

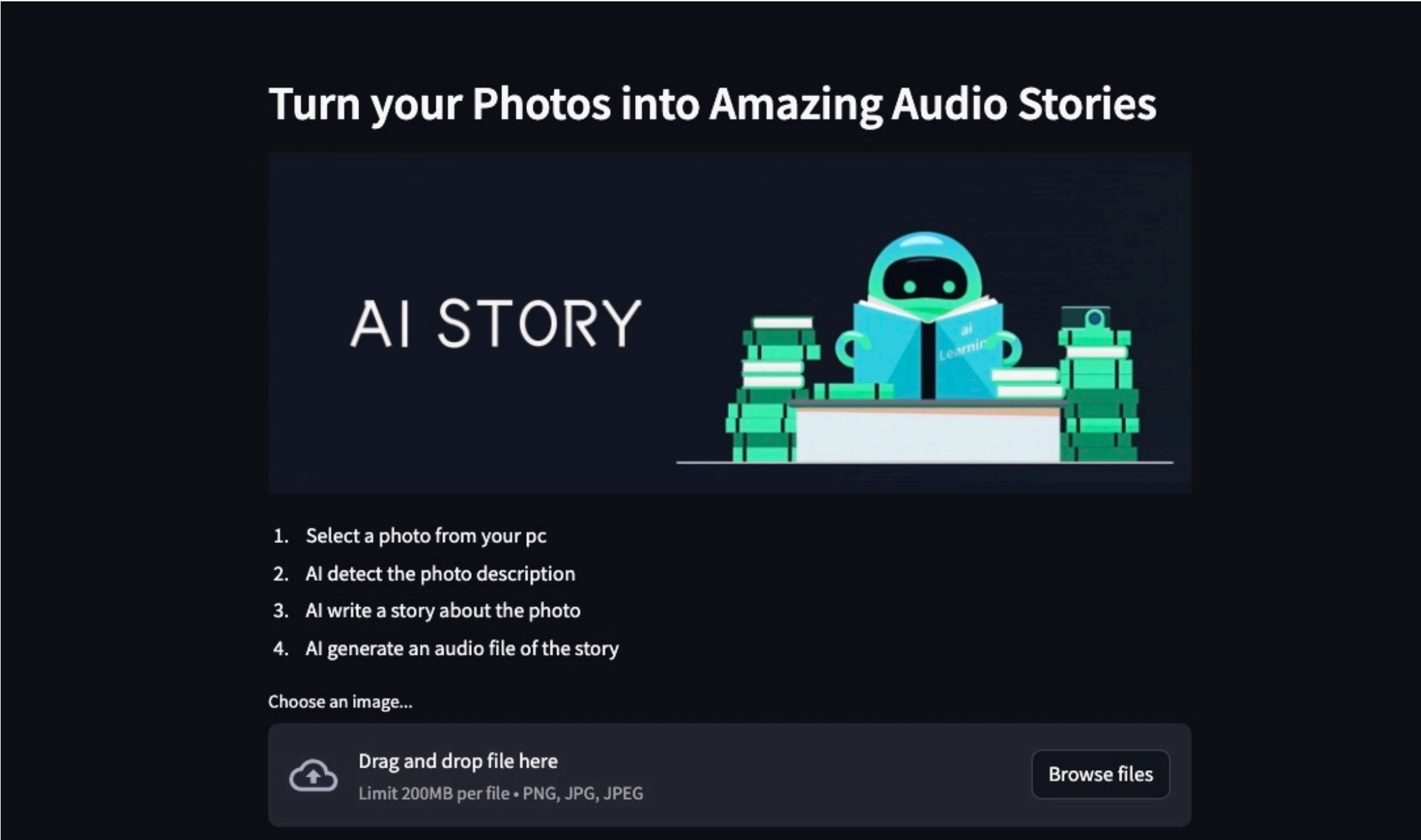
利用Streamlit 和 Hugging Face 创建免费AI故事机

使用 Python、Streamlit 和 Hugging Face 模型，构建无需 API 令牌的AI故事机，应用根据上传的图片创建音频故事。文章涉及的应用主要针对英文，可以借助应用学习AI如何看图写话。

当下生活中被人工智能模型和工具所包围，对于跟随人工智能发展的人来说，已经超负荷了，几乎每天都有各种新鲜的AI应用或者技术的出现。

但稍有不慎就觉得落后了，大公司和工具隐藏在黑盒子后面，没有途径去了解它们是如何运作的。在本文中，就来一起使用开源和免费工具探索人工智能的多模态方法，破解这个过程，把这个过程分解成简单的步骤，然后通过自学弄清楚如何去做。

本文将创建一个 `Python-Streamlit` 应用程序，将图片作为输入，通过 Hugging Face 模型识别照片的描述文本，根据它生成一个短故事，之后将根据该短篇故事生成音频。





Uploaded Image...

🤖 Generating Photo description

Photo Description
a boy sitting on top of a pile of books

🤖 Generating Photo Story

Photo Story

🤖 Generating Audio Story

▶ 0:00 🔊

Audio Story completed!

- GitHub: <https://github.com/QuintionTang/AI-Story>

第一步：创建虚拟环境

这是当前AI应用默认的流程了，从创建一个虚拟环境来开始一个新项目。首先创建一个全新的目录 `AI-Story` 并运行 `venv` 指令创建虚拟运行环境：

```
1 | mkdir AI-Story
2 | cd AI-Story
3 | python3 -m venv venv
```

激活虚拟环境：

```
1 | source venv/bin/activate  #for mac
2 | venv\Scripts\activate    #for windows users
```

第二步：安装所需的依赖项并获取 Hugging Face API Token

激活 `venv` 后，运行以下 `pip` 安装所需的软件包：

```
1 | pip install --upgrade pip
2 | pip install transformers  # 用于与LLM的互动
3 | pip install huggingface_hub # hugging face library
4 | pip install langchain
5 | pip install streamlit==1.24.0
6 | pip install python-dotenv
```

从上面可以看到，没有安装 `pytorch` 或 `tensorflow`：主要是因为这里将在免费的 Hugging Face 模型上使用 API 进行推理。为此，需要在 Hugging Face 上注册并创建 API 令牌（向LLMs发出 API 请求的个人授权密钥）。

在 Hugging Face 官方网站上注册账号并获取 API 令牌，[这里是说明](#)。

因此，需要在 Hugging Face 上创建一个帐户（如果还没有），然后[创建 API 令牌](#)。

记住！如果在向 API 发送请求时未提交 API 令牌，将无法在私有模型上运行推理。

在开始之前，先来创建一个 `config.py` 文件来存储配置变量，`config.py` 变量的参数来自于文件 `.env`。 `config.py` 的代码如下：

```
1 | from dotenv import load_dotenv
2 | import os
3 |
4 | load_dotenv()
5 |
6 | HUGGING_FACE_API_TOKEN=os.environ.get('HUGGING_FACE_API_TOKEN')
```

`.env` 的配置格式如下：

```
1 | HUGGING_FACE_API_TOKEN=" "
```

在主目录中创建一个新的 python 文件 `app.py`，先来验证依赖是否安装成功：

```
1 | # AI 推理库
2 | from huggingface_hub import InferenceClient
3 | from langchain import HuggingFaceHub
4 | import requests
5 |
6 | # 配置文件
7 | from config import (HUGGING_FACE_API_TOKEN)
8 |
9 | import os
10 | import datetime
```

保存并在激活 `venv` 的情况下从终端窗口运行：

```
1 | python3 app.py
```

如果什么没有任何输出和错误就意味着一切安装都成功了。

注意：导入 Langchain，因为 Hugging Face 尚不支持文本生成推理管道，需要 Langchain 来实现这个功能。

第三步：创建由图片生成文字的 AI 函数

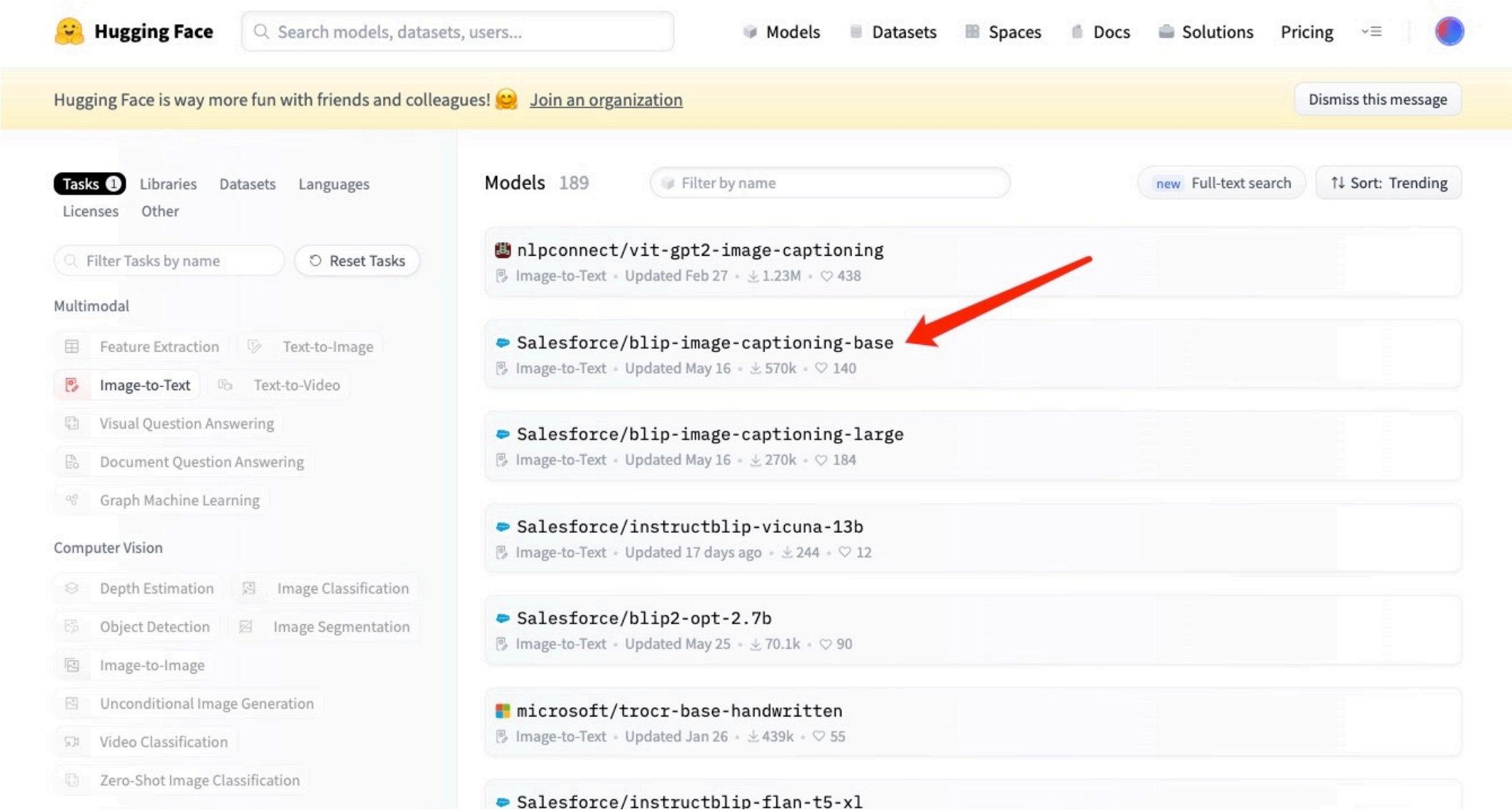
在 `app.py` 中，可以开始创建一些函数。将为每项任务创建一个函数：一个用于图像到文本、一个用于文本生成故事、最后一个用于故事文本到语音。

代码如下：



```
1 | hfApiToken = HUGGING_FACE_API_TOKEN
2 | # 仅 HuggingFace Hub 推论
3 | model_Image2Text = "Salesforce/blip-image-captioning-base"
```


使用变量 `hfApiToken` 来存储 Hugging Face 令牌，变量 `model_Image2Text` 任务相关的模型。


图像到文本任务位于 Hugging Face Multimodal 模型中。在 Hugging Face 的 Models 页面上，在左侧的 Multimodal 选择 Image-to-Text：在最受欢迎的模型中，以 `blip-base` 最知名。



单击它时，模型卡页面将打开，其中包含大量说明和快速启动代码。为了进行推理，需要遵循 API 指南的说明，仅更改模型名称：只需单击复制图标即可，如图所示：

**Hugging Face**

Models Datasets Spaces Docs Solutions Pricing 



Hugging Face is way more fun with friends and colleagues!  [Join an organization](#)

Dismiss this message

 Salesforce

blip-image-captioning-base 

 like 140

 Image-to-Text

 PyTorch

 TensorFlow

 Transformers

 blip

 text2text-generation

 image-captioning

 AutoTrain Compatible

 arxiv:2201.12086

 License: bsd-3-clause

Model card Files and versions Community 20

  Train  Deploy  Use in Transformers

 Edit model card

BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation

Model card for image captioning pretrained on COCO dataset - base architecture (with ViT base backbone).



Downloads last month
570,249



 **Hosted inference API** 

 Image-to-Text

Drag image file here or click to browse from your device

The model is loaded and running on Intel Xeon 3rd Gen Scalable CPU

 JSON Output 

 **Spaces using Salesforce/blip-image-captioning-base** 230

 microsoft/visual_chatgpt

 nielsr/comparing-captioning-models

 h2oai/h2ogpt-chatbot

 TencentARC/Caption-Anything

 Awiny/Image2Paragraph

 h2oai/h2ogpt-chatbot2

 editing-images/ledits

 RamAnanth1/visual-chatGPT

`imageToText` 函数现在有了一个模型，发送带有以下信息请求：

```
1 def imageToText(url):
2     from huggingface_hub import InferenceClient
3     client = InferenceClient(token=hfApiToken)
4     model_Image2Text = "Salesforce/blip-image-captioning-base"
5     # 来自huggingface.co/tasks
6     text = client.image_to_text(url,model=model_Image2Text)
7     print(text)
8     return text
```

函数 `imageToText` 将接受本地图像文件并返回描述该图像的文本。


文件 `app.py` 完整代码如下：

```
1 # AI 推理库
2 from huggingface_hub import InferenceClient
3 from langchain import HuggingFaceHub
4 import requests
5
6 # 配置文件
7 from config import (HUGGING_FACE_API_TOKEN)
8
9 import os
10 import datetime
11
12 hfApiToken = HUGGING_FACE_API_TOKEN
13 # 仅 HuggingFace Hub 推论
14 model_Image2Text = "Salesforce/blip-image-captioning-base"
15
16 def imageToText(url):
17     from huggingface_hub import InferenceClient
18     client = InferenceClient(token=hfApiToken)
19     # 来自huggingface.co/tasks
20     text = client.image_to_text(url,model=model_Image2Text)
```

```
21 | print(text)
22 | return text
```

下面就来运行一下这个函数:

```
1 | basetext = imageToText("./images/a-boy.jpg")
```

 **Hugging Face**

Models

Datasets

Spaces

Docs

Solutions

Pricing

Hugging Face is way more fun with friends and colleagues! 🥳 [Join an organization](#)

Dismiss this message

Salesforce/blip-image-captioning-base

like 140

Image-to-Text

PyTorch

TensorFlow

Transformers

blip

text2text-generation

image-captioning

AutoTrain Compatible

arxiv:2201.12086

License: bsd-3-clause

Model card

Files and versions

Community 20

Train

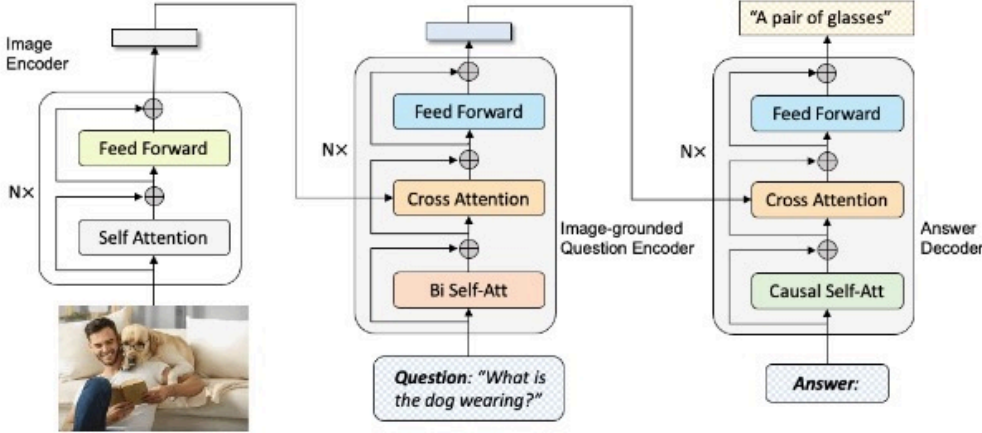
Deploy

Use in Transformers

Edit model card


BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation

Model card for image captioning pretrained on COCO dataset - base architecture (with ViT base backbone).



Downloads last month

570,249



Hosted inference API

Image-to-Text

Drag image file here or click to browse from your device

The model is loaded and running on Intel Xeon 3rd Gen Scalable CPU

JSON Output

Maximize

Spaces using Salesforce/blip-image-captioning-base 230

microsoft/visual_chatgpt

nielsr/comparing-captioning-models

h2oai/h2ogpt-chatbot

TencentARC/Caption-Anything

Awiny/Image2Paragraph

h2oai/h2ogpt-chatbot2

editing-images/ledits

RamAnanth1/visual-chatGPT

在激活 venv 下运行:

```
1 | python3 app.py
```

图片描述如下:

```
1 | a boy sitting on top of a pile of books
```

根据上述描述生成的故事如下:

```
1 | Once upon a time, there was a boy named Jack who loved to read. He would spend hours buried in a pile of books, lost in the worlds that only books could
2 | create.
3 |
4 | One day, Jack was sitting on top of the pile, lost in a book about dragons and knights. Suddenly, he felt a tug on his book. He looked up to see a small creature
5 | trying to get his attention.
6 |
7 | "What are you doing?" Jack asked.
8 |
9 | The creature looked up at him with big, round eyes. "I'm trying to get your attention," it said. "I'm a bookworm too. We're both in here, aren't we?"
10 |
11 | Jack smiled. "I suppose we are," he said. "What are you reading?"
12 |
13 | The creature looked down at the book in its hand. "This one," it said. "It's about a brave knight who saves a princess from a dragon."

Jack nodded. "I've been reading a lot of those lately," he said.
```


第四步：从文本函数创建 AI 生成的故事

imageToText 函数检索到的照片描述将是故事生成的起点。

使用 Hugging Face 模型进行文本生成推理并不是一件容易的事！首先，许多执行模型都禁用了 API；其次，文本生成推理根据选择的模型遵循不同的规则。

测试了其中的20个，最终决定选择 [togethercomputer/RedPajama-INCITE-Chat-3B-v1](#)，这是主要基于 OpenAssistant LLM 的模型之一。Open Assistant 是由 LAION 和世界各地有兴趣将该技术带给每个人的个人组织的一个项目。他们的口号是：

我们相信我们即将创造一场革命
正如 Stable Diffusion 改变了现代艺术的创作过程, 我们将透过对话式 AI 来改变世界。

将创建一个函数，使用 LangChain 作为文本生成推理的网关，指定一个类似于上面给出的提示。

```
1 | # Langchain 到 HuggingFace 的推论
2 | def LC_TextGeneration(model, basetext):
3 |     from langchain import PromptTemplate, LLMChain
4 |     os.environ["HUGGINGFACEHUB_API_TOKEN"] = hfApiToken
5 |     llm = HuggingFaceHub(repo_id=model , model_kwargs={"temperature":0.45,"min_length":30, "max_length":250})
6 |     print(f"Running repo: {model}")
7 |     print("Preparing template")
8 |     template = """<human>: write a very short story about {basetext}.
9 |     The story must be a one paragraph.
10 |    <bot>: """
11 |    prompt = PromptTemplate(template=template, input_variables=["basetext"])
12 |    llm_chain = LLMChain(prompt=prompt, llm=llm)
13 |    start = datetime.datetime.now()
14 |    print("Running chain...")
15 |    story = llm_chain.run(basetext)
16 |    stop = datetime.datetime.now()
17 |    elapsed = stop - start
18 |    print(f"Executed in {elapsed}")
19 |    print(story)
20 |    return story
```

LangChain 需要不同的方法来传递 HuggingFace API 令牌：使用 os.environ["HUGGINGFACEHUB_API_TOKEN"] 将其存储为环境变量。

该函数将接受模型（ RedPajama-INCITE-Chat-3B-v1 ）和基本文本（用于生成短篇故事的文本）作为参数。

正如所看到的，提示模板遵循快速入门部分的说明：仅添加基本文本变量以在基本指令中包含要完成的任务的详细信息：

```
1 | template = """<human>: write a very short story about {basetext}.
2 |     The story must be a one paragraph.
3 |    <bot>: """
```

为了更好的了解生成时间，有一些控制台打印指令（例如用于验证生成状态的小检查点）。现在函数已经准备好了，下面就来运行看下效果：

```
1 | basetext = imageToText("./images/a-boy.jpg")
2 | model_TextGeneration="togethercomputer/RedPajama-INCITE-Chat-3B-v1"
3 | mystory = LC_TextGeneration(model_TextGeneration, basetext)
4 | print("="*50)
5 | finalstory = mystory.split('\n\n')[0]
6 | print(finalstory)
```

运行后 AI 写的故事如下：可能已经注意到这个故事要长得多。这就是为什么将其分成几个段落，并且只采用第一段：对于一个小故事来说已经足够了。

```
1 | Once upon a time, there was a boy named Jack who loved to read. He would spend hours buried in a pile of books, lost in the worlds that only books could
2 | create.
3 |
4 | One day, Jack was sitting on top of the pile, lost in a book about dragons and knights. Suddenly, he felt a tug on his book. He looked up to see a small creature
5 | trying to get his attention.
6 |
7 | "What are you doing?" Jack asked.
8 |
```

9

10

11

12

13

The creature looked up at him with big, round eyes. "I'm trying to get your attention," it said. "I'm a bookworm too. We're both in here, aren't we?"

Jack smiled. "I suppose we are," he said. "What are you reading?"

The creature looked down at the book in its hand. "This one," it said. "It's about a brave knight who saves a princess from a dragon."

Jack nodded. "I've been reading a lot of those lately," he said.

第五步：创建从故事生成音频的函数

现在有了文本故事，接下来将使用文本转语音模型来为生成音频。在 Hugging Face Model 部分，向下滚动左侧面板选择 `text-to-speech` 选项过滤音频任务。

可以尝试任何一种流行的声音：选择 `espnet/kan-bayashi_ljspeech_vits`。

使用 `requests` 方法创建一个用于文本转语音生成的函数：将在 Header 中包含一个带有 hfApiToken 的 `f` 字符串（如果不包含，API 请求将被拒绝）。

第六步：使用 Streamlit 集成所有功能

可以在 [GitHub 存储库](#)中找到所有图像和音频文件以及最终代码。

Streamlit 是一个用于构建数据 Web 应用程序的库，无需了解任何前端技术（例如 HTML 和 CSS）。如果想了解更多信息，请查看此处[更多文档](#)。

创建一个新的文件 `main.py`，完整代码如下：

```
1  # 使用 Streamlit 进行 HuggingFace 推理的 Python 应用程序
2  # AI 推理库
3  from huggingface_hub import InferenceClient
4  from langchain import HuggingFaceHub
5  import requests
6  # 内部使用
7  import os
8  import datetime
9  import uuid
10 # STREAMLIT
11 import streamlit as st
12 # 配置文件
13 from config import (HUGGING_FACE_API_TOKEN)
14
15 hfApiToken = HUGGING_FACE_API_TOKEN
16 # 只有HuggingFace Hub 推理
17
18 model_TextGeneration="togethercomputer/RedPajama-INCITE-Chat-3B-v1"
19 model_Image2Text = "Salesforce/blip-image-captioning-base"
20 model_Text2Speech="espnet/kan-bayashi_ljspeech_vits"
21
22 def imageToText(url):
23     from huggingface_hub import InferenceClient
24     client = InferenceClient(token=hfApiToken)
25     model_Image2Text = "Salesforce/blip-image-captioning-base"
26     text = client.image_to_text(url,
27                               model=model_Image2Text)
28     print(text)
29     return text
30
31
32 def text2speech(text):
33     import requests
34     API_URL = "https://api-inference.huggingface.co/models/espnet/kan-bayashi_ljspeech_vits"
35     headers = {"Authorization": f"Bearer {hfApiToken}"}
36
37     payloads = {
38         "inputs" : "".join(text.split('\n\n'))
39     }
40     response = requests.post(API_URL, headers=headers, json=payloads)
41     with open('audiostory.flac', 'wb') as file:
```



```
42     file.write(response.content)
43
44
45 # Langchain 到 Hugging Face 的推理
46 def LC_TextGeneration(model, basetext):
47     from langchain import PromptTemplate, LLMChain
48     os.environ["HUGGINGFACEHUB_API_TOKEN"] = hfApiToken
49     llm = HuggingFaceHub(repo_id=model , model_kwargs={"temperature":0.45,"min_length":30, "max_length":250})
50     print(f"Running repo: {model}")
51     print("Preparing template")
52     template = """"<human>: write a very short story about {basetext}.
53     The story must be a one paragraph.
54     <bot>: """"
55     prompt = PromptTemplate(template=template, input_variables=["basetext"])
56     llm_chain = LLMChain(prompt=prompt, llm=llm)
57     start = datetime.datetime.now()
58     print("Running chain...")
59     story = llm_chain.run(basetext)
60     stop = datetime.datetime.now()
61     elapsed = stop - start
62     print(f"Executed in {elapsed}")
63     print(story)
64     return story
65
66 def generate_uuid():
67     return uuid.uuid4().hex
68
69 def main():
70
71     st.set_page_config(page_title="Your Photo Story Creatror App", page_icon='📷 ')
72
73     st.header("Turn your Photos into Amazing Audio Stories")
74     st.image('./assets/banner.png', use_column_width=True)
75     st.markdown("1. Select a photo from your pc\n 2. AI detect the photo description\n3. AI write a story about the photo\n4. AI generate an audio file of the
76 story")
77
78     image_file = st.file_uploader("Choose an image...", type=['png', 'jpg'] )
79     if image_file is not None:
80         print(image_file)
81         bytes_data = image_file.getvalue()
82         save_name = generate_uuid()
83         upload_path = f"./upload/{save_name}.jpg"
84         with open(upload_path, "wb") as file:
85             file.write(bytes_data)
86         st.image(image_file, caption="Uploaded Image...",
87             use_column_width=True)
88
89         st.warning("Generating Photo description", icon="🗣️")
90         basetext = imageToText(upload_path)
91         with st.expander("Photo Description"):
92             st.write(basetext)
93         st.warning("Generating Photo Story", icon="🗣️")
94         mystory = LC_TextGeneration(model_TextGeneration, basetext)
95         finalstory = mystory
96         print(f"#{50})
97         with st.expander("Photo Story"):
98             st.write(finalstory)
99         st.warning("Generating Audio Story", icon="🗣️")
100         text2speech(finalstory)
101
102
103         st.audio('audiostory.flac')
104         st.success("Audio Story completed!")
105
106
107 if __name__ == '__main__':
    main()
```

保存 python 文件并激活 venv，运行

应该看到如下效果：



完整代码在GitHub仓库：

- GitHub：<https://github.com/QuintionTang/AI-Story>

总结

通过文章的应用，发现开源语言模型的力量是强大的，庆幸生活在这个年代，时代不辜负。这只是一个简单的应用，其实可以对其进行扩展，使其成为学习英语的AI助理，陪练口语、看图写话、写作文等等。