

Tarea 8 Digitales III

Melanie Aponte

June 2025

1. Crear un dataset de 4 emociones con 40 imágenes de rasgos faciales (10 por cada emoción) usando la herramienta roboflow.
2. Se debe corregir el siguiente código e implementarlo en Pepper (Nota: El archivo deberá realizarse en la plataforma robótica en la carpeta de "D3/Cámara"). El algoritmo es:

```
photo_service = session.service("ALPhotoCapture")
photo_service.setResolution(2)
photo_service.setPictureFormat("jpg")
photo_service.takePictures(1,
    "/home/nao/recordings/cameras/", "image")
tablet_service.showImage("http://198.18.0.1/home/nao/recordi
```

3. Dentro de Pepper se encuentra un código de implementación de la API AIEmotionRecognition, donde está alojada la carpeta "D3/AIEmotionRecognition". Interactuar con la misma y sacar 5 conclusiones de este.
4. Desarrollar una revisión de la literatura presente sobre análisis de sentimientos y detallar 5 tecnologías actuales que se estén usando, se debe anexar las referencias de forma detallada y para cada tecnología se debe anexar un ejemplo sencillo (ejemplo un código en python).

1 Dataset

La creacion del dataset se llevo a cabo mediante Roboflow, descargue 40 imagenes de 4 emociones, 10 imagenes de cada emocion y las organice por carpetas: Felicidad, tristeza, enojo y sorpresa. Esta carpeta la subi a Roboflow para crear el dataset.

Projects

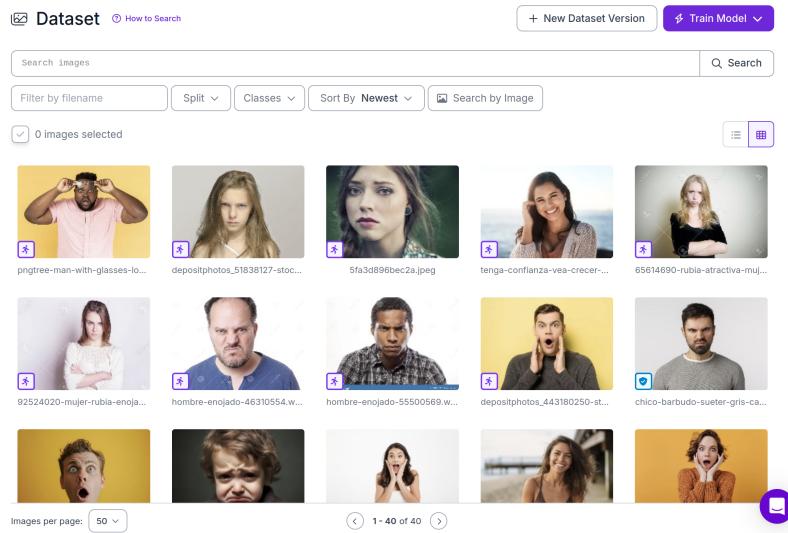
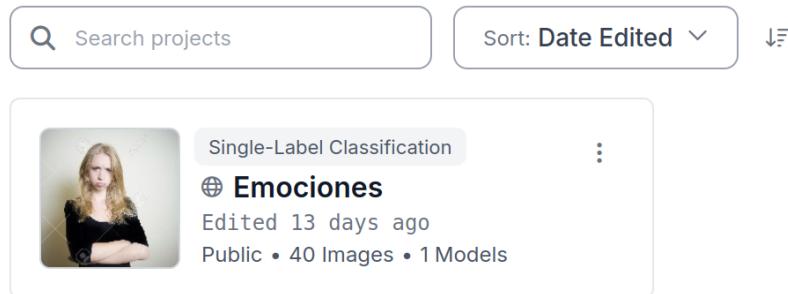


Figure 1: Dataset en roboflow.

2 Correccion del algoritmo

El algoritmo dado fue: Para llevar a cabo la implementacion del algoritmo

```
photo_service = session.service("ALPhotoCapture")
photo_service.setResolution(2)
photo_service.setPictureFormat("jpg")
photo_service.takePictures(1,
    "/home/nao/recordings/cameras/", "image")
tablet_service.showImage("http://198.18.0.1/home/nao/recordin
```

corregido entre a la carpeta D3/Camara y cree un archivo .py con mi nombre donde puse el algoritmo corregido con ayuda de nano

```
Pepper [0] ~/D3 $ cd Camara/
Pepper [0] ~/D3/Camara $ ls
Cam:_Julian.py Camara_Lu Camara_Vale Camara_Vale.py Melanie.py
Pepper [0] ~/D3/Camara $
```

```
GNU nano 2.3.2                                         File: Melanie.py
# -*- coding: utf-8 -*-
from naoqi import ALProxy

# Conexión al servicio de captura de fotos
photo_service = ALProxy("ALPhotoCapture", "192.168.0.106", 9559)
photo_service.setResolution(2) # 2 = VGA (640x480)
photo_service.setPictureFormat("jpg")

# Captura de una foto y guarda en el robot
photo_service.takePicture("/home/nao/recordings/cameras", "image")

# Mostrar la imagen en la tablet
tablet_service = ALProxy("ALTabletService", "192.168.0.106", 9559)
tablet_service.showImage("http://198.18.0.1/apps/image.jpg")
```

Figure 2: Algoritmo corregido

Paso a Paso del Código:

1. Importación de módulos

Se importan los módulos necesarios: - ”ALProxy” para conectarse a servicios del robot. - ”os” para ejecutar comandos del sistema como copiar archivos.

2. Conexión al servicio de captura de fotos Se crea un proxy al servicio "ALPhotoCapture" usando la IP del robot y el puerto 9559.
3. Configuración de la cámara - Se establece la resolución como VGA (640x480).
 - Se define el formato de la imagen como "jpg".
4. Captura y guardado de la imagen Se toma una foto y se guarda en '/home/nao/recordings/cameras' con el nombre "image.jpg".
5. Mostrar la imagen en la tablet del robot Se crea un proxy al servicio "ALTTabletService" y se utiliza "showImage" para mostrar la imagen a través de la URL "http://198.18.0.1/apps/image.jpg".

Y para verificar su funcionamiento en la carpeta Recordings se ve la generacion de la imagen jpg

```
Pepper [0] ~ $ cd recordings/
Pepper [0] ~/recordings $ ls
camera cameras
Pepper [0] ~/recordings $ cd cameras
Pepper [0] ~/recordings/cameras $ ls
image.jpg imagen.jpg myvideo.avi
```

Figure 3: Comprobacion

3 API AIEmotionRecognition

Conclusiones:

- **El robot puede detectar emociones humanas a través de la voz:** Utiliza el servicio `ALVoiceEmotionAnalysis` para identificar la emoción predominante en el tono de voz de la persona que habla.
- **Se responde empáticamente según la emoción:** Cuando se detecta alegría o tristeza, el robot responde con un mensaje animado que refleja comprensión emocional, fomentando una interacción social más natural.
- **Uso de animaciones en el habla:** La función `ALAnimatedSpeech` no solo permite hablar, sino que añade movimientos al cuerpo del robot, haciendo que la comunicación sea más expresiva y humana.
- **El código está diseñado para ejecutarse continuamente:** El bucle `while True` permite un monitoreo constante de emociones hasta que el usuario interrumpa el programa.
- **El índice de emoción requiere interpretación previa:** El código trabaja con índices numéricos (`matched_emotion_index == 3`, etc.), lo que

sugiere que hay un mapeo interno de emociones que el programador debe conocer (ej. 3 = alegría, 4 = tristeza). Este mapeo debería documentarse para facilitar mantenimiento o ampliación del sistema.

4 Análisis de sentimientos

El análisis de sentimientos es una subdisciplina del procesamiento del lenguaje natural (PLN) que busca identificar, extraer y clasificar opiniones o emociones

4.1 Google Cloud Natural Language API

La API de Cloud Natural Language proporciona a los desarrolladores tecnologías de comprensión del lenguaje natural, incluidos el análisis de opiniones, el análisis de entidades, el análisis de opiniones sobre entidades, la clasificación de contenido y el análisis sintáctico. Esta API forma parte de la más amplia familia de la API de Cloud Machine Learning.

4.1.1 Ejemplo de uso

Para esta se debe tener una configuracion previa de CPG

1. Crear un proyecto en CPG
2. Habilitar la API de Google Cloud Natural Language
3. Crear una cuenta de servicio (Service Account)
4. Generar una clave JSON
5. Establecer la variable de entorno en tu sistema

```
export GOOGLE_APPLICATION_CREDENTIALS="/ruta/completa/a/tu/clave.json"
```

6. Instalar la librería cliente en Python

```
[melanie-aponte@melanie-aponte-Vivobook-ASUSLaptop-X3400PA-K3400PA: ~]$ pip install google-cloud-language
```

```
from google.cloud import language_v1
client = language_v1.LanguageServiceClient()
document = language_v1.Document(content="The hotel was clean and well located.", type_=language_v1.Document.Type.PLAIN_TEXT)
sentiment = client.analyze_sentiment(request={'document': document}).document_sentiment
print("Sentiment score:", sentiment.score, "Magnitude:", sentiment.magnitude)
```

Figure 4: Código de aplicación

4.2 MonkeyLearn

MonkeyLearn es una plataforma de análisis de texto basada en inteligencia artificial que permite a empresas y usuarios extraer información útil de datos no estructurados como correos electrónicos, encuestas, reseñas, chats y más. Ofrece herramientas sin necesidad de programación para entrenar modelos de aprendizaje automático personalizados o utilizar modelos preentrenados para tareas como análisis de sentimientos, clasificación de temas y extracción de palabras clave.

4.2.1 Ejemplo de uso

1. Primero se debe instalar la libreria

```
melanie-aponte@melanie-aponte-Vivobook-ASUSLaptop-X3400PA-K3400PA: ~ pip install monkeylearn
```

2. implementar codigo.py

```
GNU nano 7.2                                         MonkeyLearn.py *
from monkeylearn import MonkeyLearn

# Paso 1: Inicializa el cliente con tu API key
ml = MonkeyLearn('TU_API_KEY') # Reemplaza con tu API Key

# Paso 2: Define los textos que deseas analizar
texts = [
    'I love this product, it is amazing!',
    'I hate waiting, the service is terrible.',
    'The experience was okay, nothing special.'
]

# Paso 3: Usa el modelo preentrenado para análisis de sentimientos
model_id = 'cl_p13C73L' # Modelo de análisis de sentimientos en inglés
result = ml.classifiers.classify(model_id, texts)

# Paso 4: Imprime los resultados
for i, res in enumerate(result.body):
    print(f'Texto: {texts[i]}')
    print(f'Sentimiento: {res['classifications'][0]['tag_name']}, Probabilidad: {res['classifications'][0]['confidence']:.2f}\n')
```

Figure 5: Código para Análisis de sentimientos

En mi caso no fue posible correr el programa ya que el plan api tiene un valor de 300 USD pero se esperaría una salida como:

```
Texto: I love this product, it is amazing!
Sentimiento: Positive, Probabilidad: 0.95

Texto: I hate waiting, the service is terrible.
Sentimiento: Negative, Probabilidad: 0.99

Texto: The experience was okay, nothing special.
Sentimiento: Neutral, Probabilidad: 0.78
```

Figure 6: Salida esperada

4.3 HuggingFace Transformers + RoBERTa

RoBERTa es una variante robusta de BERT entrenada con más datos y mejores configuraciones. Es ampliamente utilizada en tareas de sentimientos por su alto rendimiento.

4.3.1 Ejemplo de uso:

1. Tener Python 3.7+ instalado
2. Instalar las librerías necesarias

```
melanie-aponte@melanie-aponte-Vivobook-ASUSLaptop-X3400PA-K3400PA:~$ pip install nltk
```

3. Crear codigo.py

```
#!/usr/bin/env python3
# coding: utf-8

from transformers import AutoTokenizer, AutoModelForSequenceClassification, pipeline

# Cargar modelo y tokenizador de HuggingFace
modelo = AutoModelForSequenceClassification.from_pretrained("roberta-base")
tokenizer = AutoTokenizer.from_pretrained(modelo)
model = AutoModelForSequenceClassification.from_pretrained(modelo)

# Crear pipeline de análisis de sentimientos
claseficador = pipeline("sentiment-analysis", model=model, tokenizer=tokenizer)

# Texto a analizar
texto = "I absolutely love this new phone! The camera is amazing."
# Clasificación
resultado = claseficador(texto)
print(resultado)
```

4.4 TextBlob

Librería de Python que proporciona una API simple para tareas comunes de procesamiento de texto, incluyendo análisis de sentimientos usando una combinación de reglas y modelos entrenados.

```
(juegos) melanie-aponte@melanie-aponte-Vivobook-ASUSLaptop-X3400PA-K3400PA:~/Juegos$ python ejemplo.py
config.json: 100%
vocab.json: 100%
merges.txt: 100%
special_tokens_map.json: 100%
pytorch_model.bin: 100%
Device set to use cpu
[{'label': 'LABEL_2', 'score': 0.9927905797958374}]
(juegos) melanie-aponte@melanie-aponte-Vivobook-ASUSLaptop-X3400PA-K3400PA:~/Juegos$ python ejemplo.py
Device set to use cpu
[{'label': 'LABEL_2', 'score': 0.9927905797958374}]
(juegos) melanie-aponte@melanie-aponte-Vivobook-ASUSLaptop-X3400PA-K3400PA:~/Juegos$
```

Figure 7: Salida al ejecutar el código

4.4.1 Ejemplo de uso

1. Tener python instalado
2. Instalar TextBlob y sus dependencias en un entorno virtual

```
(analisis) melanie-aponte@melanie-aponte-Vivobook-ASUSLaptop-X3400PA-K3400PA:~$ pip install textblob
Collecting textblob
  Downloading textblob-0.15.0-py3-none-any.whl (10 kB)
```

```
Successfully installed cycler-0.10.1 joblib-1.0.1 mock-3.7.1 regex-2021.11.0 textblob-0.15.0 tqdm-4.67.1
(analisis) melanie-aponte@melanie-aponte-Vivobook-ASUSLaptop-X3400PA-K3400PA:~$ python -m textblob.download_corpora
[nltk_data] Downloading package brown to /home/melanie.
```

Figure 8: Descarga de corpus, necesario para análisis de sentimientos

3. Hacer código.py

```
GNU nano 7.2                                     ejemplo.py *
from textblob import TextBlob

# Texto a analizar
texto = "I really enjoy working with Python. It's amazing!"

# Crear objeto TextBlob
blob = TextBlob(texto)

# Obtener sentimiento
print(blob.sentiment)
```

Figure 9: Código ejemplo en python

Y la salida que se obtiene es la siguiente:

```
(analisis) melanie-aponte@melanie-aponte-Vivobook-ASUSLaptop-X3400PA-K3400PA:~$ python ejemplo.py
Sentiment(polarity=0.575, subjectivity=0.7)
(analisis) melanie-aponte@melanie-aponte-Vivobook-ASUSLaptop-X3400PA-K3400PA:~$
```

Figure 10: Salida de TextBlob

4.5 BERT

Modelo basado en transformadores desarrollado por Google. Permite realizar análisis de sentimientos con alta precisión al capturar el contexto bidireccional de las palabras.

4.5.1 Ejemplo de uso

1. Tener python 3.7 o superior
2. Instalar dependencias necesarias

```
(analisis) melanie-aponte@melanie-aponte-Vivobook-ASUSLaptop-X3400PA-K3400PA: $ pip install transformers torch
```

Figure 11: Instalación de dependencias

3. Usar un modelo BERT preentrenado para análisis de sentimientos

```
GNU nano 7.2                                     ejemplo2.py *
```

```
from transformers import pipeline

# Crear el pipeline de análisis de sentimientos
analizador = pipeline("sentiment-analysis")

# Analizar texto
resultado = analizador("I love studying machine learning. It's fascinating!")
print(resultado)
```

Figure 12: Código ejemplo

```
(analisis) melanie-aponte@melanie-aponte-Vivobook-ASUSLaptop-X3400PA-K3400PA: $ python ejemplo2.py
No model was supplied, defaulted to distilbert/distilbert-base-uncased-finetuned-sst-2-english and revision 714eb0f (https://huggingface.co/distilbert/distilbert-base-uncased-finetuned-sst-2-english).
Using a pipeline without specifying a model name and revision in production is not recommended.
config.json: 100% | 629/629 [00:00<00:00, 1.94MB/s]
model.safetensors: 100% | 268M/268M [00:26<00:00, 16.1MB/s]
tokenizer_config.json: 100% | 48.0/48.0 [00:00<00:00, 253KB/s]
vocab.txt: 100% | 232k/232k [00:00<00:00, 10.6MB/s]
Device set to use cpu
[{"label": "POSITIVE", "score": 0.999860405921936}]
(analisis) melanie-aponte@melanie-aponte-Vivobook-ASUSLaptop-X3400PA-K3400PA: $
```

Figure 13: Salida del código

5 Referencias

1. Rogalski, K. (2024, julio 11). *Las 15 mejores herramientas de análisis del sentimiento con IA [Probadas en 2025]*.
2. Blog de Brand24; Marca24. <https://brand24.com/blog/es/mejores-herramientas-de-analisis-de-opiniones/>
3. TextBlob: Simplified Text Processing — TextBlob 0.19.0 documentation.
4. (s/f). Readthedocs.Io. Recuperado el 6 de junio de 2025, de <https://textblob.readthedocs.io/en/dev/>
5. Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018).
6. BERT: Pre-training of deep bidirectional Transformers for language understanding. En *arXiv [cs.CL]*. <http://arxiv.org/abs/1810.04805>
7. (S/f). Medallia.com. Recuperado el 6 de junio de 2025, de https://www.medallia.com/platform/text-analytics/?utm_campaign=monkeylearnmigration