vertex-ai-jupyterlab

File Edit View Run Kernel Git Tabs Settings Help

e2-medium

multimodal_retail_recommen

Code Execute Python 3 (ipykernel) (Local)

Filter files by name

/

Name

multimodal_retail_rec...
notebook_template.ip...
requirements.txt

[1]:
```
# Copyright 2023 Google LLC
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the license.
# You may obtain a copy of the License at
#
#     https://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
```

Multimodal retail recommendation: Using Gemini to recommend items based on images and image reasoning

Run in Colab | Run in Colab Enterprise | Open in Vertex AI Workbench | View on GitHub | Open in Cloud Skills Boost

Share to:

Author(s) Thu Ya Kyaw

Overview

For retail companies, recommendation systems improve customer experience and thus can increase sales.

This notebook shows how you can use the multimodal capabilities of Gemini 2.0 model to rapidly create a multimodal recommendation system out-of-the-box.

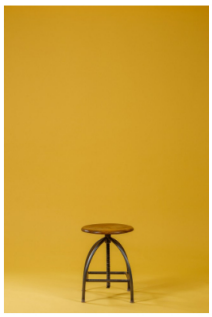
Scenario

The customer shows you their living room:

Customer photo

Below are four chair options that the customer is trying to decide between:

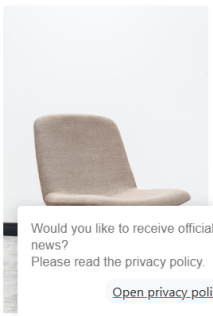
Chair 1



Chair 2

Chair 3

Chair 4



How can you use Gemini, a multimodal model, to help the customer choose the best option, and also explain why?

Objectives

Your main objective is to learn how to create a recommendation system that can provide both recommendations and explanations using a multimodal model: Gemini 2.0.

In this notebook, you will begin with a scene (e.g. a living room) and use the Gemini model to perform visual understanding. You will also investigate how the Gemini model can be used to recommend an item (e.g. a chair) from a list of furniture items as input.

By going through this notebook, you will learn:

- how to use the Gemini model to perform visual understanding

- how to take multimodality into consideration in prompting for the Gemini model
- how the Gemini model can be used to create retail recommendation applications out-of-the-box

Costs

This tutorial uses billable components of Google Cloud:

- Vertex AI

Learn about Vertex AI pricing and use the Pricing Calculator to generate a cost estimate based on your projected usage.

Getting Started

Install Gen AI SDK for Python

[2]:
```
%pip install --upgrade google-genai
```

```
Collecting google-genai
  Downloading google_genai-1.17.0-py3-none-any.whl.metadata (35 kB)
Requirement already satisfied: anyio<5.0.0,>=4.8.0 in /opt/conda/lib/python3.10/site-packages (from google-genai) (4.9.0)
Requirement already satisfied: google-auth<3.0.0,>=2.14.1 in /opt/conda/lib/python3.10/site-packages (from google-genai) (2.38.0)
Collecting httpx<1.0.0,>=0.28.1 (from google-genai)
  Downloading httpx-0.28.1-py3-none-any.whl.metadata (7.1 kB)
Requirement already satisfied: pydantic<3.0.0,>=2.0.0 in /opt/conda/lib/python3.10/site-packages (from google-genai) (2.11.0)
Requirement already satisfied: requests<3.0.0,>=2.28.1 in /opt/conda/lib/python3.10/site-packages (from google-genai) (2.32.3)
Requirement already satisfied: websockets<15.1.0,>=13.0.0 in /opt/conda/lib/python3.10/site-packages (from google-genai) (15.0.1)
Requirement already satisfied: typing-extensions<5.0.0,>=4.11.0 in /opt/conda/lib/python3.10/site-packages (from google-genai) (4.13.0)
Requirement already satisfied: exceptiongroup>=1.0.2 in /opt/conda/lib/python3.10/site-packages (from anyio<5.0.0,>=4.8.0->google-genai) (1.2.2)
Requirement already satisfied: idna>=2.8 in /opt/conda/lib/python3.10/site-packages (from anyio<5.0.0,>=4.8.0->google-genai) (3.10)
Requirement already satisfied: sniffio>=1.1 in /opt/conda/lib/python3.10/site-packages (from anyio<5.0.0,>=4.8.0->google-genai) (1.3.1)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /opt/conda/lib/python3.10/site-packages (from google-auth<3.0.0,>=2.14.1->google-genai) (5.5.2)
Requirement already satisfied: pyasn1-modules>=0.2.1 in /opt/conda/lib/python3.10/site-packages (from google-auth<3.0.0,>=2.14.1->google-genai) (0.4.2)
Requirement already satisfied: rsa<5,>=3.1.4 in /opt/conda/lib/python3.10/site-packages (from google-auth<3.0.0,>=2.14.1->google-genai) (4.9)
Requirement already satisfied: certifi in /opt/conda/lib/python3.10/site-packages (from httpx<1.0.0,>=0.28.1->google-gena
Collecting httpcore==1.* (from httpx<1.0.0,>=0.28.1->google-genai)
  Downloading httpcore-1.0.9-py3-none-any.whl.metadata (21 kB)
Collecting h11>=0.16 (from httpcore==1.*->httpx<1.0.0,>=0.28.1->google-genai)
  Downloading h11-0.16.0-py3-none-any.whl.metadata (8.3 kB)
Requirement already satisfied: annotated-types>=0.6.0 in /opt/conda/lib/python3.10/site-packages (from pydantic<3.0.0,>=2
Requirement already satisfied: pydantic-core==2.33.0 in /opt/conda/lib/python3.10/site-packages (from pydantic<3.0.0,>=2.
Requirement already satisfied: typing-inspection>=0.4.0 in /opt/conda/lib/python3.10/site-packages (from pydantic<3.0.0,>=2.0.0->google-genai) (0.4.0)
Requirement already satisfied: charset_normalizer<4,>=2 in /opt/conda/lib/python3.10/site-packages (from requests<3.0.0,>=2.28.1->google-genai) (3.4.1)
Requirement already satisfied: urllib3<3,>=1.21.1 in /opt/conda/lib/python3.10/site-packages (from requests<3.0.0,>=2.28.1->google-genai) (1.26.20)
Requirement already satisfied: pyasn1<0.7.0,>=0.6.1 in /opt/conda/lib/python3.10/site-packages (from pyasn1-modules>=0.2.1->google-auth<3.0.0,>=2.14.1->go
ogle-genai) (0.6.1)
Downloading google_genai-1.17.0-py3-none-any.whl (199 kB)
Downloading httpx-0.28.1-py3-none-any.whl (73 kB)
Downloading httpcore-1.0.9-py3-none-any.whl (78 kB)
Downloading h11-0.16.0-py3-none-any.whl (37 kB)
Installing collected packages: h11, httpcore, httpx, google-genai
  Attempting uninstall: h11
    Found existing installation: h11 0.14.0
    Uninstalling h11-0.14.0:
      Successfully uninstalled h11-0.14.0
Successfully installed google-genai-1.17.0 h11-0.16.0 httpcore-1.0.9 httpx-0.28.1
Note: you may need to restart the kernel to use updated packages.
```


Restart current runtime

To use the newly installed packages in this Jupyter runtime, you must restart the runtime. You can do this by running the cell below, which will restart the current kernel.

[3]:
```
# Restart kernel after installs so that your environment can access the new packages
import IPython

app = IPython.Application.instance()
app.kernel.do_shutdown(True)
```

[3]:
```
{'status': 'ok', 'restart': True}
```


⚠️ **The kernel is going to restart. Please wait until it is finished before continuing to the next step.** ⚠️

Authenticate your notebook environment (Colab only)

If you are running this notebook on Google Colab, run the following cell to authenticate your environment. This step is not required if you are using Vertex AI Workbench.

[1]:
```
import sys

# Additional authentication is required for Google Colab
if "google.colab" in sys.modules:
    # Authenticate user to Google Cloud
    from google.colab import auth

    auth.authenticate_user()
```

Define Google Cloud project information and initialize Vertex AI

Initialize the Vertex AI SDK for Python for your project:

[2]:
```
# Define project information
PROJECT_ID = "qwiklabs-gcp-01-18bfda7b4525"  # @param {type:"string"}
LOCATION = "us-west1"  # @param {type:"string"}

# Create the API client
from google import genai
client = genai.Client(vertexai=True, project=PROJECT_ID, location=LOCATION)
```


Import libraries

[3]:
```
from google.genai.types import GenerateContentConfig, Part
```

Using Gemini model

Gemini is a multimodal model that supports adding image and video in text or chat prompts for a text response.

Load Gemini model

[4]:
```
MODEL_ID = "gemini-2.0-flash"
```

Visual understanding with Gemini

Here you will ask the Gemini model to describe a room in details from its image. To do that you have to **combine text and image in a single prompt**.

[5]:
```
# urls for room images
room_image_url = "https://storage.googleapis.com/github-repo/img/gemini/retail-recommendations/rooms/spacejoy-c0JoR_-2x3E
room_image = Part.from_uri(file_uri=room_image_url, mime_type="image/jpeg")

prompt = "Describe what's visible in this room and the overall atmosphere:"
contents = [
    room_image,
    prompt,
]

responses = client.models.generate_content_stream(model=MODEL_ID, contents=contents)

print("\n-------Response--------")
for response in responses:
    print(response.text, end="")
```

```
-------Response--------
The room exudes a warm and inviting atmosphere, characterized by its neutral color palette and natural materials. The walls are painted in a soft off-whit
e hue, providing a serene backdrop for the carefully selected furniture and decor.

The focal point of the room is a light grey sofa adorned with an assortment of textured pillows in beige and cream tones, along with a cozy beige throw. A
bove the sofa, two round mirrors with straw frames add a touch of bohemian flair. A white marble coffee table with wooden legs sits in front of the sofa,
displaying a couple of books and a decorative bowl.

On the left side of the frame, a comfortable white armchair is paired with a matching ottoman and a fluffy sheepskin rug, creating a cozy reading nook. A
tall white vase filled with pampas grass stands beside the armchair, adding a natural element to the room.

To the right of the sofa, a floor lamp with a beige shade and a woven basket base provides additional lighting. The room is grounded by a large woven rug
in a light beige color, which anchors the furniture and adds a sense of warmth and texture. The wooden flooring peeks out from under the rug, adding anoth
er layer of natural material to the design.

Overall, the room feels peaceful, stylish, and well-curated, with a focus on comfort and natural elements.
```


Generating open recommendations based on built-in knowledge

Using the same image, you can ask the model to recommend **a piece of furniture** that would fit in it alongside with the description of the room.

Note that the model can choose **any furniture** to recommend in this case, and can do so from its only built-in knowledge.

[6]:
```
prompt1 = "Recommend a new piece of furniture for this room:"
prompt2 = "and explain the reason in detail"
contents = [prompt1, room_image, prompt2]

responses = client.models.generate_content_stream(model=MODEL_ID, contents=contents)

print("\n-------Response--------")
for response in responses:
    print(response.text, end="")
```

```
-------Response--------
Okay, given the existing furniture and the overall aesthetic of the room, I recommend adding **a long, low media console or sideboard** behind the sofa.

Here's the rationale:

*   **Balance:** The room feels a little weighted on the left side with the armchair, ottoman, and large vase. A console table behind the sofa will help b
alance the visual weight, anchoring the sofa and creating a more grounded feel.

*   **Functionality:** A console table offers practical benefits, such as:
    *   **Storage:** Provides concealed storage for items like blankets, books, games, or electronics accessories.
    *   **Display Space:** Creates a surface for displaying decorative objects, lamps, plants, or artwork, personalizing
    *   **Surface space:** Provides a great space for small snack plates and drinks when enjoying the seating area

*   **Aesthetic Cohesion:** To maintain the room's existing style, the console table should:
    *   **Be low and long:** This prevents it from overpowering the sofa and maintaining an open feel.
    *   **Have a neutral color palette:** A light wood tone, white, or a soft grey would complement the existing colors well.
    *   **Incorporate natural materials:** Rattan, woven details, or wood with a natural finish would enhance the room's bohemian-inspired vibe.

In summary, a low media console behind the sofa provides both functionality and aesthetic balance, enhancing the overall look and feel of the room.
```


In the next cell, you will ask the model to recommend **a type of chair** that would fit in it alongside with the description of the room.

Note that the model can choose **any type of chair** to recommend in this case.

[7]:
```
prompt1 = "Describe this room:"
prompt2 = "and recommend a type of chair that would fit in it"
contents = [prompt1, room_image, prompt2]

responses = client.models.generate_content_stream(model=MODEL_ID, contents=contents)

print("\n-------Response--------")
for response in responses:
    print(response.text, end="")
```

```
-------Response--------
The room is a cozy and stylish living space with a neutral, earthy color palette. The walls are a soft, off-white color. The floor is a warm wood with a j
ute rug covering most of the area.

Furniture includes:

*   A light grey sofa with several neutral-toned throw pillows and a textured throw blanket.
*   A round white marble coffee table with wooden legs.
*   A rounded, cream-colored armchair with a matching round cushion and a woven ottoman.
*   A tall white vase filled with pampas grass.
*   A floor lamp with a linen shade and a wooden base with a white accent, accompanied by a woven basket.
*   Two circular, woven mirrors are hung on the wall above the sofa.

The overall feel is warm, inviting, and contemporary with a touch of bohemian flair.

A good chair to add to this space would be a **rattan armchair** or a **mid-century modern wooden chair with a fabric seat** in a similar neutral tone. Th
ese would complement the existing aesthetic and add a touch of texture and visual interest.
```


Generating recommendations based on provided images

Instead of keeping the recommendation open, you can also provide a list of items for the model to choose from. Here you will download a few chair images and set them as

options for the Gemini model to recommend from. This is particularly useful for retail companies who want to provide recommendations to users based on the kind of room they have, and the available items that the store offers.

[8]:
```
# Download and display sample chairs
furniture_image_urls = [
    "https://storage.googleapis.com/github-repo/img/gemini/retail-recommendations/furnitures/cesar-couto-OB2F6CsMva8-unsplash.jpg",
    "https://storage.googleapis.com/github-repo/img/gemini/retail-recommendations/furnitures/daniil-silantev-1P6AnKDw6S8-unsplash.jpg",
    "https://storage.googleapis.com/github-repo/img/gemini/retail-recommendations/furnitures/ruslan-bardash-4kTbAMRAHtQ-unsplash.jpg",
    "https://storage.googleapis.com/github-repo/img/gemini/retail-recommendations/furnitures/scopic-ltd-NL1WwR4d3qU-unsplash.jpg",
]

# Load furniture images as Part Objects
furniture_images = [
    Part.from_uri(file_uri=url, mime_type="image/jpeg") for url in furniture_image_urls
]

# To recommend an item from a selection, you will need to label the item number within the prompt.
# That way you are providing the model with a way to reference each image as you pose a question.
# Labelling images within your prompt also help to reduce hallucinations and overall produce better results.
contents = [
    "Consider the following chairs:",
    "chair 1:",
    furniture_images[0],
    "chair 2:",
    furniture_images[1],
    "chair 3:",
    furniture_images[2],
    "chair 4:",
    furniture_images[3],
    "room:",
    room_image,
    "You are an interior designer. For each chair, explain whether it would be appropriate for the style of the room:",
]

responses = client.models.generate_content_stream(model=MODEL_ID, contents=contents)

print("\n-------Response--------")
for response in responses:
    print(response.text, end="")
```



```
-------Response--------
Okay, let's analyze each chair in relation to the provided room, which appears to have a neutral, modern-bohemian style.
e, natural textures (wood, woven elements), and a relaxed, comfortable vibe.

*   **Chair 1:** This chair would **NOT be appropriate**. It's a fairly industrial-looking stool with a metal frame and a
ignificantly with the room's softer aesthetic and natural materials. The stool feels out of place and lacks the warmth a
furnishings.

*   **Chair 2:** This chair is **highly appropriate**. Its tufted, upholstered design, and rounded shape complement the room's soft, inviting atmosphere.
The white color aligns with the room's neutral palette, and the style matches the existing armchair.

*   **Chair 3:** This chair would **NOT be appropriate**. This is a simple wooden stool. While wood is a theme in the room, this chair is too plain and ut
ilitarian, and doesn't fit the room's elegant and modern design.

*   **Chair 4:** This chair is **appropriate**. The chair has a more modern profile, but the neutral-toned upholstery and wood legs fit the room's color s
cheme and material palette. It could serve as a stylish accent chair without disrupting the room's overall aesthetic.
```


You can also return the responses in JSON format, to make it easier to plug recommendations into a recommendation system:

[9]:
```
contents = [
    "Consider the following chairs:",
    "chair 1:",
    furniture_images[0],
    "chair 2:",
    furniture_images[1],
    "chair 3:",
    furniture_images[2],
    "chair 4:",
    furniture_images[3],
    "room:",
    room_image,
    "You are an interior designer. Return in JSON, for each chair, whether it would fit in the room, with an explanation
]

responses = client.models.generate_content_stream(
    model=MODEL_ID,
    contents=contents,
    config=GenerateContentConfig(response_mime_type="application/json"),
)

print("\n-------Response--------")
for response in responses:
    print(response.text, end="")
```



```
-------Response--------
[
  {
    "chair_id": "chair 1",
    "fit": "no",
    "explanation": "Chair 1 is a bar stool and does not fit the aesthetic of the living room."
  },
  {
    "chair_id": "chair 2",
    "fit": "yes",
    "explanation": "Chair 2 is a cushioned chair that aligns with the style of the room."
  },
  {
    "chair_id": "chair 3",
    "fit": "no",
    "explanation": "Chair 3 is a bar stool and does not fit the aesthetic of the living room."
  },
  {
    "chair_id": "chair 4",
    "fit": "yes",
    "explanation": "Chair 4 is a cushioned chair that aligns with the style of the room."
  }
]
```


Conclusion

This notebook showed how you can easily build a multimodal recommendation system using Gemini for furniture, but you can also use the similar approach in:

- recommending clothes based on an occasion or an image of the venue
- recommending wallpaper based on the room and settings

You may also want to explore how you can build a RAG (retrieval-augmented generation) system where you retrieve relevant images from your store inventory to users who can they use Gemini to help identify the most ideal choice from the various options provided, and also explain the rationale to users.

[ ]: