PoemGen

Sarah Marzouq | Nada Al-Johani

Previous Works

News Article Classification

This project classifies Arabic news articles into seven categories: Sport, Finance, Religion, Technology, Medical, Culture, and Politics. So the Users can input Arabic news articles, and the model will classify the text, returning both the label (category) and the confidence score.

- It utilizes the **MARBERT model for news article classification**, provided by Ammar Alhaj.
- Dataset: **SANAD: Single-Label Arabic News Articles Dataset for Automatic Text Categorization**.
- The Gradio library is used to build a graphical user interface (GUI) for easy interaction with the model.

Link to the project on Hugging Face
https://huggingface.co/spaces/SarahMarzouq/newsArticleClassification
Link to the project on GitHub Repository
https://github.com/sara-abdullah1/project1-Tuwaiq-Generative-AI

News Article Classification

The Expected Output:

- Label of the Article: The category that best matches the article based on the Hugging Face MARBERT model for news article classification.
- Confidence Score: A numerical score between 0.0 and 1.0 indicating the model's confidence in the classification.

Previous Works

Story-Teller-AI_Models

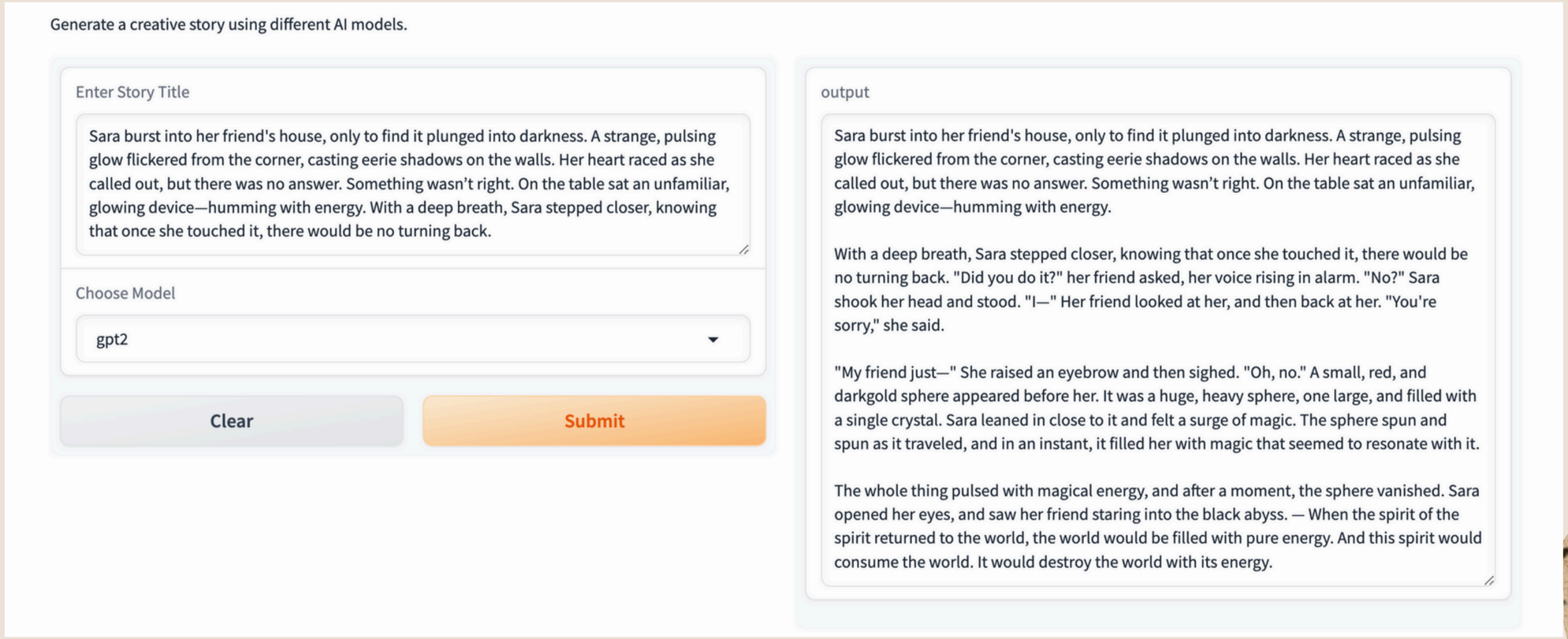
This project aims to develop an application that dynamically generates short stories based on a title entered by the user and the selection of an appropriate model for text generation. The project leverages pre-trained models from Hugging Face and allows the user to experiment with different AI models to create creative text outputs in an interactive way.

The application allows users to choose from a variety of pre-trained models available on Hugging Face (such as GPT-2 or others) for text generation.

Link to the project on Hugging Face
https://huggingface.co/spaces/NadaAljohani/StoryTellerAIModelsLink to the project on GitHub Repository
https://github.com/Nada-hs/StoryTellerAI

Previous Works

Story-Teller-AI_Models



PoemGen

The project allows users to generate poems in both Arabic and English based on their input, converts the poems to speech, translates Arabic poem into English to support text-to-image generation, and produces images that capture the essence of the poem.

Objectives

01

Generate poems in both Arabic and English based on user input

02

Convert the generated poems into speech in their respective languages

03

Translate Arabic poems into English to support image generation

04

Generate images that visually represent the essence of the generated poem

Pipelines

Text Generation:

In our project, we used two text generation pipelines to generate poems: one in Arabic and the other in English.

- **Arabic Text Generation:**
  - We chose the **akhooli/ap2023** model for Arabic poem generation because the availability of high-quality Arabic poem generator models is limited, and many existing models lack consistency. This model stood out as the best option to support our idea compared to others.

- **English Text Generation:**
  - We chose the **ashiqabdulkhader/GPT2-Poet** model. Although there is also a lack of high-quality English poem generation models, this one was the best among the available options. While it can generate poems as expected, it sometimes lacks coherence in meaning. However, compared to other models, it provided the most reliable results.

Pipelines

Text-To-Speech:

We used two text-to-speech pipelines to convert the generated poems into Arabic or English speech.

- **Arabic Text-To-Speech:**
  - The **facebook/mms-tts-ara** model was selected for Arabic text-to-speech. After evaluating multiple Text-to-Speech pipelines for Arabic, this one provided superior pronunciation and clarity compared to the others.

- **English Text-To-Speech:**
  - The **microsoft/speecht5_tts** model, developed by Microsoft, was selected for English text-to-speech due to its high-quality pronunciation and clarity. Being from Microsoft, we are confident that this model benefits from state-of-the-art technology and is highly reliable for generating natural and clear speech, making it the best choice for our project.

Pipelines

Text Translation:

Since the image generation pipeline we used in our project does not support Arabic, we use the **Helsinki-NLP/opus-mt-ar-en** pipeline to translate Arabic poems into English.
This ensures that the model can interpret the text effectively and generate accurate images based on the translation.
We chose this model as it is one of the most downloaded and highly rated models on Hugging Face specifically for translating Arabic to English.

Text-To-Image:

We chose the **runwayml/stable-diffusion-v1-5** for its superior ability to generate detailed and relevant images from textual descriptions.

Implementation

Arabic: Text Generation

```
[4]  pipe_ar = pipeline('text-generation', framework='pt', model='akhooli/ap2023', tokenizer='akhooli/ap2023')
```

GPT2-Arabic-Poetry-2023 to generate poem in the Arabic language.

English: Text Generation

```
[5]  pipe_en = pipeline("text-generation", model="ashiqabdulkhader/GPT2-Poet")
```

GPT2-Poet to generate poetry in the English language.

English: Text-To-Image:

```
[7]  pipe_image = DiffusionPipeline.from_pretrained("runwayml/stable-diffusion-v1-5")
```

runwayml/stable-diffusion-v1-5 is used to convert a poem into an image.

Implementation

Translator from Arabic to English:

```
[8]  pipe_translator = pipeline("translation", model="Helsinki-NLP/opus-mt-ar-en")
```

Since the text-to-image model doesn't support Arabic, we need to translate the Arabic poem into English using the opus-mt-ar-en model in order to generate the image.

Arabic and English: Text-To-Speech:

```
[6]  # Initialize text-to-speech models for Arabic and English
     # Arabic: text-to-speech
     synthesiser_arabic = pipeline("text-to-speech", model="facebook/mms-tts-ara")

     # English: text-to-speech
     synthesiser_english = pipeline("text-to-speech", model="microsoft/speecht5_tts")
     embeddings_dataset_english = load_dataset("Matthijs/cmu-arctic-xvectors", split="validation")
     speaker_embedding_english = torch.tensor(embeddings_dataset_english[7306]["xvector"]).unsqueeze(0)
```

- Massively Multilingual Speech: The mms-tts-ara model is used to convert Arabic poetry into speech.
- SpeechT5 (TTS task): The SpeechT5 model is used to convert English poetry into speech.

Implementation

Primary Function:

```
[8]  # Generate poem based on language and convert it to audio and image
     def generate_poem(selected_language, text):
         if selected_language == "English":
             poem = generate_poem_english(text) #retrun the generated poem from the generate_poem_english function
             sampling_rate, audio_data = text_to_speech_english(poem) #return the audio from the text_to_speech_english function
             image = generate_image_from_poem(poem) #return the image from the generate_image_from_poem function
         elif selected_language == "Arabic":
             poem = generate_poem_arabic(text) #retrun the generated poem from the generate_poem_arabic function
             sampling_rate, audio_data = text_to_speech_arabic(poem) #return the audio from the text_to_speech_arabic function
             translated_text = translate_arabic_to_english(poem) #return the translated poem from arabic to englsih, using translate_arabic_to_english function
             image = generate_image_from_poem(translated_text) #return the image from the generate_image_from_poem function

         return poem, (sampling_rate, audio_data), image
```

This function will receive 2 inputs from the Gradio interface and execute the following tasks, returning 3 outputs:

1. **The generated poem.**
2. **The audio.**
3. **The image.**

Implementation

Poem Generation Function:

```
[ ]  # Poem generation for Arabic
     def generate_poem_arabic(text):
         generated_text = pipe_ar(text, do_sample=True, max_length=96, top_k=50, top_p=1.0, temperature=1.0, num_return_sequences=1,
                                  no_repeat_ngram_size = 3, return_full_text=True)[0]["generated_text"]
         clean_text = generated_text.replace("-", "") #To get rid of the dashs generated by the model.
         return clean_text

     # Poem generation for English
     def generate_poem_english(text):
         generated_text = pipe_en(text, do_sample=True, max_length=100, top_k=0, top_p=0.9, temperature=1.0, num_return_sequences=3)[0]['generated_text']
         clean_text = generated_text.replace("</s>", "") #To get rid of the </s> generated by the model.
         return clean_text
```

This function is responsible for generating a poem (text) in either Arabic or English, based on the provided input.

Implementation

Audio Function:

```
[ ]  # Text-to-speech conversion for Arabic
     def text_to_speech_arabic(text):
         speech = synthesiser_arabic(text)
         audio_data = speech["audio"][0]  # Flatten to 1D
         sampling_rate = speech["sampling_rate"]
         return (sampling_rate, audio_data)

     # Text-to-speech conversion for English
     def text_to_speech_english(text):
         speech = synthesiser_english(text, forward_params={"speaker_embeddings": speaker_embedding_english})
         audio_data = speech["audio"]
         sampling_rate = speech["sampling_rate"]
         return (sampling_rate, audio_data)
```

This function is responsible for generating audio in either Arabic or English, based on the poem.

Implementation

Image Function:

```
[ ]  #Image Function
     def generate_image_from_poem(poem_text):
         image = pipe_image(poem_text).images[0]
         return image
```

This function is responsible for generating an image based on the poem.

Translation Function:

```
[ ]  #Translation Function from Arabic to English
     def translate_arabic_to_english(text):
         translated_text = pipe_translator(text)[0]['translation_text']
         return translated_text
```

This function is responsible for translating the Arabic poem into English, to be used by the image function, which only accepts English inputs.

Implementation

CSS Styling:

```
custom_css = """
body {
    background-color: #f4f4f9;
    color: #333;
}
.gradio-container {
    border-radius: 10px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    background-color: #fff;
}
label {
    color: #4A90E2;
    font-weight: bold;
}

input[type="text"],
textarea {
    border: 1px solid #4A90E2;
}
textarea {
    height: 150px;
}
```

```
button {
    background-color: #4A90E2;
    color: #fff;
    border-radius: 5px;
    cursor: pointer;
}
button:hover {
    background-color: #357ABD;
}

.dropdown {
    border: 1px solid #4A90E2;
    border-radius: 4px;
}

"""
```

Implementation

Examples for Gradio:

```
[ ]  examples = [
        #First parameter is for the dropdown menu, and the second parameter is for the starter of the poem
        ["English", "The shining sun rises over the calm ocean"],
        ["Arabic", "الورود تتفتح في الربيع"],
        ["English", "The night sky is filled with stars and dreams"],
        ["Arabic", "اشعة الشمس المشرقة"]
    ]
```

Provide 4 predefined inputs to demonstrate how the interface works

Implementation

Gradio Interface:

```
[ ]  my_model = gr.Interface(
        fn=generate_poem,  #The primary function that will recives the inputs (language and the starter of the poem)
        inputs=[
            gr.Dropdown(["English", "Arabic"], label="Select Language"), #Dropdown menu to select the language, either "English" or "Arabic" for the poem
            gr.Textbox(label="Enter a sentence")], #Textbox where the user will input a sentence or phrase to generate the poem (starter of the peom)

        outputs=[
            gr.Textbox(label="Generated Poem", lines=10), # Textbox to display the generated poem
            gr.Audio(label="Generated Audio", type="numpy"), #Audio output for the generated poem
            gr.Image(label="Generated Image")], #Display an image generated from the starter of the peom

        examples=examples,  #Predefined examples to guide the user how to use the interface
        css=custom_css  #Applying CSS Custeom
    )
    my_model.launch()
```

Creating a Gradio interface to generate a poem, read the poem, and generate an image based on that poem.

Results

Select Language

Arabic

Enter a sentence

نجوم السماء المضيئة

Clear

Submit

Generated Poem

نجوم السماء المضيئة في الدياجي إذا ما الليل أظلم فاستطارا.
إذا جن الظلام أرين صبحا حسبت سنا الصباح به أنارا.
كأن نجومها درر تبدت فأبدت من مطالعها خمارى.
أقول لصاحبي لما أتاني وقد أزف الرحيل عن المزار.
لقد طال البعاد وطال حزني وشبت لواعج الاشواق نارا.

Generated Audio

0:00

0:33

1x

Generated Image

Results

Select Language

English

Enter a sentence

The starry night

Clear

Submit

Generated Poem

The starry night
and in the westerly shivering cell
in summer, or the glancing eye
of pine-trees in sunlight, or the tepid,
image of a nymph whom the old goose slept;
lovely sanctitude on her side,
morrow as evening blooms,
halils the little lily by sunlight.
yet happy in the garden, sad and still,
as the savagery summer draws,
upon the stars

Generated Audio

0:00

0:33

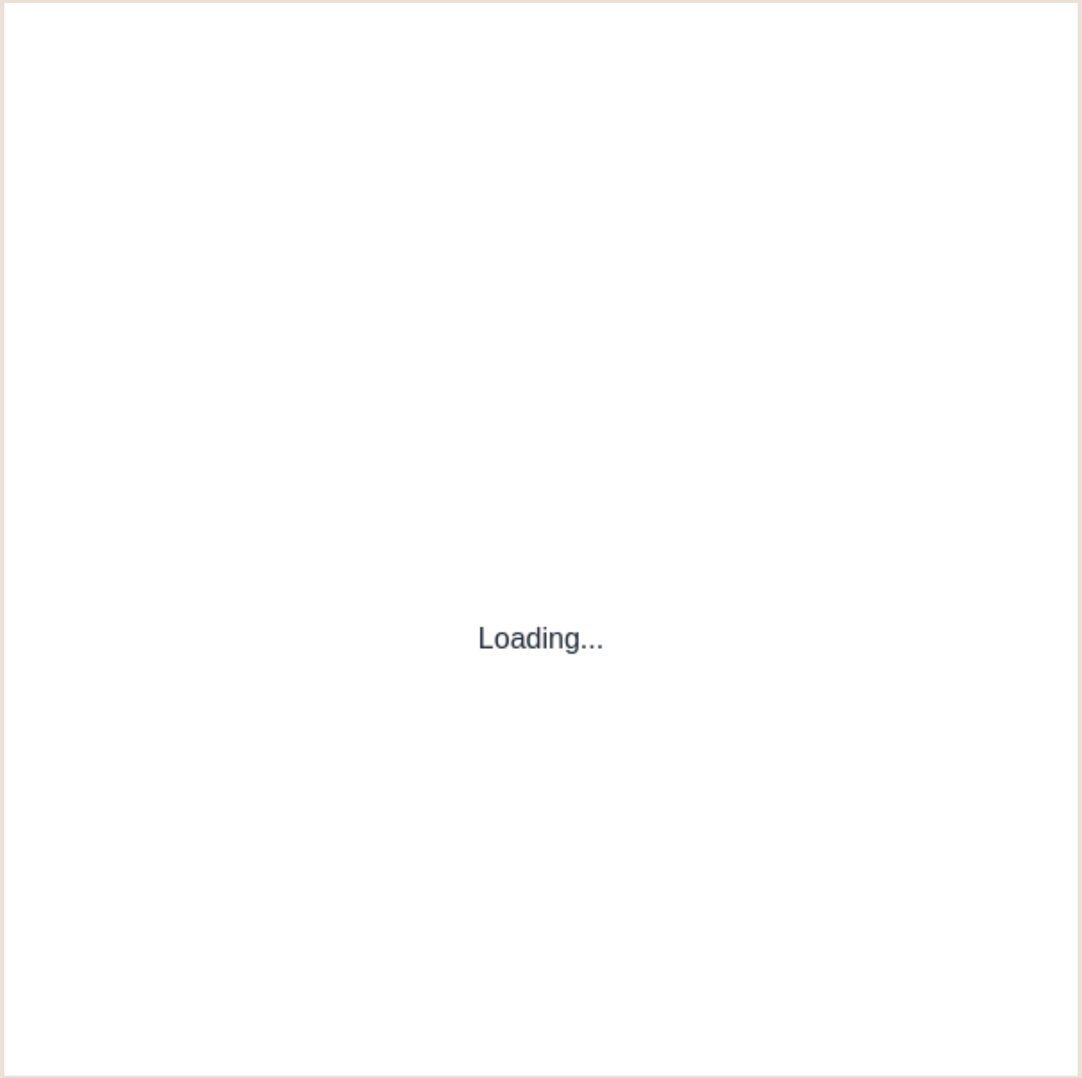
1x

Generated Image

GitHub Repository

https://github.com/sara-abdullah1/Final-Project-Tuwaiq-Generative-AI.git

Hugging Face



https://huggingface.co/spaces/SarahMarzouq/PoemGen

GitHub Repository

Nada-hs/**PoemGen**

1 Contributor 0 Issues 0 Stars 0 Forks

Nada-hs/PoemGen

Contribute to Nada-hs/PoemGen development by creating an account on GitHub.

GitHub

https://github.com/Nada-hs/PoemGen

Hugging Face

https://huggingface.co/spaces/NadaAljohani/PoemGen

Thank you