Actividad 2: Reconocimiento Multimodal de Emociones en Texto e Imagen

Objetivo de la Actividad
Diseñar e implementar un sistema que sea capaz de detectar emociones expresadas tanto en una frase escrita como en una imagen fácil, y comparar si ambas expresan la misma emoción.
Grupo : 2

Competencias que se desarrollan

- Procesamiento simbólico de emociones desde texto.
- Uso de modelos computacionales para emociones visuales (FER2013)
- Integración y comparación de señales multimodales.

Material entregado

1. Un conjunto de frases en español asociadas a emociones básicas. Este archivo es en formato JSON.
2. Un conjunto de imágenes faciales con emociones visibles.
3. Un modelo pre-entrenado para clasificar emociones desde imagen (Fer2013)
4. Un diccionario emocional simbólico para analizar emociones desde texto (diccionario_emociones.py)

Se encuentra en el siguiente link

Librerías que posiblemente requerirán

1- pip install tensorflow : clasificador de imágenes
2- pip install numpy: operaciones numéricas con imágenes
3- pip install Pillow: para cargar imágenes
4- pip install opencv-python (opcional) : permite mostrar imágenes y depurar fácilmente.
5- pip install matplotlib : para hacer histogramas o visualizar emociones detectadas.

Requisitos:

Python >= 3.8

Tareas a realizar

1. **Análisis de emociones desde texto**
    - Usar el diccionario entregado para identificar la emoción dominante en cada frase.
    - Aplicar puntajes y reglas simples (e.g. negación)
    - Almacenar resultados por frase.
2. **Análisis de emociones desde imagen**
    - Cargar cada imagen facial
    - Usar el modelo para predecir la emoción facial.
    - Almacenar resultados por imagen.
3. **Comparación entre modalidades**
    - Comparar emoción detectada en la frase y en la imagen asociada.
    - Determinar si convergen (coinciden) o divergen (no coinciden) todas las frases.
    - Analizar posibles causas de divergencia.

Entregables

Un archivo .zip que contenga:

1. clasificador_texto.py
2. clasificador_imagen.py
3. comparador_multimodal.py
4. resultados.csv (frase, imagen, emocion_texto, emoción_imagen, coincide)
5. informe_reflexion.pdf (máx 2 páginas)

Criterio	Puntos
Clasificador de texto funcional y preciso	25
Clasificador de imagen implementado y funcional	25
Comparación precisa entre ambas emociones	20
Análisis reflexivo en el informe	20
Organización, claridad del código y entregables	10
Total	**100**

Más información: Paso a paso a seguir

Paso 1: Preparar el entorno

1- Instalar las librerías necesarias.
2- Opcional (para visualizar imágenes o gráficas) las librerías opencv-python y matplotlib.

Comando para instalar librerías pip install.

Sino tienen instalado pip pueden usar python -m pip install

Paso 2: Analizar emociones desde texto

1. Crear un archivo clasificador_texto.py
2. Importa el diccionario diccionario_emociones.py. Para importar es usando: **from diccionario_emociones import diccionario_emociones**
3. Preprocesar el texto
   Se define una función para normalizar y dividir la frase en palabras:

```
import re

def preprocesar(frase):
        frase = frase.lower()
        tokens = re.findall(r'\b\w{3,}\b', frase)  # Solo palabras de al
      menos 3 letras
  return tokens
```

4. Analizar la emoción
   La función principal recorre los tokens y suma los puntajes de las emociones encontradas:

```
from collections import defaultdict

def analizar_emocion(frase, diccionario):
  tokens = preprocesar(frase)
```

```
    puntuaciones = defaultdict(int)
    negacion = False

    for token in tokens:
        if token in ["no", "nunca", "jamás"]:
            negacion = True
            continue

        if token in diccionario:
            for emocion, puntaje in diccionario[token].items():
                if negacion:
                    puntuaciones[emocion] -= puntaje
                    negacion = False
                else:
                    puntuaciones[emocion] += puntaje

    if not puntuaciones:
        return "indefinida", {}

    emocion_predominante = max(puntuaciones.items(), key=lambda x: x[1])
    return emocion_predominante[0], dict(puntuaciones)
```

Ejemplo de uso

```
frase = "Estoy muy feliz con el resultado"
emocion, detalle = analizar_emocion(frase, diccionario_emociones)
print(f"Emoción detectada: {emocion}")
print(f"Puntajes: {detalle}")
```

Salida esperada:

Emoción detectada: alegría
Puntajes: {'alegría': 2}

(opcional) Puedes usar spacy si quieres lematizar los verbos.

¿Qué es lematizar? Significa reducir una palabra a su forma base o canónica, llama lema. Por ejemplo “disfruté”, el lema es disfrutar. Esto evitaría crear un diccionario por varias conjugaciones del mismo verbo.

Si quieres usar spacy debes considerar el modelo de español

```
pip install spacy
python -m spacy download es_core_news_sm
# Cargar el modelo de español
nlp = spacy.load("es_core_news_sm")
```

y definir una función llamada lematizar, el cual reemplazaría a preprocesar.

```
def lematizar(frase):
  doc = nlp(frase.lower())
  return [token.lemma_ for token in doc if token.is_alpha]
```

Paso 3: Analizar emociones desde imagen

1. Se requiere tener instalado tensorflow, numpy y Pillow.
2. Cargar el modelo fer2013_mini_XCEPTION.99-0.65.hdf5

```
from tensorflow.keras.models import load_model
model = load_model("fer2013_mini_XCEPTION.99-0.65.hdf5")
```

Este modelo espera imágenes de entrada de 64x64 píxeles con 1 canal (escala de grises).

3. Preprocesar la imagen

```
from tensorflow.keras.preprocessing import image
import numpy as np

def cargar_imagen_gris_64x64(path):
  img = image.load_img(path, color_mode="grayscale", target_size=(64, 64))
  x = image.img_to_array(img)       # (64, 64, 1)
  x = x / 255.0                     # Normaliza entre 0 y 1
  x = np.expand_dims(x, axis=0)     # (1, 64, 64, 1)
  return x
```

4. Predecir la emoción

```
emotion_labels_en = ['angry', 'disgust', 'fear', 'happy', 'sad', 'surprise', 'neutral']
emotion_labels_es = {
  'angry': 'enojo',
  'disgust': 'asco',
  'fear': 'miedo',
  'happy': 'alegría',
  'sad': 'tristeza',
  'surprise': 'sorpresa',
  'neutral': 'neutral'
}

def predecir_emocion_desde_imagen(path_imagen):
  x = cargar_imagen_gris_64x64(path_imagen)
  pred = model.predict(x)
  idx = np.argmax(pred)
  emocion_en = emotion_labels_en[idx]
  return emotion_labels_es[emocion_en]
```

Ejemplo de uso

```
imagen = "imagenes/01.jpg"
emocion = predecir_emocion_desde_imagen(imagen)
print(f"Emoción detectada en imagen: {emocion}")
```

Salida esperada

```
Emoción detectada en imagen: alegría
```

A tener en cuenta: Si se usa una imagen de tamaño incorrecto (por ejemplo, 48x48), el modelo fallará. Por eso el *target_size=(64, 64)* es obligatorio.