Unstructured Data

16:33

卢浮部笃刻 关注

门窿智商测试题：以左下角的球为起点、右上角的球为终点，选择9个互相连接的球，并将球上的数字相加，能得出的最大值会是多少？

A:457 B:51 C:其他

算法比答案更重要

这是从一道所谓门萨智商测试题修改而来的新题目。我想知道解这种题目的算法到底是什么样的。
除了暴力遍历，还有其他做法吗？

说点什么... 10 3 22

You are moving from the bottom-left ball to the top-right ball. Each round you can move up or move right, so you will pass 9 balls in total. Summing up these 9 numbers, what is the maximum value you can get?

Write an algorithm for it!

What's Unstructured Data?

- Structured Data: Data that has a standardized format for efficient access by software and humans alike.
- Unstructured Data: Information that either does not have a pre-defined data model or is not organized in a pre-defined manner.
- Examples of Unstructured Data: Text data (emails, news articles, books, reviews), images, videos, audio data etc.

Processing Text Data in Python

Processing Text Data in Python

#1: Word Frequency

Word Frequency

Given a document, you may want to find out the most frequently used words in the document; this is called word frequency analysis. It can be easily done with Python; but please remove the stop words (i.e., words such as “the”, “it”, “mine”, “but”) from the list!

Word Frequency

Consider a text dataset which is available here:

https://ximarketing.github.io/data/LDAshort.txt

Ask AI to help you visualize the most frequent words used in the text document!

Processing Text Data in Python
#2: Topic Models

Topic Models

Topic modeling is an unsupervised learning algorithm. It takes a large number of documents (aka corpus) as input, and automatically generates a few topics from the text.

Question: What is a topic?

Topic Models

How does topic modeling work? The basic idea is, words of the same topic tend to appear simultaneously. For instance, in the topic that described “higher education”, some key words “university”, “library”, “examination”, “enrolment” often appear together.

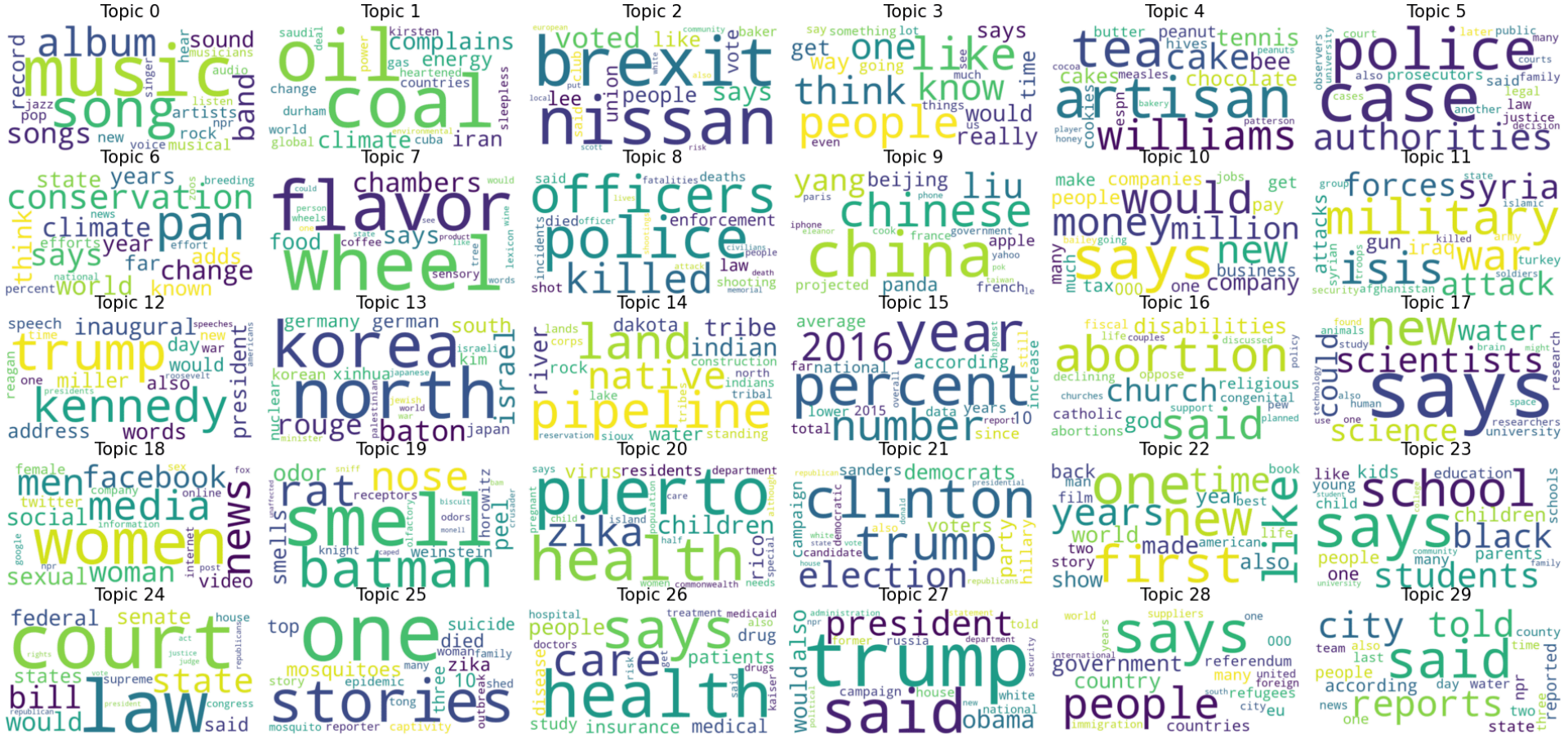
You don’t need to understand all these. Using topic models directly to get your results!

https://www.youtube.com/embed/p1I9Sa1lRvk

Topic Models

The most popular method for topic modeling is latent Dirichlet allocation (LDA). Let me show you an example based on LDA analysis of articles published by Pew Research Center, an American think tank.

I did not want to read any of their articles, but I do want to know what topics they are discussing.


Topic 0
album
music
song
record
sound
band
songs
Topic 1
oil
coal
complains
energy
climate
iran
Topic 2
brexit
nissan
voted
people
says
Topic 3
like
know
think
people
one
says
Topic 4
tea
cake
artisan
williams
chocolate
Topic 5
police
case
authorities
prosecutors
Topic 6
conservation
pan
climate
change
world
Topic 7
flavor
wheel
chambers
food
says
Topic 8
officers
police
killed
Topic 9
china
chinese
liu
beijing
yang
Topic 10
would
money
says
million
new
Topic 11
forcessyria
military
isis
war
attack
Topic 12
trump
kennedy
inaugural
speech
Topic 13
korea
north
germany
german
south
Topic 14
land
pipeline
native
tribe
indian
Topic 15
year
percent
number
2016
Topic 16
abortion
church
said
Topic 17
new
water
scientists
says
science
Topic 18
men
facebook
media
women
news
Topic 19
smell
batman
rat
nose
Topic 20
puerto
health
zika
Topic 21
clinton
trump
election
Topic 22
one
time
like
years
new
first
Topic 23
school
says
children
black
students
Topic 24
court
law
bill
state
Topic 25
one
stories
Topic 26
says
care
health
Topic 27
trump
said
president
Topic 28
says
people
government
Topic 29
city told
said
reports

Interpreting the topics:

Topic 0: music, song, album, band, ...

Topic 1: oil, coal, energy, climate, ...

Topic 9: china, chinese, beijing, panda, ...

Topic 18: woman, man, female, sexual, ...

Topic 21: trump, clinton, election, sanders, ...

Topic 27: trump, obama, president, russia, ...

Let's try to play with LDA!

The original dataset is too large and I am unable to share it with you. Instead, we have a shorter version of the data with it stored at the following location:

https://ximarketing.github.io/data/LDAshort.txt

Here, each line is an article and there are 1,000+ articles.

Ask AI to develop write code for you, and visualize your result (note: you need to install several modules).

```python
import nltk, requests
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from gensim import corpora
from gensim.models import LdaModel
from wordcloud import WordCloud
import matplotlib.pyplot as plt

# Download NLTK data files (only the first time)
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('punkt_tab')

file_url = 'https://ximarketing.github.io/data/LDAshort.txt'
response = requests.get(file_url)
if response.status_code == 200:
    content = response.text
documents = content.split('\n')
stop_words = set(stopwords.words('english'))

def preprocess(text):
    tokens = word_tokenize(text.lower())
    tokens = [word for word in tokens if word.isalnum()]
    tokens = [word for word in tokens if word not in stop_words]
    return tokens

processed_documents = [preprocess(doc) for doc in documents]

dictionary = corpora.Dictionary(processed_documents)
corpus = [dictionary.doc2bow(doc) for doc in processed_documents]
num_topics = 30
lda_model = LdaModel(corpus, num_topics=num_topics, id2word=dictionary, passes=15)
for idx, topic in lda_model.print_topics(-1):
    print(f'Topic: {idx} \nWords: {topic}\n')
```

Processing Text Data in Python
#3: Summarizing Text

A research team led by Professor Kevin Tsia, programme director of the Biomedical Engineering Programme under the Faculty of Engineering of the University of Hong Kong (HKU), has developed an artificial intelligence (AI)-driven imaging tool that enables speedy and precise diagnosis of cancer patients, greatly enhancing the effectiveness of their medical treatment.

In joint collaborations with HKU's Li Ka Shing Faculty of Medicine (HKUMed) and Queen Mary Hospital, the team headed by Professor Tsia, has successfully demonstrated the effective use of their latest generative AI method, the Cyto-Morphology Adversarial Distillation (CytoMAD), on lung cancer patients as well as drug tests. Combined with their proprietary microfluidic technology, CytoMAD allows fast and cost-effective 'label-free' imaging of human cells to help clinicians assess the patients' tumour at the precision of individual cells, and also determine whether the patients have the risk of metastasis.

CytoMAD uses AI to automatically correct cell imaging inconsistencies, enhance cell images, and extract previously undetectable information from cell images. Such all-round capability in CytoMAD ensures accurate and reliable downstream data analysis and diagnosis. CytoMAD's capabilities have the potential to revolutionize cell imaging for meaningful analysis of cell properties and related health and disease information.

TOP

Here is a lengthy article about HKU research.

What does it talk about?

Text Summarization

Imagine that you want to understand the meaning of the text, but you do not really want to read the whole paragraphs to grasp its meaning. You just need a short summary of the text. This can be done with text summarization. Here is my output:

a team led by Professor Kevin Tsia has developed an artificial intelligence-driven imaging tool that enables speedy and precise diagnosis of cancer patients .

Abstractive Summarization Using Transformers

```
from transformers import pipeline
# Load the summarization pipeline
summarizer = pipeline("summarization")
# Sample text
text = """
Input your text here
"""
# Generate a summary
summary = summarizer(text, max_length=40, min_length=15,
do_sample=False)
print(summary[0]['summary_text'])
```

Processing Text Data in Python
#4: Using APIs

This will be covered later in the class.

Processing Images in Python
#1: Display an Image

Image I/O and display

```
 1 from PIL import Image, ImageFont, ImageDraw
 2 from io import BytesIO
 3 import requests
 4 import matplotlib.pylab as plt
 5 import PIL.ImageStat as stat
 6 url = "https://ximarketing.github.io/data/mainbuilding.jpg"
 7 response = requests.get(url)
 8 image = Image.open(BytesIO(response.content))
 9 print(image.width, image.height, image.mode, image.format)
10 image.show()
```

If you are reading image from a local folder,
you can replace line 6 with your path:

```
1 image = Image.open("../mainbuilding.jpg")
```

Processing Images in Python
#2: Color image to grayscale

A grayscale image only has two colors, black and white, and each pixel is given by a value $0 \leq x_i \leq 255$. When x_i is close to 0, the pixel is blacker, and when x_i is closer to 255, the pixel is whiter. We can use a simple formula to covert color image:

$$x_i = 0.299 \cdot r_i + 0.587 \cdot g_i + 0.114 \cdot b_i.$$

Color image to gray scale

```
from PIL import Image
import requests
from io import BytesIO
url = "https://ximarketing.github.io/data/mainbuilding.jpg"
response = requests.get(url)
img = Image.open(BytesIO(response.content))
gray_image = img.convert('L')
gray_image.save('gray_image.jpg')
gray_image.show()
```

How about adding colors to grayscale images?

This is much harder because we lose a lot of information when creating grayscale images: Given x_i, there are many combinations (r_i, g_i, b_i) that satisfy

$$x_i = 0.299 \cdot r_i + 0.587 \cdot g_i + 0.114 \cdot b_i.$$

Processing Images in Python
#3: Adjusting Brightness

How about adjusting brightness of images?

When the color of a pixel is $(0, 0, 0)$, it is black.

When the color of a pixel is $(255, 255, 255)$, it is white.

Suppose that the color of a pixel is (r, g, b). If you move the color toward $(0, 0, 0)$, it becomes darker. If you move the color toward $(255, 255, 255)$, it becomes brighter.

This makes your image brighter.

```
1  from PIL import Image, ImageEnhance
2  import requests
3  from io import BytesIO
4  # URL of the image
5  url = 'https://ximarketing.github.io/data/mainbuilding.jpg'
6  # Fetch the image from the URL
7  response = requests.get(url)
8  img = Image.open(BytesIO(response.content))
9  image = img.point(lambda x: (255+x)/2)
10 image.show()
```

If you want darker, replace line 9 with:

```
1 image = img.point(lambda x: x/2)
```

Exercise: Develop an APP with the following functions

- The user can upload an image from his or her folder.
- There is a grayscale button. When a user clicks on the button, it converts the original image into grayscale and displays the grayscale image on a display panel to the user.
- Similarly, there is a brighter button, which makes the image brighter, and a darker button, which makes the image darker, and display the new image.

Processing Images in Python
#4: Changing colors

Making the Image Red

```
from PIL import Image, ImageEnhance
import requests
from io import BytesIO
# URL of the image
url = 'https://ximarketing.github.io/data/mainbuilding.jpg'
# Fetch the image from the URL
response = requests.get(url)
image = Image.open(BytesIO(response.content))
for i in range(image.width):
    for j in range(image.height):
        pixel = (image.getpixel((i, j)))
        r = int((pixel[0] + 255)/2)
        g = pixel[1]
        b = pixel[2]
        image.putpixel((i, j), (r, g, b))
image.show()
```

Negating the Image

```
from PIL import Image, ImageEnhance
import requests
from io import BytesIO
# URL of the image
url = 'https://ximarketing.github.io/data/mainbuilding.jpg'
# Fetch the image from the URL
response = requests.get(url)
image = Image.open(BytesIO(response.content))
for i in range(image.width):
    for j in range(image.height):
        pixel = (image.getpixel((i, j)))
        r = 255 - pixel[0]
        g = 255 - pixel[1]
        b = 255 - pixel[2]
        image.putpixel((i, j), (r, g, b))
image.show()
```

Processing Images in Python
#5: Color Channels

Separating colors in an image

```
from PIL import Image, ImageFont, ImageDraw
from io import BytesIO
import matplotlib.pylab as plt
import requests
url = "https://ximarketing.github.io/data/parrot.jpg"
response = requests.get(url)
image = Image.open(BytesIO(response.content))
ch_r, ch_g, ch_b = image.split()
# split the RGB image into 3 channels: R, G and B
plt.imshow(ch_r, cmap = plt.cm.Reds)
plt.show()
plt.imshow(ch_g, cmap = plt.cm.Greens)
plt.show()
plt.imshow(ch_b, cmap = plt.cm.Blues)
plt.show()
```

Changing the color of your parrot!

```
1  from PIL import Image, ImageFont, ImageDraw
2  from io import BytesIO
3  import matplotlib.pylab as plt
4  import requests
5  url = "https://ximarketing.github.io/data/parrot.jpg"
6  response = requests.get(url)
7  image = Image.open(BytesIO(response.content))
8  ch_r, ch_g, ch_b = image.split()
9  # split the RGB image into 3 channels: R, G and B
10 image = Image.merge('RGB', (ch_b, ch_g, ch_r))
11 image.show()
```

Processing Images in Python
#6: Compare Images

Discover the differences between two images:

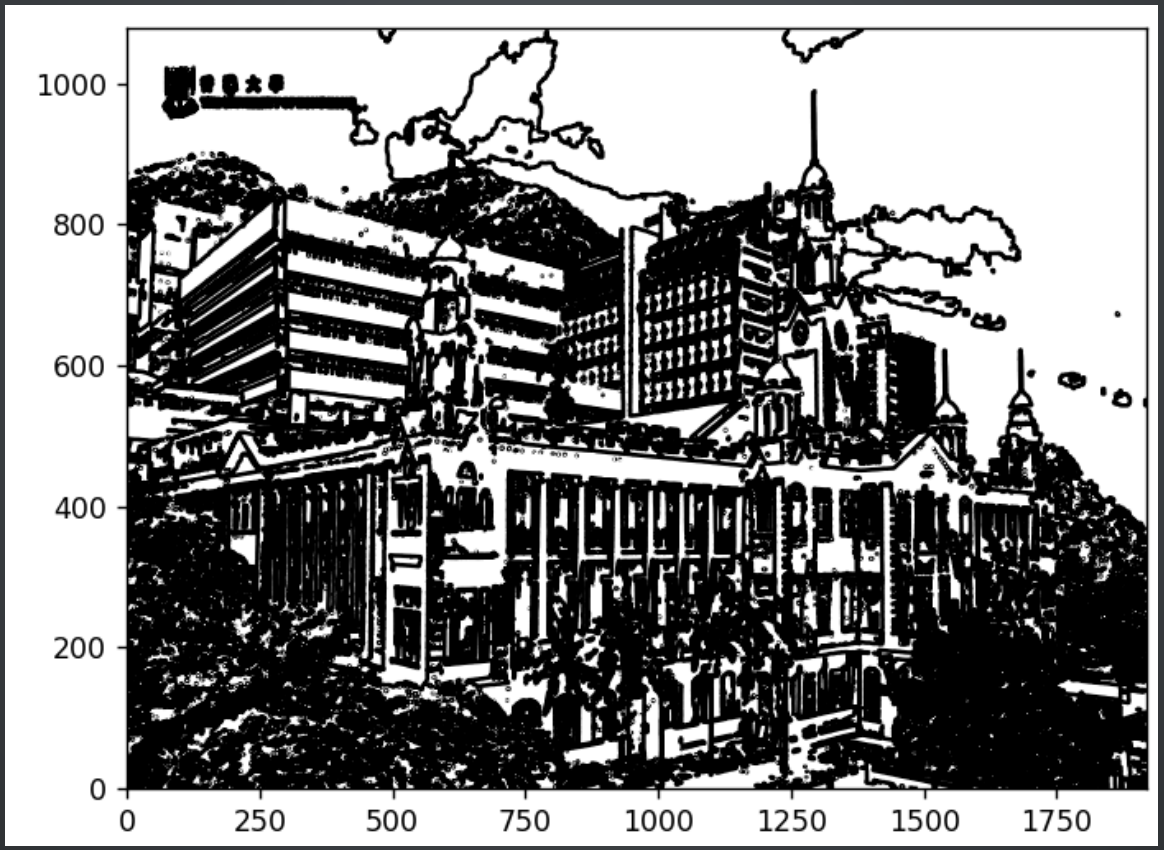
Discover the differences between two images:

```
from PIL.ImageChops import difference
from PIL import Image
from io import BytesIO
import requests
url = "https://ximarketing.github.io/data/????.jpeg"
response = requests.get(url)
image1 = Image.open(BytesIO(response.content))
url = "https://ximarketing.github.io/data/????.jpeg"
response = requests.get(url)
image2 = Image.open(BytesIO(response.content))
image = difference(image1, image2)
image.show()
```

Question: What are the applications of image comparison?

Processing Images in Python
#7: Contour Line

How to turn your image into the following one?



Drawing Contour Line of an Image

A contour line for an image is a curve connecting all of the pixels where they have the grayscale value. We need to first convert an image into grayscale and then connect the pixels with the same grayscale intensity.

Drawing Contour Line of an Image

```
from PIL import Image
from io import BytesIO
import requests
import matplotlib.pylab as plt
import numpy as np
url = "https://ximarketing.github.io/data/trump2.jpg"
response = requests.get(url)
image = Image.open(BytesIO(response.content))
image = image.convert('L')
plt.contour(np.flipud(image), colors ='k',
            levels = np.logspace(-15, 15, 100))
plt.show()
```

Processing Images in Python
#8: Convolution

What is convolution?

Convolution is an operation that operates on two images, one being an input image and the other one being a mask (also called the kernel) as a filter on the input image, producing an output image. It works by determining the value of a central pixel by adding the weighted values of all of its neighbors together to compute the new value of the pixel in the output image.

https://www.youtube.com/embed/yb2tPt0QVPY?enablejsapi=1

Why convolve an image?

Convolution applies a general-purpose filter effect on the input image. This is done in order to achieve various effects with appropriate kernels on an image, such as smoothing, sharpening, and embossing, and in operations such as edge detection.

Edge Detection

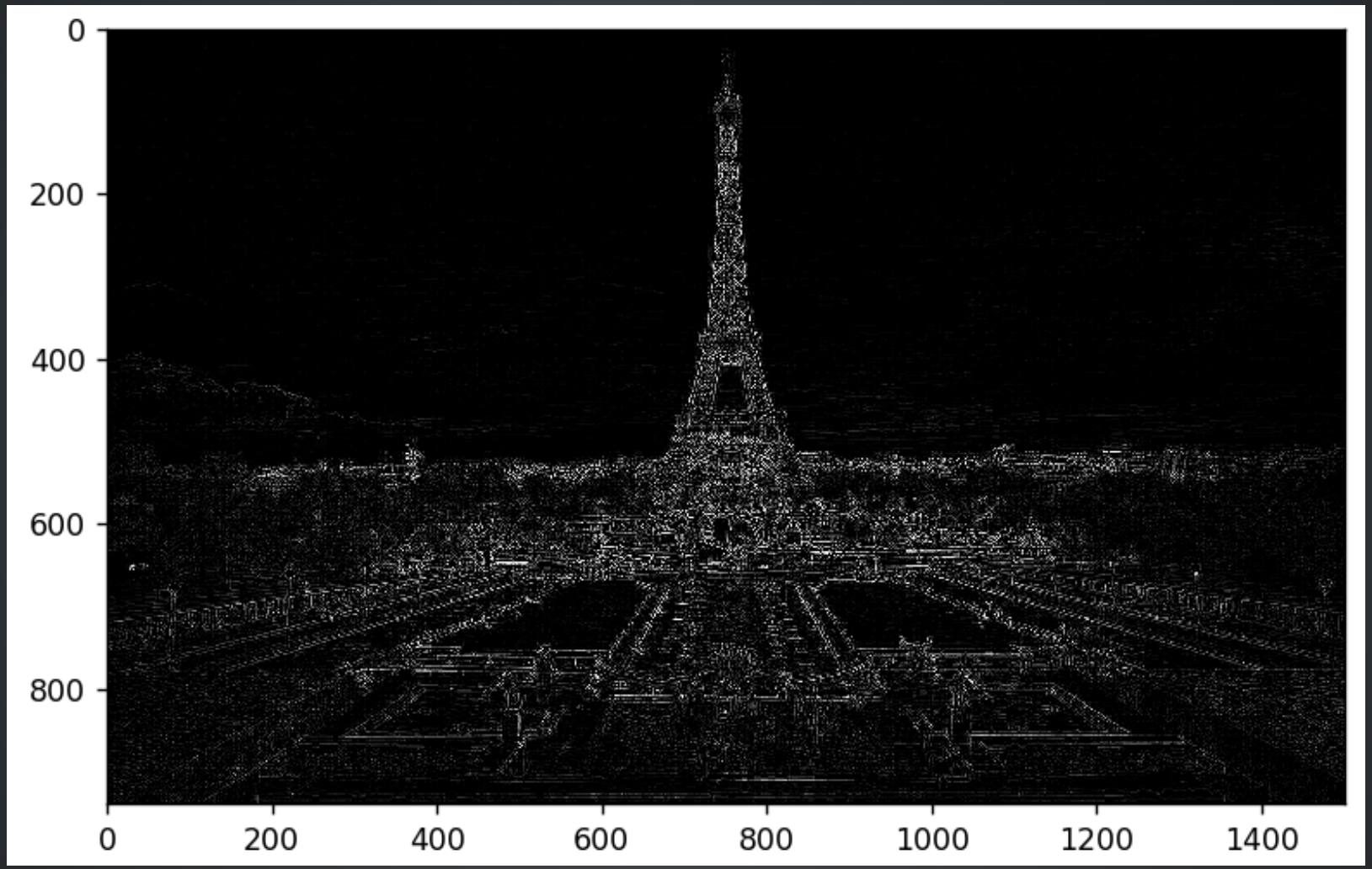
The following Laplace kernel is famous for edge detection. When applying the kernel to convolve images, you will get the edges of the original image.

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Let's try the following code:

The convolve2d function allows you to convolve a 2D image.

```
1  from skimage import io
2  from skimage.io import imshow
3  from scipy import signal
4  from skimage.color import rgb2gray
5  import matplotlib.pylab as pylab
6  import numpy as np
7  url = 'https://ximarketing.github.io/data/effiel.jpg'
8  image = rgb2gray(io.imread(url)).astype(float)
9  edge_laplace_kernel = np.array([[0,1,0],[1,-4,1],[0,1,0]])
10 image_edges = np.clip(signal.convolve2d(image, edge_laplace_kernel), 0, 1)
11 imshow(image_edges)
12 pylab.show()
```


0
200
400
600
800
0
200
400
600
800
1000
1200
1400

Emboss an Image ("图像浮雕")

The following emboss kernel will be used. When applying the kernel to convolve images, you will get the embossed verision of the original image.

$$\begin{bmatrix} -2 & -1 & 0 \\ -1 & 1 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$

Let's try the following code:

```
from skimage import io
from skimage.io import imshow
from scipy import signal
import matplotlib.pylab as pylab
import numpy as np
url = 'https://ximarketing.github.io/data/effiel.jpg'
image = io.imread(url)/255 #
emboss_kernel = np.array([[-2,-1,0],[-1,1,1],[0,1,2]])
image_embossed = np.ones(image.shape)
for i in range(3):
    image_embossed[...,i] = np.clip(
        signal.convolve2d(image[...,i], emboss_kernel,
                          mode='same', boundary="symm"),0,1)
imshow(image_embossed)
pylab.show()
```

More Kernels...

Kernel for Sharpening ("锐化")

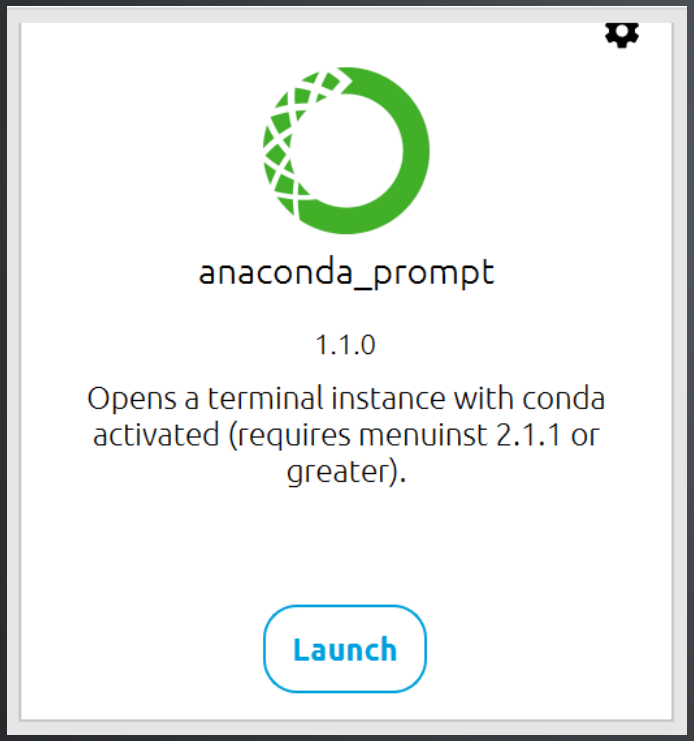
$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Box Blur Kernel

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$


0
200
400
600
800
0
200
400
600
800
1000
1200
1400

Additional: Change Background Color



Using prompt to install packages:

pip install rembg

pip install rembg[cpu]

Remove Photo Background

```
from rembg import remove
from skimage import io
from skimage.io import imshow
import matplotlib.pyplot as plt
url = "https://whyy.org/wp-content/uploads/2025/01/donald-trump-2025-01-08-768x512.jpg"
image = io.imread(url)
no_bg_img = remove(image)
imshow(no_bg_img)
plt.show()
```

Replace Backrgound Color

```
from rembg import remove
from skimage import io
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt
from skimage.io import imshow
def certphoto(fore_image, bgcolor):
    base_image = Image.new("RGB", fore_image.size, bgcolor)
    fore_arr = np.array(fore_image)
    scope_map = fore_arr[:,:, -1] / 255
    scope_map = scope_map[:,:, np.newaxis]
    scope_map = np.repeat(scope_map, repeats=3, axis=2)
    res_image = np.multiply(scope_map, fore_arr[:,:,:3]) + \
                np.multiply((1 - scope_map), np.array(base_image))
    return Image.fromarray(np.uint8(res_image))
url = "https://whyy.org/wp-content/uploads/2025/01/donald-trump-2025-01-08-768x512.jpg"
image = io.imread(url)
no_bg_img = remove(image)
fore_image_pil = Image.fromarray(no_bg_img)
result_image = certphoto(fore_image_pil, (0, 128, 255))
plt.imshow(result_image)
plt.show()
```

An APP

An APP (with camera capture)

Group Project (20%)

3 + 1 Options

1. Develop a gaming APP (desktop or Android) which demonstrates and solves a puzzle game. It features some algorithms that we learned in class (e.g., Hanoi tower).
2. Scrape some data yourself (to be covered later) and perform some analysis of the data. In this case, an APP is not required.
3. Develop an image APP (desktop or Android) which allows users to upload images and perform some operations on the image.
4. Other ideas: Discuss with me first.

Submission

1. Python code for desktop apps / apk file for Adriod apps.
2. Your entire chat history with AI.
3. A user manual of the APP (for options 1,3) or a report of your findings (for option 2).
4. Supplementary data files, if needed.

Deadline: 23:59 PM, May 1, 2026

Next Class

Each group demonstrates the APP you have developed for the previous assignment [data analysis and visualization] to the class, followed by a peer evaluation.

1~2 minutes for each group demonstration.