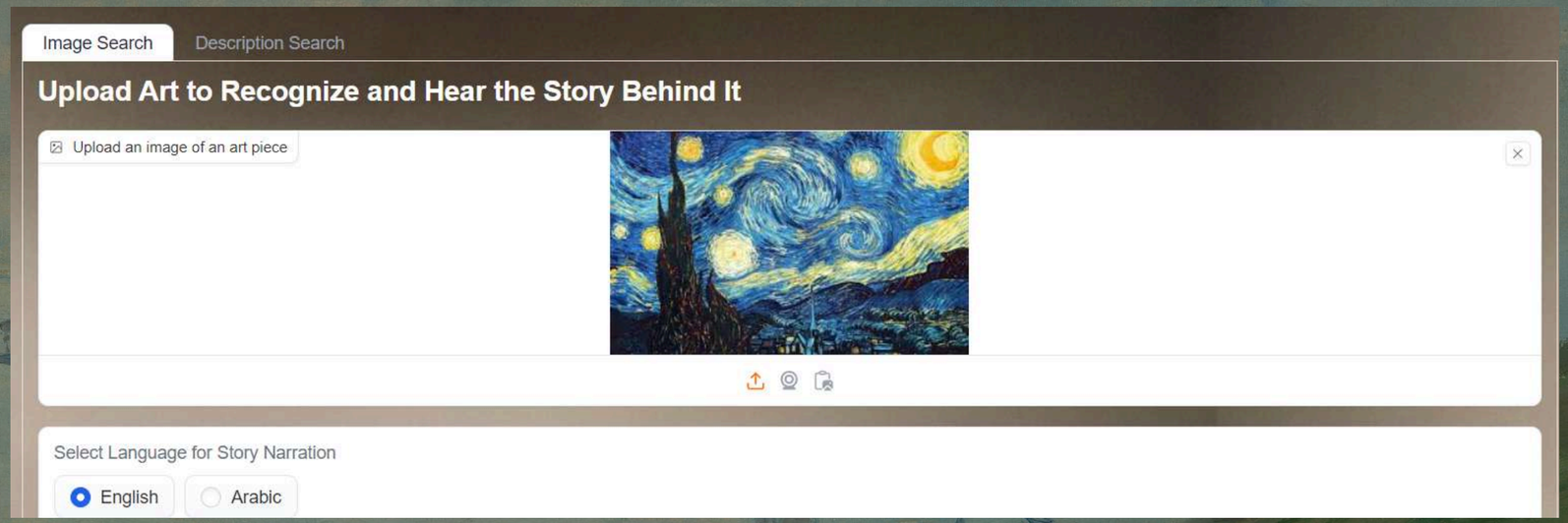
Artwork Display Present the matching artwork visually, along with detailed information about the piece.

Multilingual Support Handle both English and Arabic input and output seamlessly.

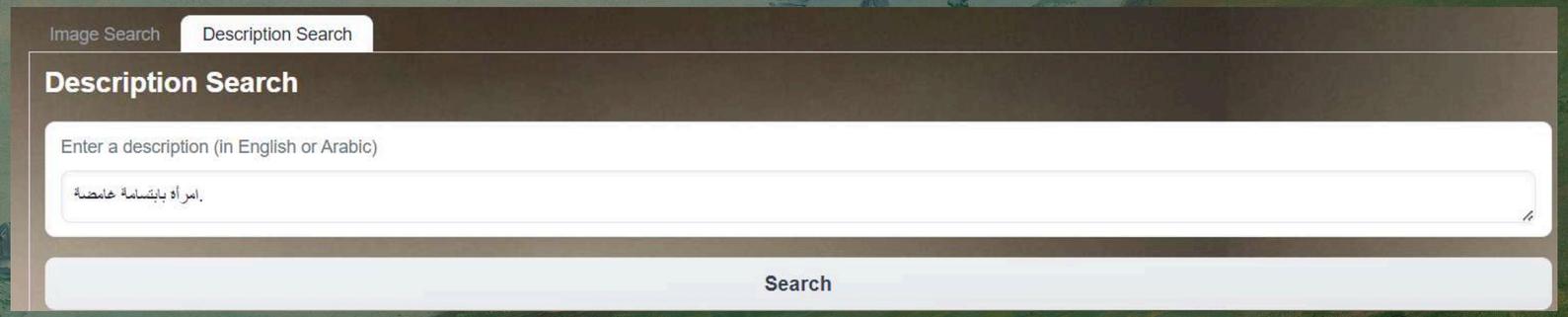
Voice Narration Provide natural-sounding voice narration of the artwork's story enhancing the user experience.

SYSTEM CHOICES

Image Search



Description Search



SYSTEM CHOICES

Image Search

1. User uploads an image and selects the output language (English or Arabic).
2. CLIP model extracts visual features from the image.
3. Image features are compared with the artwork images in the dataset.
4. The best matching artwork is displayed, along with its description and audio narration in the selected language.

Description Search

1. User enters a description.
2. Language detection determines if the input is in English or Arabic.
3. If in Arabic, the description is translated to English.
4. Semantic similarity is computed between the input and artwork descriptions in the dataset.
5. The best matching artwork is shown, along with the story in text and audio form.

SYSTEM CHOICES

Image Search Function

```python
# Function to match the uploaded image with the DataFrame to retrive the image of painting from the Dataframe
def compare_images(image, language):
    try:
        inputs = processor(images=image, return_tensors="pt")# Process the uploaded image and extract its featu
        image_features = model.get_image_features(**inputs)

        best_score = -2.0  # Initialize 'best_score' with -2.0 to ensure any valid similarity score (-1.0 to 1
        best_match_idx = None

        for idx, image_url in enumerate(df['image_url']):
            db_image = fetch_image_from_url(image_url)
            if db_image is None:
                continue

            try:
                db_inputs = processor(images=db_image, return_tensors="pt")  # Extract features of the database
                db_image_features = model.get_image_features(**db_inputs)

                similarity = torch.nn.functional.cosine_similarity(image_features, db_image_features).item() #
                if similarity > best_score:
                    best_score = similarity
                    best_match_idx = idx
            except UnidentifiedImageError:
                continue

        if best_match_idx is None:
            return None, "Error: No valid image match found in the database.", None

        best_match = df.iloc[best_match_idx]
        return process_best_match(best_match, language)

    except UnidentifiedImageError:
        return None, "Error: The uploaded file is not a valid image.", None
    except Exception as e:
        return None, f"Error: {str(e)}", None
```

Description Search Function

```python
# Function to compare user input with descriptions in the DataFrame and return the best match Paint
def compare_description(input_text):
    try:
        language = detect(input_text) #detect the langauge of input
        if language == 'ar':
            input_text = translator_ar_to_en(input_text)[0]['translation_text']

        input_embedding = semantic_model.encode(input_text, convert_to_tensor=True)
        df_embeddings = semantic_model.encode(df["Description"].tolist(), convert_to_tensor=True)


        similarities = util.pytorch_cos_sim(input_embedding, df_embeddings).squeeze() # Compute cos
        best_match_idx = torch.argmax(similarities).item()
        best_match = df.iloc[best_match_idx]

        return process_best_match(best_match, language)

    except Exception as e:
        return None, f"Error: {str(e)}", None
```

RESULTS AND OUTPUTS

Image Matching

Displays the closest matching artwork based on the uploaded image.

Text Search

Matches description to the most similar artwork and provides the story.

Voice Narration

The story is narrated in either English or Arabic, depending on the user's preference.

RESULTS AND OUTPUTS

Process output Function

One Function since the output would be the same even if the user choose to upload image of painting or describe it

```python
# Process the result where result is shown base on selected language
def process_best_match(best_match, language):
    best_image_url = best_match["image_url"]
    best_story = best_match["Story"]

    # Translate to Arabic if the language is Arabic
    if language == "Arabic":
        best_story_translated = translate_story_to_arabic(best_story)
        info_html = f"<div dir='rtl' style='font-size: 18px; color: white; font-family: Arial, sans-serif;'>{best_story_translated}</div>"
        audio_file = text_to_speech_arabic(best_story_translated)
        return best_image_url, info_html, audio_file

    # Otherwise, use English
    info_html = f"<div style='font-size: 18px; color: white;'>{best_story}</div>"
    audio_file = text_to_speech_english(best_story)
    return best_image_url, info_html, audio_file
```

MODEL JUSTIFICATIONS

Models

```python
# Load Models
# Determine if a GPU (CUDA) is available
device = "cuda" if torch.cuda.is_available() else "cpu"

#  English Text-to-speech model
narrator = pipeline("text-to-speech", model="kakao-enterprise/vits-ljs", device=device)

# CLIP model and processor to get Embeddings of images
model = CLIPModel.from_pretrained("openai/clip-vit-base-patch32").to(device)
processor = CLIPProcessor.from_pretrained("openai/clip-vit-base-patch32")

# semantic similarity model for description search
semantic_model = SentenceTransformer('sentence-transformers/all-MiniLM-L6-v2', device=device)

# translation models for Arabic to English and English to Arabic translations
translator_ar_to_en = pipeline("translation_ar_to_en", model="Helsinki-NLP/opus-mt-ar-en", device=0 if device == "cuda" else -1)
translator_en_to_ar = pipeline("translation_en_to_arabic", model="Helsinki-NLP/opus-mt-en-ar", device=0 if device == "cuda" else -1)
```

MODEL JUSTIFICATIONS

CLIP Model (openai/clip-vit-base-patch32)

- **Purpose**: Used for **image-to-image** matching, comparing user-uploaded images to the stored images of famous artworks.
- **Justification**: CLIP's ability to extract visual features and compare them effectively makes it ideal for this task.

Sentence-Transformer (sentence-transformers/all-MiniLM-L6-v2)

- **Purpose**: Used for **semantic similarity** between user descriptions and artwork descriptions.
- **Justification**: Efficient for text comparison, ensuring accurate and fast retrieval of matching artworks based on descriptions.

MODEL JUSTIFICATIONS

English Text-to-Speech (TTS) (kakao-enterprise/vits-ljs)

- **Purpose**: To provide clear and **natural-sounding English narration** of the artwork's story, enhancing user engagement and storytelling.
- **Justification**: This model delivers high-quality, lifelike English speech, making it ideal for narrating stories in a museum-like setting.

Arabic Text-to-Speech (TTS) (gTTS)

- **Purpose**: To generate accurate and clear **Arabic speech for narrating** the artwork's story to Arabic-speaking users.
- **Justification**: Chosen over MBZUAI/speecht5_tts_clartts_ar due to faster performance, easier integration, fewer pronunciation issues, and reduced silence periods, ensuring smooth and reliable Arabic narration.

MODEL JUSTIFICATIONS

Arabic to English Translation model: (Helsinki-NLP/opus-mt-ar-en)

- **Purpose**: To **translate Arabic input into English** for text comparison during the semantic similarity search.
- **Justification**: This model is known for its high accuracy in translating from Arabic to English, ensuring that user-provided Arabic descriptions are properly understood and processed by the system.

English to Arabic Translation model: (Helsinki-NLP/opus-mt-en-ar)

- **Purpose**: To **translate English** output (text descriptions or stories) **into Arabic** for Arabic-speaking users.
- **Justification**: Chosen for its reliable performance in translating from English to Arabic, this model ensures that Arabic-speaking users receive content in their preferred language with high accuracy.

→

SPECIAL MEASURES FOR ARABIC SUPPORT

gTTS for Arabic TTS

- Provides clear, fast, and reliable Arabic narration.
- Chosen over **MBZUAI/speecht5_tts_clartts_ar** due to issues with pronunciation, silence periods, and slower performance.

Language Detection

Automatically detects if the input is in English or Arabic.

RTL Text Display

Helsinki-NLP/opus-mt-ar-en and opus-mt-en-ar are used to translate between Arabic and English, ensuring seamless handling of multilingual input and output. This allows users to interact in either language and receive content in their preferred language.

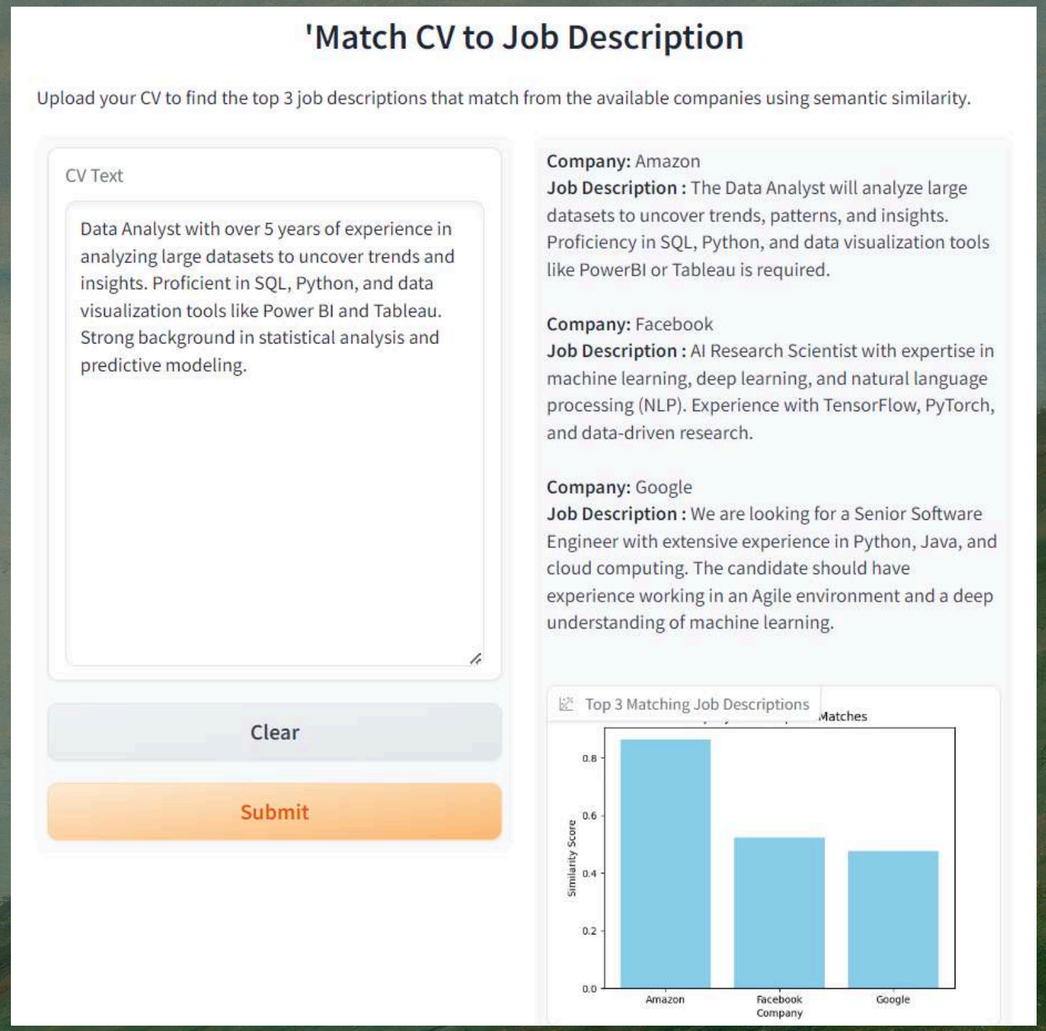
RTL Text Display

Ensures that Arabic text is presented clearly in a right-to-left (RTL) format, improving readability for Arabic-speaking users.

PREVIOUS WORK

CV to Job Description Matching Project

This project aims to help job seekers by matching their CVs to job descriptions from major companies using semantic similarity. By analyzing the text content of the CV, the system identifies and presents the top three job descriptions that best match the provided CV. Additionally, it provides translations for CVs that are not in English, as well as the corresponding job descriptions.



PREVIOUS WORK

Models

```
# Load the Hugging Face model for semantic similarity and translation model
model = SentenceTransformer('sentence-transformers/all-MiniLM-L6-v2')
translator = pipeline(task="translation", model="facebook/nllb-200-distilled-600M")
```

PREVIOUS WORK

Main Function

```python
# Function to find top 3 job descriptions matching the CV to job descriptions
def find_top_matches(cv_text):
    if not cv_text:
        return "Error: CV is empty", None

    # Translate CV to english if not in english
    cv_text, detected_lang = translate_to_english(cv_text)

    # Get job descriptions from the DataFrame as list
    descriptions = df['Job_Description'].tolist()

    # Encode both the CV and job descriptions to calcuate cosine similarities
    descriptions_embeddings = model.encode(descriptions, convert_to_tensor=True)
    cv_embedding = model.encode([cv_text], convert_to_tensor=True)
    similarities = util.pytorch_cos_sim(cv_embedding, descriptions_embeddings)[0]

    # Get the top 3 matches based on similarity scores
    top_3_indices = similarities.argsort(descending=True)[:3]  # Get the indices of the t
    top_3_matches = df.iloc[top_3_indices]
    top_3_similarities = similarities[top_3_indices].numpy()
```

```python
    #create vertical bar of top 3 match jobs to cv
    plt.bar(top_3_matches['Company'], top_3_similarities, color='skyblue')
    plt.ylabel('Similarity Score')
    plt.xlabel('Company')
    plt.title('Top 3 Job Description Matches')

    # Create a detailed summary for the top 3 job descriptions
    job_summaries = ""
    for _, row in top_3_matches.iterrows():
        # Translate job description if the detected language is not English
        job_desc_translated = translate_job_description_if_needed(row['Job_Description'], detected_lang)

        if detected_lang == 'arb_Arab':
            # Use dir="rtl" for right-to-left languages
            job_summaries += f'<div dir="rtl"><strong>الشركة:</strong> {row["Company"]} <br>'
            job_summaries += f'<strong>وصف الوظيفه :</strong> {job_desc_translated}<br><br></div>'
        else:
            # Normal left-to-right display
            job_summaries += f"<strong>Company:</strong> {row['Company']}<br>"
            job_summaries += f"<strong>Job Description :</strong> {job_desc_translated}<br><br>"

    return job_summaries, plt
```

PROJECT LINKS

GADAH AlMUIKEL's LINKS

- GitHub Repository:

https://github.com/GadahAlmuaikel/ArtVoice_Tour

- Hugging Face Space:

https://huggingface.co/spaces/ghadaAlmuaikel/ArtVoice_Tour

LUBNA AlHammadims LINKS

- GitHub Repository:

https://github.com/Lubna-Alhammadi/ArtVoice-Tour

- Hugging Face Space:

https://huggingface.co/spaces/Lubna25/ArtVoice_Tour

Thank you for listening!

Don't hesitate to ask any questions!