3. Jean Valverde Zamora., Código Orcid

UNIVERSIDAD PRIVADA DE TACNA

PRÁCTICA: "Sistema experto reconocimiento de imagenes"

Escuela profesional de ingeniería de sistemas

SI-881, Inteligencia Artificial

1. Anthony Cano Sucso, 0009-0003-1188-4526

2. Jose Luis Jarro C., Código Orcid

Docente:
Dr. Oscar J. Jimenez Flores

Orcid:
0000-0002-7981-8467

Semestre:
UPT-EPIS, 2024-I

Resumen: Los sistemas expertos son herramientas poderosas para la automatización de tareas basadas en el conocimiento en campos como la medicina, la ingeniería, la gestión empresarial y muchos otros, y han sido objeto de investigación y desarrollo durante décadas.

Palabras clave: Toma de decisiones, Base de conocimiento, Motor de inferencia .

1. Introducción

En el vasto universo de la visión por computadora, la detección de imágenes juega un papel crucial en una variedad de aplicaciones, desde la seguridad y vigilancia hasta el diagnóstico médico y la automatización industrial. Sin embargo, la tarea de identificar y clasificar objetos en imágenes puede ser desafiante debido a la variabilidad en la apariencia, iluminación y condiciones de captura.
En este contexto, la implementación de un sistema experto para la detección de imágenes se presenta como una solución innovadora y eficiente. Este sistema aprovecha el conocimiento experto en el procesamiento de imágenes y la identificación de patrones para realizar tareas como la detección de objetos, reconocimiento facial, clasificación de imágenes y más, con una precisión y velocidad sin precedentes.

2. Procedimiento

Primero importamos lo necesario

```
from ctypes import sizeof
from token import LEFTSHIFT
from logging import RootLogger
from operator import length_hint
from select import select
from tkinter import *
from tkinter import filedialog as fd
import shutil
import copy
import os
import tkinter
from turtle import width
from PIL import ImageTk,Image
import numpy as np
import time
import matplotlib.pyplot as plt
import matplotlib
matplotlib.use('Agg')
import threading
import os
import random
matplotlib.use('TkAgg')
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
```

Definimos las clases necesarias para nuestro sistema:

```
class graph_frame(Frame):
    def __init__(self):
        Frame.__init__(self,root)

    def add_graph(self,fig):
```

Veremos tambien nuestra clase de clasificacion:

una funcion donde se cargaran las caracteristicas:

```
def load_characteristics(self):
    self.characteristics = []
    self.characteristics.append("nombre")
    self.characteristics.append("tamaño")
    self.characteristics.append("color")
    self.characteristics.append("habitat")
    self.characteristics.append("descripcion")
    self.characteristics.append("cola")
    self.characteristics.append("orejas")
    self.characteristics.append("patas")
    self.characteristics.append("pelaje")
```

nuestra clase principal sobre el sistema:

```
class addRoedor:
    def __init__(self,menu,frame1,classifier)->None:
        self.frame1=frame1
        self.main_menu=menu
        self.classifier=classifier
        self.load_characteristics()
        self.labels = []
        self.entries = []

        for characteristic in self.characteristics:
            self.labels.append(Label(self.frame1,text=characteristic.capitalize(),background='#353437',fg="white"))
            if(characteristic=="descripcion" or characteristic=="habitat"):
                self.entries.append(Text(self.frame1, height=2, width=45))
            else:
                self.entries.append(Entry(self.frame1,width=60))
```

El listado de opciones que tenemos:

Las preguntas que ayudaran a nuestro sistema a dar el resultado:

```
def question(self,q,opt):
    options=[]
    options.append("Otro")
    for key in opt.keys():
        options.append(key)
    self.selection=StringVar()
    self.chooses=StringVar()
    self.chooses.set("Otro")
    self.instructions=Label(self.frame1,text="Seleccione la característica correspondiente al roedor:\n\n",background='#353437',fg
    self.instructions.configure(font=("Arial",25))
    self.instructions.pack()
    self.caracteristica=Label(self.frame1,text=q,background='#353437',fg="white")
    self.caracteristica.configure(font=("Arial",25))
    self.caracteristica.pack()
    self.drop=OptionMenu(self.frame1,self.chooses,*options)
```

Funcion donde se muestra el resultado:

```
def showRoedor(self):
    self.name=Label(self.frame1,text=self.roedor.nombre,background='#353437',fg="white")
    self.name.configure(font=("Arial",35))

    openImage=Image.open(self.roedor.imagen)
    img=openImage.resize((200,200))
    self.photo=ImageTk.PhotoImage(img)
    self.image=Label(self.frame1,image=self.photo)

    self.size=Label(self.frame1,text=self.roedor.tamaño,wraplength=1200,background='#353437',fg="white")
    self.size.configure(font=("Arial",14))
    self.description=Label(self.frame1,text=self.roedor.descripcion,wraplength=1200,background='#353437',fg="white")
    self.description.configure(font=("Arial",14))
    self.habitat=Label(self.frame1,text=self.roedor.habitat,wraplength=1200,background='#353437',fg="white")
    self.habitat.configure(font=("Arial",14))
    exp="\n\n\nEl roedor fue encontrado en base a las siguientes características:\n"
    for key in self.rules.keys():
        exp+=key+":"+self.rules[key]+"\n"

    self.explanation=Label(self.frame1,text=exp,wraplength=1200,background='#353437',fg="white")
    self.explanation.configure(font=("Arial",14))
```

GitHub: (https://github.com/anthonycs4/SistemaExperto.git)