# -\*- coding: utf-8 -\*-

"""

Created on Sun Dec 12 13:00:19 2021

@author: MontalJ

"""

from pickle import load

from numpy import argmax

from keras.preprocessing.sequence import pad\_sequences

from keras.applications.vgg16 import VGG16

from keras.preprocessing.image import load\_img

from keras.preprocessing.image import img\_to\_array

from keras.applications.vgg16 import preprocess\_input

from keras.models import Model

from keras.models import load\_model

from gtts import gTTS

import os

import time

# extract features from each photo in the directory

def extract\_features(filename):

# load the model

model = VGG16()

# re-structure the model

model = Model(inputs=model.inputs, outputs=model.layers[-2].output)

# load the photo

image = load\_img(filename, target\_size=(224, 224))

# convert the image pixels to a numpy array

image = img\_to\_array(image)

# reshape data for the model

image = image.reshape((1, image.shape[0], image.shape[1], image.shape[2]))

# prepare the image for the VGG model

image = preprocess\_input(image)

# get features

feature = model.predict(image, verbose=0)

return feature

# map an integer to a word

def word\_for\_id(integer, tokenizer):

for word, index in tokenizer.word\_index.items():

if index == integer:

return word

return None

# generate a description for an image

def generate\_desc(model, tokenizer, photo, max\_length):

# seed the generation process

in\_text = 'startseq'

# iterate over the whole length of the sequence

for i in range(max\_length):

# integer encode input sequence

sequence = tokenizer.texts\_to\_sequences([in\_text])[0]

# pad input

sequence = pad\_sequences([sequence], maxlen=max\_length)

# predict next word

yhat = model.predict([photo,sequence], verbose=0)

# convert probability to integer

yhat = argmax(yhat)

# map integer to word

word = word\_for\_id(yhat, tokenizer)

# stop if we cannot map the word

if word is None:

break

# append as input for generating the next word

in\_text += ' ' + word

# stop if we predict the end of the sequence

if word == 'endseq':

break

return in\_text

# load the tokenizer

tokenizer = load(open('tokenizer.pkl', 'rb'))

# pre-define the max sequence length (from training)

max\_length = 34

# load the model

# model = load\_model('model-ep002-loss3.245-val\_loss3.612.h5')

model = load\_model('model-ep004-loss3.536-val\_loss3.876.h5')

# load and prepare the photograph

image\_name = 'example10.jpg'

photo = extract\_features(image\_name)

# generate description

description = generate\_desc(model, tokenizer, photo, max\_length)

# Display the image

import matplotlib.pyplot as plt

import matplotlib.image as mpimg

img = mpimg.imread(image\_name)

imgplot = plt.imshow(img)

plt.show()

# The text that you want to convert to audio

mytext = description.replace("startseq", "")

mytext = mytext.replace("endseq", "")

mytext = mytext.strip()

mytext = mytext.upper()

print('\n \nEnglish')

print(mytext)

# Language in which you want to convert

language = 'en'

# Passing the text and language to the engine,

# here we have marked slow=False. Which tells

# the module that the converted audio should

# have a high speed

myobj = gTTS(text = mytext, lang = language, slow = True)

# Saving the converted audio in a mp3 file named

# welcome

myobj.save(image\_name + "english.mp3")

time.sleep(3)

# Playing the converted file

os.system(image\_name + "english.mp3")

time.sleep(10)

from translate import Translator

translator= Translator(to\_lang = "zh")

translation = translator.translate(mytext)

print('\nMainland China')

print(translation)

myobj = gTTS(text = translation, lang = 'zh-CN', slow = True)

myobj.save(image\_name + "Main\_China.mp3")

time.sleep(3)

os.system(image\_name + "Main\_China.mp3")

# img\_viewer.py

import PySimpleGUI as sg

import os.path

# First the window layout in 2 columns

file\_list\_column = [

[

sg.Text("Image Folder"),

sg.In(size=(25, 1), enable\_events=True, key="-FOLDER-"),

sg.FolderBrowse(),

],

[

sg.Listbox(

values=[], enable\_events=True, size=(40, 20), key="-FILE LIST-"

)

],

]

# For now will only show the name of the file that was chosen

image\_viewer\_column = [

[sg.Text("Choose an image from list on left:")],

[sg.Text(size=(40, 1), key="-TOUT-")],

[sg.Image(key="-IMAGE-")],

]

# ----- Full layout -----

layout = [

[

sg.Column(file\_list\_column),

sg.VSeperator(),

sg.Column(image\_viewer\_column),

]

]

window = sg.Window("Image Viewer", layout)

# Run the Event Loop

while True:

event, values = window.read()

if event == "Exit" or event == sg.WIN\_CLOSED:

break

# Folder name was filled in, make a list of files in the folder

if event == "-FOLDER-":

folder = values["-FOLDER-"]

try:

# Get list of files in folder

file\_list = os.listdir(folder)

except:

file\_list = []

fnames = [

f

for f in file\_list

if os.path.isfile(os.path.join(folder, f))

and f.lower().endswith((".png", ".jpg"))

]

window["-FILE LIST-"].update(fnames)

elif event == "-FILE LIST-": # A file was chosen from the listbox

try:

filename = os.path.join(

values["-FOLDER-"], values["-FILE LIST-"][0]

)

window["-TOUT-"].update(filename)

window["-IMAGE-"].update(filename=filename)

except:

pass

window.close()