Herramientas para la Computación en la Nube Dirigida a Inteligencia Artificial

Tema 8. Integración de servicios cognitivos en aplicaciones de negocio

Índice

Esquema

8.1. Introducción y objetivos

En este tema, se describen las características de los servicios cognitivos de Azure que emplean modelos de IA preentrenados o personalizados para resolver un amplio abanico de escenarios de negocio.

En primer lugar, se describen las capacidades de visión artificial con **Computer Vision** y **Custom Vision,** que permiten extraer información detallada y relevante del contenido de imágenes. Esta posibilidad reduce significativamente el esfuerzo y coste asociado a procesos manuales basados en equipos humanos. También se describen las capacidades del **servicio Face** para identificar caras en imágenes, cuyo principal caso de uso es la verificación automática de identidades.

Asimismo, se describen los servicios de procesamiento automatizado de formularios *(form recognizer),* de análisis de vídeo *(video indexer),* de análisis de textos (Azure AI Language) y de conversión de audio en texto y de texto en audio (Azure AI Speech). Se proporcionan ejemplos de uso de las APIs expuestas por algunos de estos servicios a través de la SDK para Python.

Finalmente, se proporcionan ejemplos de integración de los servicios descritos en tres tecnologías de nube de Microsoft: Power BI, Power Apps y Power Automate.

Por último, este tema se plantea los siguientes objetivos para el alumno:

- Reconocer los escenarios de negocio en los que un servicio cognitivo en la nube puede generar una ventaja competitiva a una organización.
- Identificar qué servicio cognitivo es el adecuado para resolver un caso de uso determinado en una organización.

Ideas clave

- Ser capaces de integrar servicios cognitivos en aplicaciones, de manera que estas puedan optimizar los procesos de negocio de una organización, generando eficiencias en coste.

8.2. Servicios cognitivos de visión artificial con Azure Cognitive Services

Introducción

Los servicios de visión artificial son capaces de analizar imágenes o vídeos que se suben a Azure desde el PC del usuario o a través de una URL. Cada servicio está especializado en un caso de uso, como se muestra en la siguiente tabla:

Servicio	Aplicaciones
Computer Vision	▸ Proporciona modelos preentrenados para ser usados directamente en el análisis de imágenes. ▸ Permite detectar objetos y conceptos en imágenes, y generar descripciones.
Custom Vision	▸ Proporciona modelos preentrenados a los que es posible añadir imágenes propias de un ámbito determinado para aplicaciones especializadas.
Face	▸ Proporciona modelos especializados en el reconocimiento de caras y atributos faciales. ▸ Permite implementar sistemas de verificación de identidad basados en los atributos faciales del usuario.
Form Recognizer	▸ Extrae el contenido de imágenes que contienen texto en forma de pares clave-valor.
Video Indexer	▸ Proporciona modelos para la extracción de información relacionada con el contenido de los vídeos analizados.

Tabla 1. Resumen de servicios de visión artificial de Azure AI Services. Fuente: elaboración propia.

A continuación, se describen en detalle las características de cada servicio.

Computer Vision

Este servicio se utiliza para extraer metadatos acerca de una colección de imágenes, de manera que puedan ser correctamente etiquetadas y clasificadas.

Las capacidades del servicio son las siguientes:

- **Etiquetas de contenido.** Permite detectar conceptos o características presentes en una imagen. Por ejemplo: paisaje, exteriores.
- **Detección de objetos.** Permite detectar objetos presentes en una imagen, junto con sus coordenadas de posición.
- **Detección de marcas.** Permite detectar logotipos de marcas.
- **Categorización de imágenes.** Se utiliza para asignar una categoría a la imagen de acuerdo con una colección predefinida de categorías/subcategorías.
- **Descripciones de imágenes.** Se genera una o varias frases que describen el contenido de la imagen.
- **Detección de caras.** Proporciona las coordenadas del rectángulo que rodea a una cara en la imagen, el sexo (hombre o mujer) y la edad. También puede detectar la pose de la persona e identificarla a partir de sus rasgos faciales.
- **Estilo de imagen.** Permite identificar el tipo de imagen de acuerdo con su estilo: fotografía, ilustración, pintura.
- **Dominios específicos.** Permite realizar una clasificación de acuerdo con un ámbito o dominio específico. Actualmente, solo están soportados dos: puntos destacados y celebridades.

- **Paleta de colores.** Permite identificar los colores presentes en una imagen.
- **Generación de miniaturas.** Recorta el área clave de una imagen para generar una miniatura óptima.
- **Áreas de interés.** Proporciona las coordenadas de los rectángulos que identifican las áreas más relevantes de la imagen.
- **Reconocimiento de texto manuscrito e impreso.** Extrae el contenido textual de la imagen tanto si está manuscrito como impreso.
- **Contenidos inadecuados.** Permite detectar contenidos violentos, sexuales o racistas en imágenes.

En la Figura 1 se muestra un ejemplo de los resultados obtenidos por el servicio tras analizar varias imágenes:

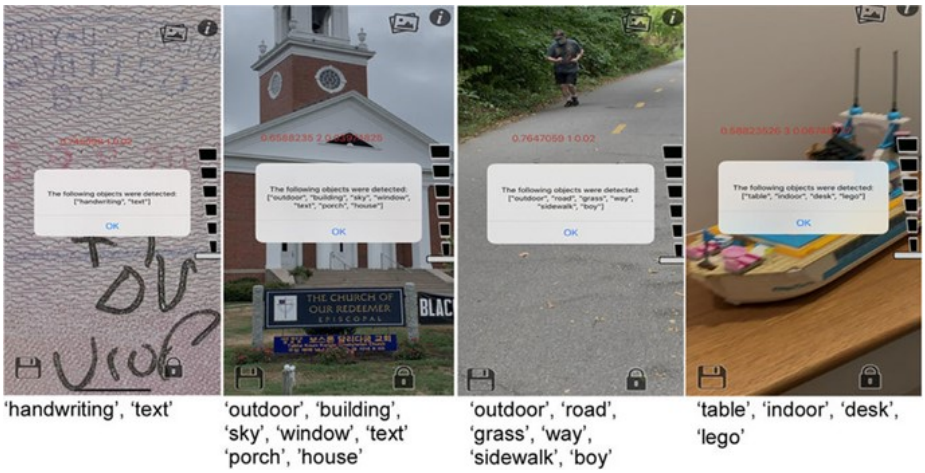


Figura 1. Ejemplos de imágenes analizadas por Computer Vision. Fuente: Luo, 2021.

Ejemplo de uso con Python SDK

El primer paso es crear una cuenta para acceder a la API de Computer Vision. Para ello, se utiliza la CLI de Azure. Se debe especificar la región, el grupo de recursos al que se asigna la cuenta y el nombre de la cuenta.

```
RES_REGION=westeurope

RES_GROUP=[grupo de recursos]

ACCT_NAME=[nombre de cuenta]

az cognitiveservices account create --resource-group $RES_GROUP --name
$ACCT_NAME --location $RES_REGION --kind ComputerVision --sku S1 --yes
```

A continuación, se instala el **paquete pip** del servicio:

```
pip install azure-cognitiveservices-vision-computervision
```

E l *script* en Python arranca con la extracción de dos variables que almacenan **parámetros** de la cuenta necesarios para las sucesivas llamadas:

```
export ACCOUNT_REGION=$(az cognitiveservices account show --resource-group
$RES_GROUP --name $ACCT_NAME --query location --output tsv)

export ACCOUNT_KEY=$(az cognitiveservices account keys list --resource-
group $RES_GROUP --name $ACCT_NAME --query key1 --output tsv)
```

Para acceder a la API se necesita un **cliente.** Se utiliza la región y la *key* de la cuenta, a través de la que se generan las credenciales, para inicializar el cliente.

```
from azure.cognitiveservices.vision.computervision import
ComputerVisionClient

from azure.cognitiveservices.vision.computervision.models import
VisualFeature-Types

from msrest.authentication import CognitiveServicesCredentials
```

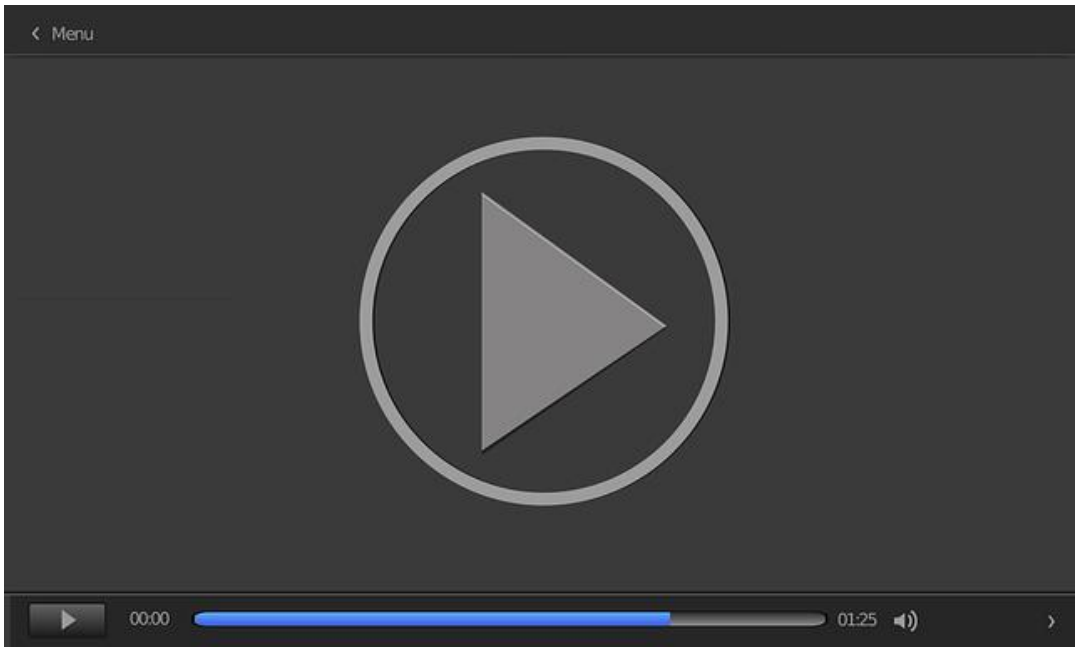
```
import os



region = os.environ['ACCOUNT_REGION']

key = os.environ['ACCOUNT_KEY']



credentials = CognitiveServicesCredentials(key)

client = ComputerVisionClient(endpoint="https://" + region +
".api.cognitive.microsoft.com/", credentials=credentials)
```

En este momento ya es posible acceder a la API. Por ejemplo, se puede utilizar el siguiente fragmento de código para analizar una imagen. En este caso, se extraen las etiquetas asignadas por el servicio a la imagen.

```
url = "https://es.wikipedia.org/wiki/Universidad_Internacional_de_La_Rioja#
/media/Archivo:Universidad_Internacional_de_La_Rioja.jpg"



image_analysis = cli-ent.analyze_image(url,visual_features=
[VisualFeatureTypes.tags])



for tag in image_analysis.tags:

    print(tag)
```

Ideas clave

En el siguiente vídeo, *Uso de servicios cognitivos de visión artificial en Azure con Computer Vision y Face API,* se muestran las posibilidades de análisis de imagen y vídeo de Azure Cognitive Services (AI Services) mediante visión artificial, incluyendo: la detección de objetos, la generación de descripciones de imágenes en lenguaje natural, la transcripción de texto, el reconocimiento de caras o la detección del movimiento de personas en un vídeo.



Accede al vídeo:

https://unir.cloud.panopto.eu/Panopto/Pages/Embed.aspx?id=5353abd1-480a-4028-859c-b16800d582f4

Custom Vision

Este servicio se diferencia de Computer Vision en que se proporcionan imágenes propias para reentrenar los modelos disponibles, de manera que puedan realizar un análisis personalizado para resolver un caso de uso específico.

El proceso de análisis se divide, por tanto, en entrenamiento y predicción. En la **fase de entrenamiento** deben realizarse las siguientes actividades:

- **Creación del proyecto,** que albergará las nuevas imágenes usadas para entrenar el modelo. En estas imágenes aparecen los objetos que queremos identificar en nuevas imágenes. El conjunto de imágenes debe considerar diferentes tipos de iluminación, ángulos y fondos para que la detección posterior sea precisa.
- **Asignación de etiquetas** a estas imágenes.
- **Entrenamiento** del modelo.
- **Exportación del modelo entrenado.** Están disponibles las siguientes opciones: TensorFlow para Android, Core ML para iOS, ONNX para Windows o un contenedor Docker.

En la **fase de predicción** pueden realizarse las siguientes peticiones:

- **Clasificar el objeto** que aparece en una imagen, conforme a las imágenes facilitadas durante el entrenamiento.
- **Detección de la posición del objeto** identificado a través de las coordenadas del rectángulo que lo enmarca.

Ejemplo de uso con Python SDK

En este ejemplo se utiliza el SDK de Python para entrenar un modelo que detecta dos tipos de objetos no contemplados por defecto en Computer Vision.

El primer paso es crear un **recurso de Custom Vision** en la consola web de Azure. Para poder utilizar el SDK de Python, es necesario obtener de la página del recurso el identificador, las *keys* de entrenamiento y predicción y los *endpoints* desplegados. Estos parámetros se almacenan como variables de entorno para que estén disponibles desde el *script* en Python.

```
export VISION_TRAINING_KEY=[key entrenamiento]

export VISION_TRAINING_ENDPOINT=[endpoint entrenamiento]

export VISION_PREDICTION_KEY=[key predicción]

export VISION_PREDICTION_ENDPOINT=[endpoint predicción]

export VISION_PREDICTION_RESOURCE_ID=[identificador de recurso Custom
Vision]
```

La **instalación del paquete pip** del servicio permite acceder posteriormente a la API desde Python:

```
pip install azure-cognitiveservices-vision-customvision
```

El primer fragmento del *script* incluye la importación de las **librerías** necesarias y la asignación de las **variables** definidas previamente:

```
from azure.cognitiveservices.vision.customvision.training import
CustomVision-TrainingClient

from azure.cognitiveservices.vision.customvision.prediction import
CustomVision-PredictionClient

from azure.cognitiveservices.vision.customvision.training.models import
```

```
ImageFile-CreateBatch, ImageFileCreateEntry, Region

from msrest.authentication import ApiKeyCredentials

import os, time, uuid


ENDPOINT = os.environ["VISION_TRAINING_ENDPOINT"]

training_key = os.environ["VISION_TRAINING_KEY"]

prediction_key = os.environ["VISION_PREDICTION_KEY"]

prediction_resource_id = os.environ["VISION_PREDICTION_RESOURCE_ID"]
```

A continuación, se instancian los clientes de entrenamiento y predicción, utilizando las credenciales generadas a partir de las respectivas *keys:*

```
credentials = ApiKeyCredentials(in_headers={"Training-key": training_key})

trainer = CustomVisionTrainingClient(ENDPOINT, credentials)

prediction_credentials = ApiKeyCredentials(in_headers={"Prediction-key":
prediction_key})

predictor = CustomVisionPredictionClient(ENDPOINT, prediction_credentials)
```

El siguiente paso es **crear un proyecto** en el que se almacenarán las imágenes usadas para entrenamiento. El concepto de iteración se refiere a cada proceso de entrenamiento ejecutado que da como resultado un nuevo modelo. En este caso, la primera iteración es un modelo de clasificación, por lo que el nombre que le damos es classifyModel .

```
publish_iteration_name = "classifyModel"

project_name = uuid.uuid4()
```

```
project = trainer.create_project(project_name)
```

A continuación, se definen los **valores de las etiquetas** que asignaremos a las imágenes de entrenamiento. En este caso, se utilizarán únicamente dos etiquetas para poder identificar dos objetos en nuevas imágenes.

```
etiqueta1 = trainer.create_tag(project.id, "reloj de pulsera")

etiqueta2 = trainer.create_tag(project.id, "reloj de pared")
```

El siguiente paso es **subir las imágenes de entrenamiento.** En este ejemplo se dispone de 100 imágenes para cada objeto. Se asume que los ficheros están disponibles en una carpeta local a la que tiene acceso el *script.*

```
base_image_location = os.path.join (os.path.dirname(__file__), "imagenes")

image_list = []



for image_num in range(1, 101):

    file_name = "pulsera_{}.jpg".format(image_num)

    with open(os.path.join (base_image_location, "pulsera", file_name),
"rb") as image_contents:

        image_list.append(ImageFileCreateEntry(name=file_name,
contents=image_contents.read(), tag_ids=[etiqueta1.id]))



for image_num in range(1, 101):

    file_name = "pared_{}.jpg".format(image_num)

    with open(os.path.join (base_image_location, "pared", file_name), "rb")
as image_contents:
```

```
        image_list.append(ImageFileCreateEntry(name=file_name,
contents=image_contents.read(), tag_ids=[etiqueta2.id]))



trainer.create_images_from_files(project.id,
ImageFileCreateBatch(images=image_list))
```

Una vez que tenemos las imágenes y las etiquetas, podemos iniciar el proceso de entrenamiento. El bucle `while` espera a que el entrenamiento haya finalizado. Finalmente, se publica la iteración en el *endpoint*.

```
iteration = trainer.train_project(project.id)

while (iteration.status != "Completed"):

    iteration = trainer.get_iteration(project.id, iteration.id)

    time.sleep(10)

trainer.publish_iteration(project.id, iteration.id, publish_iteration_name,
prediction_resource_id)
```

Ahora es posible enviar nuevas imágenes al *endpoint* de predicción para tratar de detectar en ellas los dos objetos (relojes de pared o de pulsera). El resultado se expresa en **términos de probabilidad** de pertenencia a cada una de las etiquetas.

```
with open(os.path.join (base_image_location, "pruebas/reloj_pulsera.jpg"),
"rb") as image_contents:

    results = predictor.classify_image(

        project.id, publish_iteration_name, image_contents.read())

    for prediction in results.predictions:

        print("\t" + prediction.tag_name + ":
{0:.2f}%".format(prediction.probability * 100))
```

El proceso descrito desde la SDK para Python puede replicarse a través de la interfaz gráfica que ofrece el portal de Azure. A través del portal, es posible obtener métricas de rendimiento para los datos de entrenamiento, como se muestra en la Figura 2:

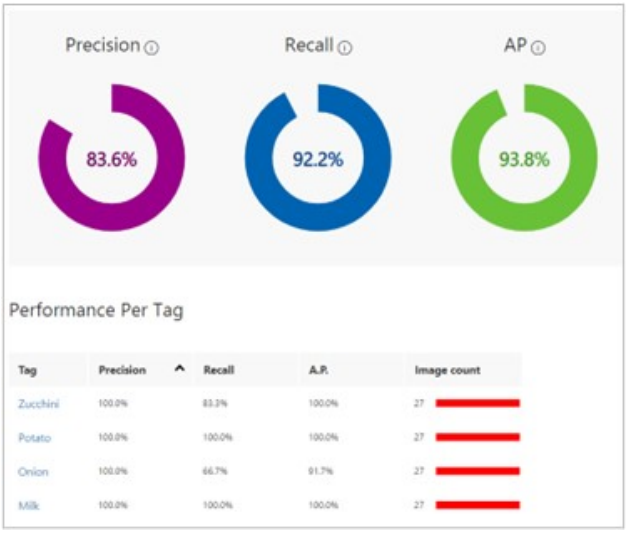
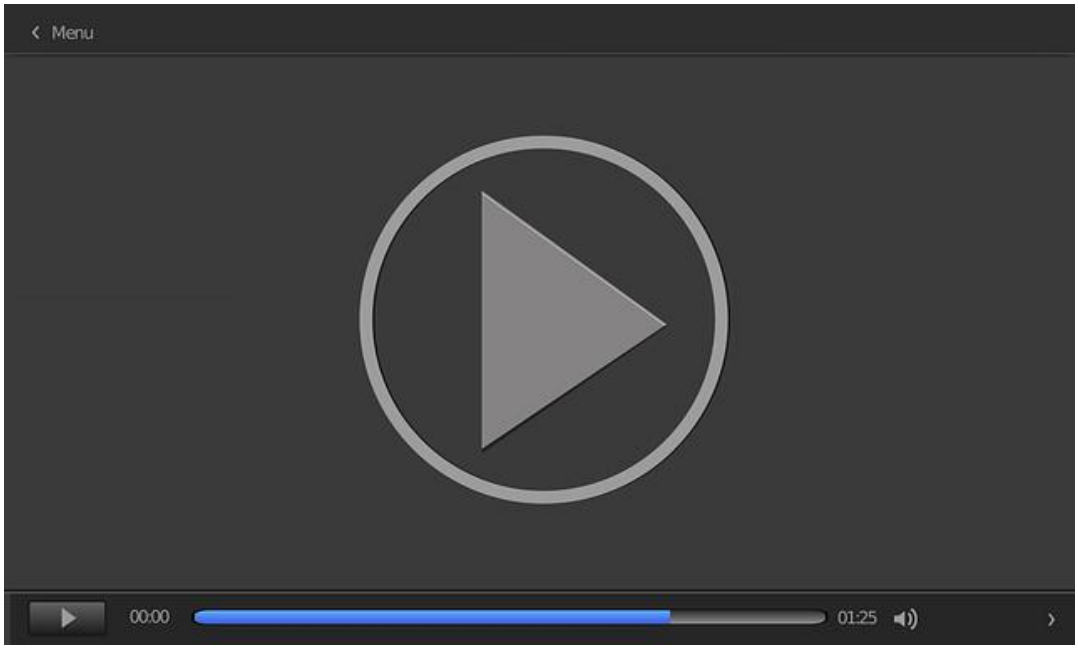


Figura 2. Métricas de rendimiento del proceso de entrenamiento en Custom Vision . Fuente: Eljand, 2021.

En el siguiente vídeo, *Creación de modelos personalizados de clasificación basados en visión artificial con Azure Custom Vision,* se describe, a través de un ejemplo, el proceso para entrenar un modelo de clasificación personalizado en Azure Custom Vision a través de la consola web de Azure.

Ideas clave



Accede al vídeo:

https://unir.cloud.panopto.eu/Panopto/Pages/Embed.aspx?id=dd81a359-cfd2-49a6-9e4f-b16800f41935

Face

Este servicio permite **identificar caras de personas en una imagen,** junto con una colección de atributos faciales: pose, sexo (hombre o mujer), edad, emociones, vello facial y gafas. Además, puede utilizarse como mecanismo de verificación de identidades, comprobando si dos imágenes de caras pertenecen a la misma persona.

En particular, ofrece las siguientes **capacidades:**

- **Detección de caras y atributos.** Se proporcionan las coordenadas de posición de las caras detectadas.
- **Verificación de caras.** Comprueba si dos caras pertenecen a la misma persona.
- **Caras similares.** Permite encontrar caras similares a una dada en un conjunto de imágenes.

- **Agrupación de imágenes.** Permite generar grupos con caras similares a partir de un conjunto de imágenes.
- **Identificación de personas.** Permite localizar a una persona en una base de datos de caras a partir de una imagen.

Ejemplo de uso con Python SDK

En este ejemplo se utiliza la SDK de Python para detectar caras en una imagen y extraer las coordenadas de sus posiciones para incluirlas en esta.

Para utilizar la API de Face desde Python, debe crearse un **recurso Face** desde la consola web de Azure. En la sección «Keys and Endpoint» puede localizarse el valor de la *key,* utilizada para autenticación en la API, y el *endpoint,* utilizado para enviar peticiones a la API.

El primer paso es importar un **conjunto de librerías** comunes de Python, que permiten el manejo de imágenes y otras funciones relacionadas con I/O, fechas y acceso al sistema de ficheros y variables de entorno.

```
import asyncio

import io

import glob

import os

import sys

import time

import uuid

import requests
```

```
from urllib.parse import urlparse

from io import BytesIO

from PIL import Image, ImageDraw
```

A continuación, se importan también las **clases de los servicios cognitivos** de Azure, incluyendo varias específicas del servicio Face.

```
from azure.cognitiveservices.vision.face import FaceClient

from msrest.authentication import CognitiveServicesCredentials

from azure.cognitiveservices.vision.face.models import TrainingStatusType,
Person, SnapshotObjectType, OperationStatusType
```

El siguiente paso es **inicializar el cliente,** utilizando los datos de *endpoint* y *key* rescatados de la consola de Azure para el recurso Face.

```
KEY = os.environ['FACE_SUBSCRIPTION_KEY']

ENDPOINT = os.environ['FACE_ENDPOINT']

face_client = FaceClient(ENDPOINT, CognitiveServicesCredentials(KEY))
```

En este ejemplo se utiliza una imagen alojada en una URL. Se llama a la función face_detect_with_url() para invocar la API del servicio Face, con una **petición de detección de caras** en la imagen.

```
face_image_url = ' https://www.unir.net/wp-content/uploads/2024/04/Catedra-
ARSYS-UNIR.jpg '

image_name = os.path.basename(face_image_url)

detected_faces = face_client.face.detect_with_url(url=face_image_url)
```

El siguiente ejemplo muestra el formato JSON de detected_faces, que almacena el resultado de la detección. Como se puede observar, se incluyen datos como la edad o el sexo de la persona identificada.

```
"faces": [

        {

            "age": 11,

            "gender": "Male",

            "faceRectangle": { "top": 62, "left": 22, "width": 45,
"height": 45

            }

        },
```

La función getRectangle() se emplea para extraer las **coordenadas de posición de los rectángulos** que enmarcan las zonas de la imagen con caras.

```
def getRectangle(faceDictionary):

    rect = faceDictionary.face_rectangle

    left = rect.left

    top = rect.top

    bottom = left + rect.height

    right = top + rect.width

    return ((left, top), (bottom, right))
```

Por último, se descarga la imagen y se dibujan sobre ella dichos rectángulos, a partir de las coordenadas proporcionadas en el resultado devuelto por la API.

```
response = requests.get(face_image_url)

img = Image.open(BytesIO(response.content))

draw = ImageDraw.Draw(img)

for face in detected_faces:

    draw.rectangle(getRectangle(face), outline='red')

img.show()
```

A continuación, se muestra un ejemplo de imagen con las caras detectadas enmarcadas en sus respectivos rectángulos:

Figura 3. Ejemplo de imagen con la cara detectada por el servicio Face. Fuente: PatrickFarley, nitinme y eric-urban, 2024a.

El servicio también puede utilizarse directamente desde la consola web de Azure. En ese caso, los resultados de la detección muestran información adicional (sexo, edad) sobre la imagen original:

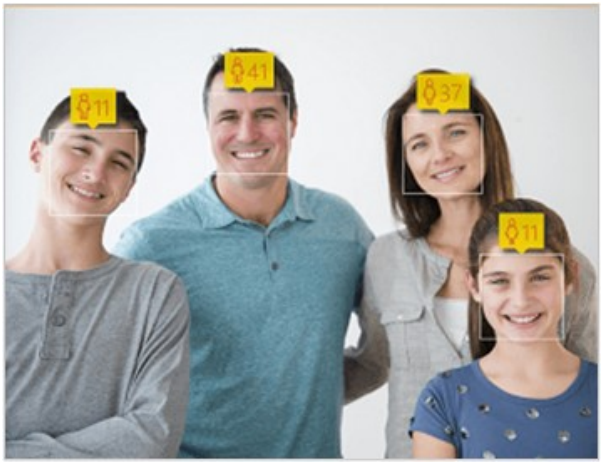


Figura 4. Ejemplo de detección de caras con atributos de las personas. Fuente: PatrickFarley, nitinme y eric-urban, 2024b.

Form Recognizer

Este servicio extrae los textos encontrados en imágenes o documentos PDF. Se utiliza en gran medida para el **procesamiento de formularios.** Los textos extraídos se formatean conforme a una estructura de datos aprendida durante el entrenamiento (con formularios vacíos) y pueden almacenarse en bases de datos. Se basa en **tecnología de reconocimiento de textos (OCR).**

Las características de este servicio son las siguientes:

- **Extracción personalizada.** Es posible extraer texto, tablas, campos de formulario y valores de formulario para cualquier documento, independientemente de su formato.
- **Procesamiento de recibos.** Incorpora múltiples modelos preentrenados que reconocen diferentes formatos de recibos para extraer la información categorizada de forma automática.
- **Identificación de estructura.** Permite identificar la estructura de las tablas y textos de un documento o imagen.

Ejemplo de uso con Python SDK

En este ejemplo se muestra cómo **extraer el texto de un documento** que contiene tablas, listas de selección, campos/valores de formulario y texto manuscrito. Para ello, se utiliza el General Document Model, un modelo preentrenado que *a priori* puede funcionar sobre cualquier tipo de documento, independientemente de su formato.

El primer paso es crear un **recurso de tipo Form Recognizer** en Azure. Para ello, se utiliza el Azure CLI. Solo es necesario especificar el nombre del recurso, el grupo de recursos en el que se incluirá y la región en la que será creado.

```
az cognitiveservices account create --name <your-resource-name> --resource-
group <your-resource-group-name> --kind FormRecognizer --sku <sku> --
location <location> --yes
```

Como en los ejemplos de secciones previas, se deben extraer los valores de *key* y *endpoint* del recurso para las llamadas a la API que haremos a continuación. Estos valores deben incluirse en variables de entorno para ser utilizadas desde el *script.*

```
az cognitiveservices account show --name "resource-name" --resource-group
"re-source-group-name" --query "properties.endpoint"
```

```
az cognitiveservices account keys list --name "<resource-name>" --resource-
group "<resource-group-name>"
```

A continuación, se debe instalar el paquete pip de Python que proporciona acceso a la API de Form Recognizer:

```
pip install azure-ai-formrecognizer
```

El siguiente paso es inicializar el cliente de Form Recognizer (DocumentAnalysisClient) con los valores de *endpoint* y credenciales (obtenidas a partir de la *key*).

```
from azure.core.credentials import AzureKeyCredential

from azure.ai.formrecognizer import DocumentAnalysisClient

endpoint = os.environ["AZURE_FORM_RECOGNIZER_ENDPOINT"]

key = os.environ["AZURE_FORM_RECOGNIZER_KEY"]

credential = AzureKeyCredential(key)

document_analysis_client = DocumentAnalysisClient(endpoint, credential)
```

En este punto es posible empezar a extraer texto del documento. Para ello, se utiliza la función begin_analyze_document() . Con la función result() se retornan los resultados del análisis.

```
with open(path_to_sample_documents, "rb") as f:

    poller = document_analysis_client.begin_analyze_document(

        "prebuilt-document", document=f

    )

result = poller.result()
```

En styles se almacenan los diferentes tipos de texto identificados. Se puede utilizar para extraer el texto manuscrito e imprimirlo:

```
for style in result.styles:

    if style.is_handwritten:

        print(",".join([result.content[span.offset:span.offset +
span.length] for span in style.spans]))
```

En key_value_pairs se almacenan los campos/valores de formulario identificados. Se puede iterar sobre la lista para imprimir cada uno de ellos en el orden en que fueron encontrados en el documento.

```
for kv_pair in result.key_value_pairs:

    if kv_pair.key:

        print("Key '{}' found within '{}' bounding
regions".format(kv_pair.key.content, kv_pair.key.bounding_regions))

    if kv_pair.value:

        print("Value '{}' found within '{}' bounding
regions\n".format(kv_pair.value.content, kv_pair.value.bounding_regions))
```

En pages se puede encontrar información de alto/ancho de las páginas. Para cada página se pueden extraer las líneas de texto, incluyendo coordenadas de posición y la probabilidad de detección correcta de cada palabra, así como las listas de selección encontradas y sus respectivas probabilidades:

```
for page in result.pages:



    print("Page has width: {} and height: {}, measured with unit:
{}".format(page.width, page.height, page.unit))



    for line_idx, line in enumerate(page.lines):

        words = line.get_words()

        print("Line # {} has {} words and text '{}' within bounding polygon
'{}'".format(line_idx, len(words), line.content, line.polygon))

        for word in words:
```

```
            print("Word '{}' has a confidence of {}".format(word.content,
word.confidence))



    for selection_mark in page.selection_marks:

        print("Selection mark is '{}' within bounding polygon '{}' and has
a confidence of {}".format(selection_mark.state, selection_mark.polygon,
selection_mark.confidence))
```

En tables se almacena la información de tablas. Para cada tabla se identifica el número de filas y columnas que contiene, sus coordenadas de posición y el valor de cada celda —referenciada a partir del número de fila y columna—.

```
for table_idx, table in enumerate(result.tables):

    print("Table # {} has {} rows and {} columns".format(table_idx,
table.row_count, table.column_count))



    for region in table.bounding_regions:

        print("Table # {} location on page: {} is {}".format(table_idx,
region.page_number, region.polygon))



    for cell in table.cells:

        print("Cell[{}][{}] has content '{}'".format(cell.row_index,
cell.column_index, cell.content))
```

Ideas clave

Si la extracción de datos de los documentos de ejemplo es correcta, puede **estandarizarse** el proceso para procesar volúmenes elevados de documentos similares. Esta automatización representa un considerable ahorro de tiempo y esfuerzo con respecto a la alternativa de un equipo humano que realice la misma labor.

Video Indexer

Este servicio permite analizar de forma automática un vídeo para extraer diferentes tipos de **metadatos.** Realiza, por tanto, el equivalente a un visionado manual, pero de forma mucho más rápida, utilizando modelos de IA para detectar los elementos clave del contenido. Incorpora todos los mecanismos de detección descritos para Computer Vision, Face y Form Recognizer, pero añade la siguiente información adicional que aplica de manera exclusiva a vídeos:

- **Personas.** Genera información acerca de la frecuencia de aparición y en qué *frames* aparece cada persona.
- **Temas.** Es capaz de identificar las temáticas de un vídeo, la frecuencia con la que se tratan y en qué momentos del vídeo se cubren.
- **Tono.** Permite identificar el tono o tonos presentes en el vídeo, indicando también los *frames* correspondientes.
- **Transcripciones.** Puede extraer y traducir el contenido hablado del vídeo a varios idiomas.

Ejemplo de uso con Python

Para acceder a la API de Video Indexer desde Python, puede utilizarse el paquete pip video-indexer , desarrollado por la comunidad —es decir, no oficial de Microsoft—.

```
pip install video_indexer
```

La clase VideoIndexer proporciona acceso a la API. Se debe especificar la *key* e identificador de la cuenta asociada al servicio Azure Video Indexer y la ubicación en la que se realizará el procesamiento.

Con la función upload_to_video_indexer() se puede subir un vídeo a Azure, asignándole un idioma y un identificador (video_name) que debe ser único. Para obtener los resultados del análisis, se puede llamar a la función get_video_info() .

```
from video_indexer import VideoIndexer

CONFIG = { 'SUBSCRIPTION_KEY': '', 'LOCATION': '', 'ACCOUNT_ID': '' }



vi = VideoIndexer(vi_subscription_key=CONFIG['SUBSCRIPTION_KEY'],
vi_location=CONFIG['LOCATION'], vi_account_id=CONFIG['ACCOUNT_ID'])



video_id = vi.upload_to_video_indexer(input_filename='some-video.mp4',
video_name='some-video-name', video_language='English')



info = vi.get_video_info(video_id, video_language='English')
```

Ideas clave

La variable `info` almacena los fotogramas clave del vídeo. El último paso consiste en invocar la API de Computer Vision para extraer los elementos encontrados. Para ello, se puede utilizar la función `analyze_image_in_stream()` del cliente API de Computer Vision, que recibe como argumentos la imagen correspondiente al *frame* y la lista de elementos que se desean extraer (por ejemplo, marcas, personas u objetos).

8.3. Servicios cognitivos de comprensión del lenguaje y traducción de textos con Azure Cognitive Services

Azure AI Language agrupa todos los servicios de **comprensión de lenguaje natural** y otras funcionalidades relacionadas. Estos servicios pueden utilizarse a través de la consola web de Azure (Language Studio), las APIs y los SDKs para los lenguajes de programación más populares.

A continuación, se describen las **características** principales de los servicios integrados en Azure AI Language.

Clasificación personalizada de textos

Este servicio permite **clasificar textos en categorías** especificadas por el usuario. Para ello, se entrena un modelo con datos de entrenamiento que proporciona el usuario, como **ejemplos de las categorías** que desea identificar posteriormente.

El servicio soporta la clasificación de un texto de acuerdo con una o múltiples categorías. Los casos de uso más habituales son los siguientes:

- **Categorización automática de correos o tiques.** Por ejemplo, se utiliza en departamentos de atención al cliente para agilizar el tratamiento de las peticiones.
- **Buscadores semánticos.** Por ejemplo, las organizaciones pueden etiquetar sus documentos conforme a un esquema de categorías para proporcionar resultados más precisos a las búsquedas en la intranet.

El **proceso de implantación** consta de los mismos pasos que cualquier otra tarea de clasificación: definición de clases, etiquetado de datos, entrenamiento del modelo, revisión del rendimiento del modelo y despliegue del modelo.

A continuación, se muestra un ejemplo con el SDK para Python. En este ejemplo se detecta una **única categoría** para el texto proporcionado, de acuerdo con un modelo personalizado entrenado previamente. Los parámetros necesarios son: el *endpoint* en el que está desplegado el modelo, la *key* del servicio AI Language, el nombre del proyecto bajo el que se ha creado el modelo y el nombre del despliegue.

```
import os

from azure.core.credentials import AzureKeyCredential

from azure.ai.textanalytics import TextAnalyticsClient



endpoint = os.environ["AZURE_LANGUAGE_ENDPOINT"]

key = os.environ["AZURE_LANGUAGE_KEY"]

project_name = os.environ["SINGLE_LABEL_CLASSIFY_PROJECT_NAME"]

deployment_name = os.environ["SINGLE_LABEL_CLASSIFY_DEPLOYMENT_NAME"]

path_to_sample_document = os.path.abspath(

os.path.join(os.path.abspath(__file__), "..",
"./text_samples/custom_classify_sample.txt"))
```

La función `begin_single_label_classify()` realiza la clasificación, y `result()` retorna el resultado obtenido. Por último, el resultado se almacena en `classifications[0]`.

```
text_analytics_client = TextAnalyticsClient(

    endpoint=endpoint, credential=AzureKeyCredential(key))

with open(path_to_sample_document) as fd:

    document = [fd.read()]
```

```
poller = text_analytics_client.begin_single_label_classify(
    document, project_name=project_name, deployment_name=deployment_name)
document_results = poller.result()
for doc, classification_result in zip(document, document_results):
    if classification_result.kind == "CustomDocumentClassification":
        classification = classification_result.classifications[0]
        print("The document text '{}' was classified as '{}' with
confidence score {}.".format(
            doc, classification.category, classification.confidence_score))
```

Como en ejemplos anteriores, el resultado tiene asociado un grado de confianza que determina en qué medida pertenece el documento a la categoría asignada.

Comprensión de lenguaje conversacional

Este servicio permite entrenar modelos propios para reconocer el significado de los mensajes del usuario —escritos en lenguaje natural— en una conversación. Se utiliza como parte del **proceso de creación de chatbots.** En cualquier caso, Azure dispone de alternativas mucho más ágiles para el desarrollo de este tipo de tecnologías: Azure Bot Framework y, especialmente, Microsoft Copilot Studio, que utiliza los modelos de IA generativa de OpenAI.

Detección de entidades

Este servicio permite extraer las **entidades clave de un texto,** tales como lugares, fechas, cantidades, nombres o marcas. Estos metadatos son útiles para generar una respuesta a un mensaje o ejecutar una acción a partir de su contenido. También puede utilizarse para extraer los datos más importantes de la narrativa de un documento.

El servicio incluye unas **capacidades de detección** de entidades de carácter general. Estas capacidades pueden ampliarse, mediante entrenamiento de modelos propios, para detectar otras entidades específicas del caso de uso considerado. Por ejemplo, una organización puede utilizar un modelo propio para detectar a qué proyecto hace mención cada correo electrónico para determinar qué proyectos requieren un mayor volumen de comunicaciones entre empleados.

Detección de idiomas

Este servicio permite identificar el idioma principal de un texto o documento. Soporta la detección de más de 100 idiomas diferentes. Generalmente, se utiliza como una etapa más de un ***pipeline* de procesamiento** o **preparación de datos,** como se muestra en la Figura 5:

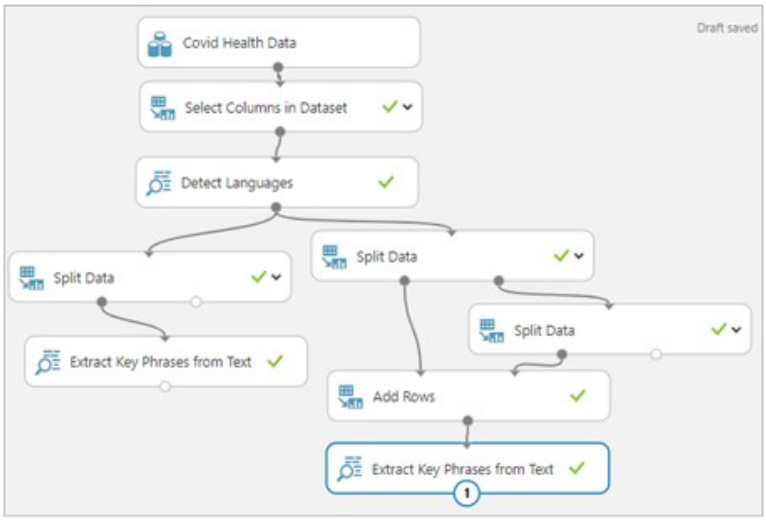


Figura 5. Ejemplo de uso de detección de idiomas con Azure AI Language. Fuente: Asanka, 2021.

Detección de conceptos clave

De forma similar al servicio anterior, la detección de conceptos clave extrae los fragmentos de un texto que considera más relevantes para la comprensión de este. Por ejemplo, en la frase «La comida estaba deliciosa y el servicio era muy amable», los aspectos más relevantes son «comida» y «servicio amable».

Detección de información personal o sensible

Este servicio permite **identificar datos de carácter personal** o sensible en un texto o documento. También puede redactar de nuevo el texto para excluir estos datos —se sustituyen por asteriscos (*)—. En particular, los datos identificados pertenecen a una de las siguientes categorías:

- Nombres de personas.
- Puesto que ocupa una persona.
- Números de teléfono.
- Nombres de organizaciones (empresas, clubes, partidos políticos, etc.).
- Direcciones postales.
- Direcciones de correo electrónico.
- Fechas.
- Direcciones IP.
- Números de cuenta o tarjetas de crédito.
- Datos de configuración de servicios en Azure.

Detección de tono y opiniones

Este servicio categoriza el tono de un texto como **positivo, negativo** o **neutral**. También puede realizar un análisis más detallado para extraer adjetivos que definan el sentimiento u opinión descrito por el usuario en un mensaje. En la Figura 6 se muestra un ejemplo de ambas capacidades:

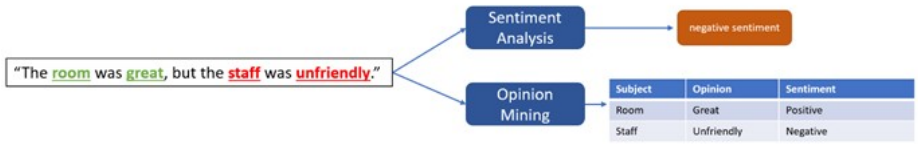


Figura 6. Ejemplo de detección de tono y opiniones. Fuente: Jboback, 2023a.

Generación de resúmenes

Este servicio utiliza modelos de IA preentrenados para generar resúmenes de documentos o conversaciones. Estos pueden ser de n**aturaleza extractiva o abstractiva.** En el primer enfoque se incluyen los fragmentos más relevantes del texto o la conversación, mientras que en el segundo se capturan las principales ideas, sin mencionar el contenido del texto original.

Análisis de textos clínicos

Este servicio se utiliza para extraer la información clave de narrativa clínica. Los resultados del análisis generalmente se utilizan como parte de un proceso de tratamiento de datos más amplio, por ejemplo, para construir modelos predictivos.

En las siguientes capturas se muestran ejemplos de los resultados que puede ofrecer este servicio:

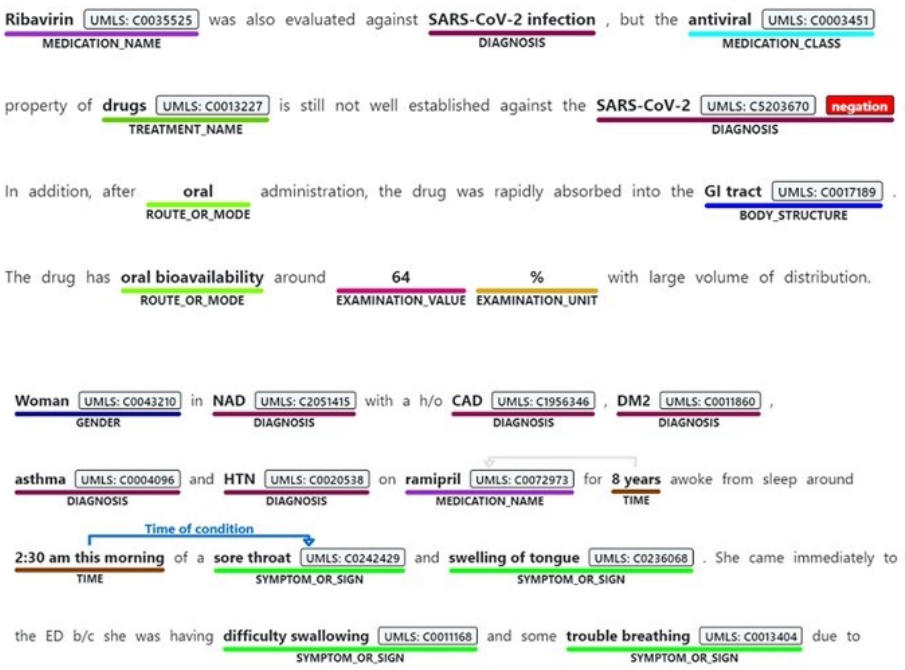


Figura 7. Ejemplo de resultados obtenidos por Azure AI Language para el análisis de textos clínicos.

Fuente: Jbo-back, 2023b.

8.4. Servicios cognitivos de text-to-speech y speech-to-text con Azure Cognitive Services

Este servicio **traduce audio a texto** (transcripción) y **texto a audio.** El sonido de las voces que producen el audio a partir de texto es natural, como si estuviese hablando una persona. También permite traducir el audio a otros **idiomas** y reconocer quién está hablando durante una conversación.

Los **casos de uso más habituales** son los siguientes:

- **Generación de subtítulos.** Estos subtítulos se sincronizan con el audio. Pueden filtrarse los contenidos inadecuados o detectar el idioma del audio.
- **Interacción con chatbots mediante voz** como alternativa a la interfaz basada en texto.
- **Conversión de libros en audiolibros.**
- **Incorporación de voz** a los sistemas de navegación en vehículos.
- **Transcripción de conversaciones** en un *call-center* para análisis de calidad o cumplimiento regulatorio.
- **Aprendizaje de idiomas.** Por ejemplo, puede utilizarse para dictado de textos en otro idioma o corrección de la pronunciación de los alumnos.

En el siguiente ejemplo se muestra cómo **traducir la voz grabada a texto,** utilizando el **SDK de Python.** Como en ejemplos anteriores, se utilizan los valores de *key* del servicio y la región en la que se opera.

```
import os

import azure.cognitiveservices.speech as speechsdk



speech_config =
speechsdk.SpeechConfig(subscription=os.environ.get('SPEECH_KEY'),
region=os.environ.get('SPEECH_REGION'))
```

Para facilitar la labor, se especifica el idioma del audio (en-US). A continuación, se graba desde el micrófono del PC la voz del usuario. El resultado de la transcripción se obtiene con la llamada `recognize_once_async().get()` .

```
speech_config.speech_recognition_language="en-US"

audio_config = speechsdk.audio.AudioConfig(use_default_microphone=True)

speech_recognizer = speechsdk.SpeechRecognizer(speech_config=speech_config,
audio_config=audio_config)

print("Speak into your microphone.")

speech_recognition_result = speech_recognizer.recognize_once_async().get()
```

Finalmente, se imprime este resultado, almacenado en la variable `text` .

```
if speech_recognition_result.reason ==
speechsdk.ResultReason.RecognizedSpeech:

    print("Recognized: {}".format(speech_recognition_result.text))

elif speech_recognition_result.reason == speechsdk.ResultReason.NoMatch:

    print("No speech could be recognized:
{}".format(speech_recognition_result.no_match_details))
```

8.5. Integración de servicios cognitivos con Microsoft Power Platform

Power Platform es una suite de servicios que incluye tres tecnologías: **Power BI,** para visualización y análisis de datos; **Power Apps,** para desarrollo de aplicaciones *low code;* y **Power Automate,** para la creación de automatizaciones a través de *workflows.*

A continuación, se muestran dos ejemplos de integración de los servicios descritos en secciones previas con Power BI y Power Automate.

Integración con Power BI

En este ejemplo se utiliza un *dataset* que contiene, entre otros datos, una columna con los comentarios de los usuarios acerca de su experiencia con un producto o servicio. Este *dataset* se importa en PowerBI y se le añade una columna con el tono de dicho comentario con un valor cercano a **0,** indicando un **tono negativo,** y un valor cercano a **1,** indicando un **tono positivo.**

En la Figura 8 se muestra el detalle de la configuración y los resultados obtenidos — en la columna CognitiveServices.ScoreSentiment —.

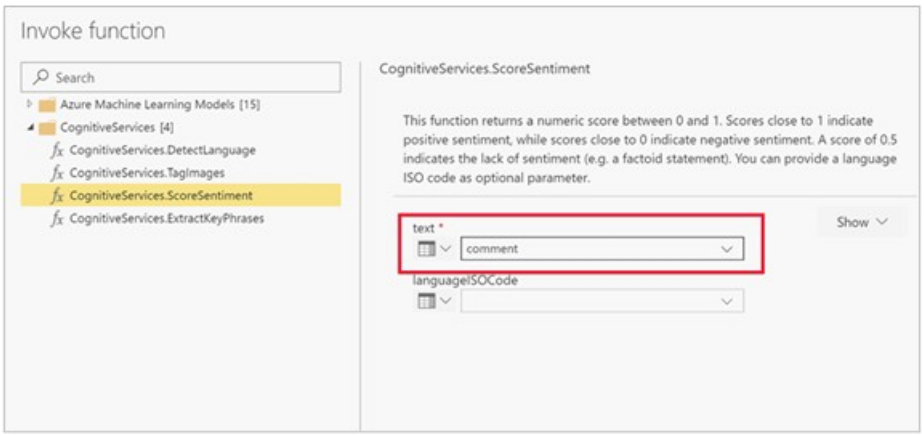


Figura 8. Llamada a la función de análisis de tono de Azure AI Language desde Power BI. Fuente: Davidiseminger, sdgilley, mohitp930, TimShererWithAquent, aphilip94, maggiesMSFT, DCtheGeek, v-kents, 2023.

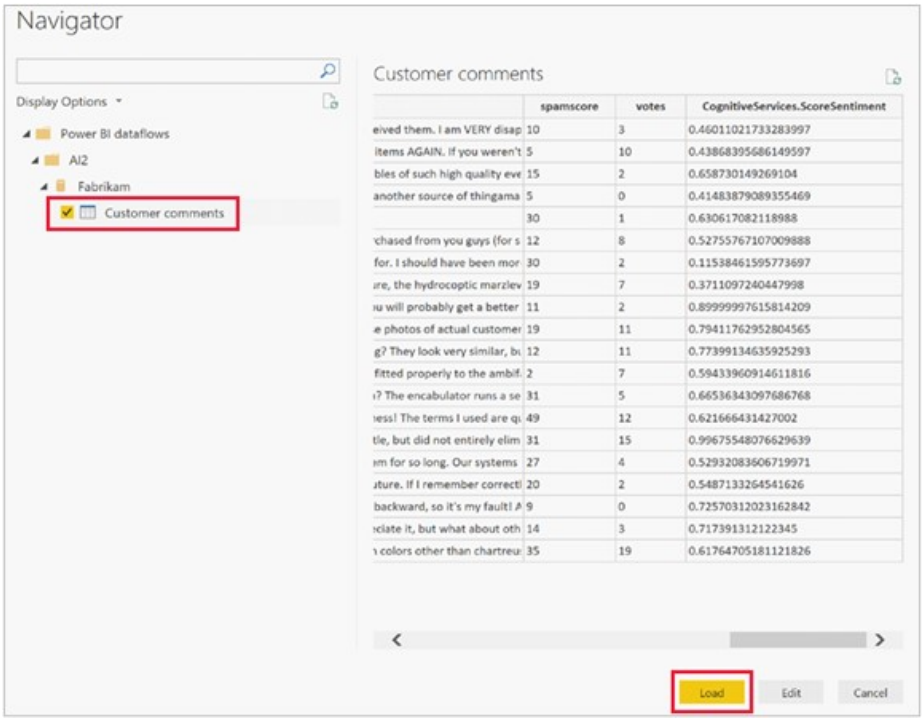


Figura 9. Nueva columna con el resultado del análisis de tono para cada comentario. Fuente: Davidiseminger, sdgilley, mohitp930, TimShererWithAquent, aphilip94, maggiesMSFT, DCtheGeek, v-kents, 2023.

Integración con Power Automate

En este ejemplo se registra en una lista de SharePoint la respuesta a una pregunta lanzada a todos los empleados de una organización.

El flujo de Power Automate se encarga de detectar cuándo se ha incluido una **nueva respuesta en la lista** y analizar el tono de dicha respuesta. El resultado del análisis se incluye también en la lista.

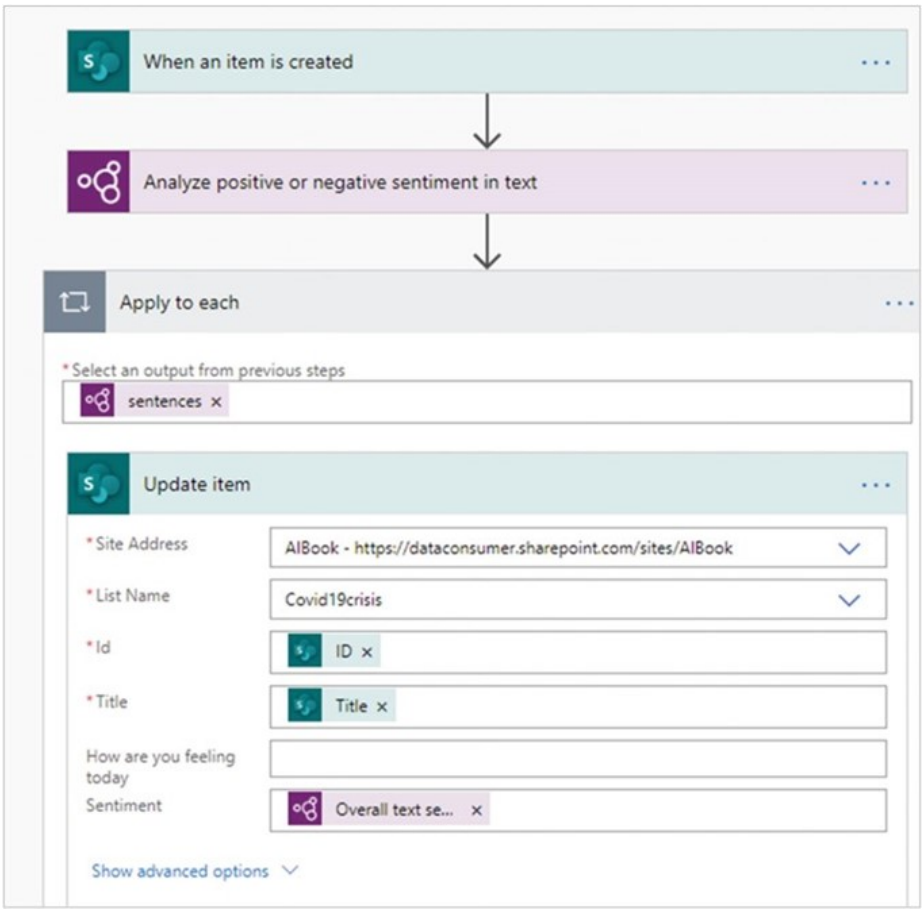


Figura 10. Ejemplo de flujo de Power Automate que analiza el tono de un mensaje incluido en una lista.

Fuente: Moniz, Gordon, Bergum, Chang y Grant, 2021.

8.6. Cuaderno de ejercicios

1. Utilizando el SDK para Python del servicio Computer Vision, extraer el texto de la imagen alojada en la siguiente URL: https://user-images.githubusercontent.com/194400/66840284-b9399100-ef5f-11e9-80e4-6d62f872f908.jpg. Asumir que la variable client (de clase ComputerVisionClient) ya ha sido inicializada y puede realizar llamadas al servicio.

```
from azure.cognitiveservices.vision.computervision.models import
OperationStatus-Codes



url = "https://user-images.githubusercontent.com/194400/66840284-b9399100-
ef5f-11e9-80e4-6d62f872f908.jpg"

raw = True

numberOfCharsInOperationId = 36



rawHttpResponse = client.read(url, language="en", raw=True)



operationLocation = rawHttpResponse.headers["Operation-Location"]

idLocation = len(operationLocation) - numberOfCharsInOperationId

operationId = operationLocation[idLocation:]



result = client.get_read_result(operationId)
```

```
if result.status == OperationStatusCodes.succeeded:

    for line in result.analyze_result.read_results[0].lines:

        print(line.text)

        print(line.bounding_box)
```

2. Utilizando el SDK para Python del servicio Computer Vision, generar una miniatura de la imagen alojada en la siguiente URL: https://github.blog/wp-content/uploads/2020/12/102393310-07478b80-3f8d-11eb-84eb-392d555ebd29.png. Asumir que la variable client (de clase ComputerVision-Client) ya ha sido inicializada y puede realizar llamadas al servicio.

```
from PIL import Image

import io



width = 50

height = 50

url = "https://github.blog/wp-content/uploads/2020/12/102393310-07478b80-
3f8d-11eb-84eb-392d555ebd29.png"



thumbnail = client.generate_thumbnail(width, height, url)



for x in thumbnail:

    image = Image.open(io.BytesIO(x))

image.save('thumbnail.jpg')
```

3. Utilizando el SDK para Python del servicio Computer Vision, describir el contenido de la imagen alojada en la siguiente URL: https://simple.wikipedia.org/wiki/File:GoldenGateBridge-001.jpg. Asumir que la variable client (de clase ComputerVision-Client) ya ha sido inicializada y puede realizar llamadas al servicio.

```
domain = "landmarks"

url = "https://simple.wikipedia.org/wiki/File:GoldenGateBridge-001.jpg"

language = "en"

max_descriptions = 3


analysis = client.describe_image(url, max_descriptions, language)


for caption in analysis.captions:

    print(caption.text)

    print(caption.confidence)
```

4. Utilizando el SDK para Python de Form Recognizer, entrenar un modelo personalizado para extraer información de documentos, considerando que:

- Las variables *endpoint* y *key* contienen la URL de despliegue y la *key* del servicio Form Recognizer, respectivamente.
- La variable training_docs_url contiene la ruta al contenedor de Blob Storage con los documentos usados para el proceso entrenamiento, que tienen diferentes formatos.
- Para cada tipo de documento (formato) identificado, imprimir el nombre de cada campo y el *confidence score* asociado.

```
from azure.ai.formrecognizer import (
    DocumentModelAdministrationClient,
    ModelBuildMode,
)
from azure.core.credentials import AzureKeyCredential


document_model_admin_client = DocumentModelAdministrationClient(
    endpoint, AzureKeyCredential(key))
poller = document_model_admin_client.begin_build_document_model(
    ModelBuildMode.TEMPLATE, blob_container_url=training_docs_url,
description=" modelo propio para extraer datos de documentos")
model = poller.result()


for name, doc_type in model.doc_types.items():
```

```
    for field_name, field in doc_type.field_schema.items():
        print(
            f"Field: '{field_name}' has type '{field['type']}' and
confidence score "
            f"{doc_type.field_confidence[field_name]}")
```

5. Utilizando el paquete `video-indexer`, extraer y mostrar las miniaturas de los *frames* clave de un vídeo previamente analizado. La variable `info` contiene el resultado de la llamada `get_video_info()`.

```
keyframes = []
for shot in info["videos"][0]["insights"]["shots"]:
    for keyframe in shot["keyFrames"]:
        keyframes.append(keyframe["instances"][0]['thumbnailId'])
print("Found #{} keyframes in video".format(len(keyframes)))


from IPython.display import display
from PIL import Image
import io


for keyframe in keyframes:
    img_str = vi.get_thumbnail_from_video_indexer(video_id, keyframe)
    img = Image.open(io.BytesIO(img_str))
```

```
display(img)
```

Extraído de: Borsntein, A. (2020, mayo 13). Custom Video Key Frame Image Classification and Brand Detection for Azure Video Indexer. GitHub. https://github.com/aribornstein/AzureVideoIndexerVisualBrandDetection/blob/master/Video%20Indexer%20Keyframe%20Brand%20Detection.ipynb

8.7. Referencias bibliográficas

Asanka, D. (2021, octubre 1). Language Detection in Azure Machine Learning with basic Text Analytics Techniques. *SQL Shack - Articles About Database Auditing, Serv-er Performance, Data Recovery, and More.* https://www.sqlshack.com/language-detection-in-azure-machine-learning-with-basic-text-analytics-techniques/

Borsntein, A. (2020, mayo 13). Custom Video Key Frame Image Classification and Brand Detection for Azure Video Indexer. GitHub. https://github.com/aribornstein/AzureVideoIndexerVisualBrandDetection/blob/master/Video%20Indexer%20Keyframe%20Brand%20Detection.ipynb

Davidiseminger, sdgilley, mohitp930, TimShererWithAquent, aphilip94, maggiesMSFT, DCtheGeek, v-kents (2023, junio 2). Tutorial: Uso de Cognitive Services en Power BI - Power BI. *Microsoft Learn.* https://learn.microsoft.com/es-es/power-bi/connect-data/service-tutorial-use-cognitive-services

Eljand, K. (2021, diciembre 11). Step-by-Step Tutorial of Azure Custom Vision - Kristjan Eljand - Medium. *Medium.* https://eljand.medium.com/step-by-step-tutorial-of-azure-custom-vision-ae23541db511

Jboback, aahill, nitinme, JonVenezia, eric-urban (2023a, diciembre 19). How to perform sentiment analysis and opinion min-ing - Azure AI services. *Microsoft Learn.* https://learn.microsoft.com/en-us/azure/ai-services/language-service/sentiment-opinion-mining/how-to/call-api

Jboback, aahill, nitinme, JonVenezia, eric-urban (2023b, diciembre 19). What is the Text Analytics for health in Azure AI Language? - Azure AI services. *Microsoft Learn.* https://learn.microsoft.com/en-us/azure/ai-services/language-service/text-analytics-for-health/overview?tabs=assertion-detection

Luo, G. (2021). What visual targets are viewed by users with a handheld mobile magnifier app. *Translational Vision Science & Technology, 10* (3), 16. https://doi.org/10.1167/tvst.10.3.16

PatrickFarley y Eric-urban (2024, enero 21). Quickstart: Image classification with Custom Vision client library or REST API - Azure AI services. *Microsoft Learn.* https://learn.microsoft.com/en-us/azure/ai-services/Custom-Vision-Service/quickstarts/image-classification?tabs=windows%2Cvisual-studio&pivots=programming-language-python

PatrickFarley, nitinme y eric-urban (2024a, abril 30). Face detection, attributes, and input data - Face - Azure AI services. *Microsoft Learn.* https://learn.microsoft.com/en-us/azure/ai-services/computer-vision/concept-face-detection

PatrickFarley, nitinme y eric-urban (2024b, marzo 19). Face detection - Azure AI Vision - Azure AI services. *Microsoft Learn.* https://learn.microsoft.com/en-us/azure/ai-services/computer-vision/concept-detecting-faces

Build a Computer Vision app with Azure Cognitive Services

Popa, C. y Microsoft (s. f.). *Build a computer vision app with Azure Cognitive Services* [Curso]. Coursera. https://www.coursera.org/projects/build-a-computer-vision-app-with-azure-cognitive-services

En este curso gratuito de Coursera se desarrolla un proyecto completo relacionado con el servicio de Computer Vision de Azure AI Services. De esta forma, el alumno puede adquirir experiencia práctica en el uso de esta API para resolver un caso de uso extremo a extremo.

AI Fundamentals – Lesson 6

UDACity. Y Microsoft (s. f.). *AI Fundamentals* [Curso]. https://www.udacity.com/course/ai-fundamentals--ud099

La lección 6 de este curso gratuito de Udacity, desarrollado en colaboración con Microsoft, permite al alumno profundizar en los conceptos vistos en este tema acerca del procesamiento de lenguaje natural. En particular, se incluyen detalles relacionados con los servicios de comprensión, traducción y transcripción incluidos en Azure AI Language.

Test

1. ¿Cuál de las siguientes no es una característica del servicio Computer Vision?

A. Detección de objetos.

B. Clasificación de imágenes clínicas.

C. Reconocimiento de texto manuscrito e impreso.

D. Paleta de colores.

2. ¿En qué se diferencian los servicios Computer Vision y Custom Vision?

A. En Computer Vision se proporcionan imágenes propias para reentrenar los modelos de acuerdo con categorías personalizadas.

B. En Custom Vision se proporcionan imágenes propias para reentrenar los modelos de acuerdo con categorías personalizadas.

C. No existen ninguna diferencia, son dos nombres utilizados para referirse al mismo servicio.

D. Custom Vision proporciona una mayor precisión en la información extraída, dado que utiliza modelos con mejores procesos de entrenamiento.

3. ¿Qué características deben tener las imágenes de entrenamiento de Custom Vision?

A. Deben tener el formato JPG.

B. Deben incluir diferentes tipos de iluminación, ángulos y fondos.

C. Deben tener una resolución adecuada, de al menos 1920x1080px.

D. Deben incluir el mismo tipo de iluminación, ángulos y fondos.

4. ¿Cuál de las siguientes no es una característica del servicio Face?

A. Detección de caras similares a una dada.

B. Incorporación de filtros de imagen a las caras detectadas para corregir circunstancias habituales, como por ejemplo los ojos cerrados.

C. Agrupación de imágenes con caras de aspecto similar.

D. Comprobación de que las caras mostradas en dos imágenes pertenecen a la misma persona.

5. ¿Qué atributos incluye el JSON de resultados devuelto por **detect_with_url()** en el servicio Face?

A. Solo indica que se trata de una cara, pero no aporta ningún dato adicional.

B. Edad, sexo y coordenadas de posición.

C. Coordenadas de posición.

D. Coordenadas de posición y sexo.

6. El servicio Form Recognizer permite reconocer campos y valores de un formulario sin necesidad de entrenar un modelo específico para nuestros documentos:

A. Falso.

B. Verdadero.

7. ¿Cuáles son los principales casos de uso de la funcionalidad de clasificación personalizada de textos en Azure AI Language?

A. Resumen de libros.

B. Categorización automática de correos o tiques y buscadores semánticos.

C. Creación de chatbots.

D. Integración en Power BI.

8. ¿En qué consiste la funcionalidad de detección de entidades en Azure AI Language?

A. Permite identificar contenido ofensivo en un texto y reemplazarlo por asteriscos.

B. Permite extraer las entidades clave de un texto, tales como lugares, fechas, cantidades, nombres o marcas.

C. Permite extraer nombres de personas que aparecen en el texto.

D. Permite extraer información personal o sensible detectada en el texto.

9. ¿Cuáles son las tecnologías de Power Platform en las que pueden integrarse servicios cognitivos de Azure?

A. Power Apps y Power BI.

B. Power Apps, Power BI y Power Automate.

C. Power Automate y Power BI.

D. Los servicios cognitivos de Azure no pueden integrarse en la Power Platform.

10. ¿Cuál de los siguientes tipos de información no es identificada por la funcionalidad de detección de información personal o sensible de Azure AI Language?

A. Direcciones postales.

B. Números de identificación de asociaciones regionales.

C. Fechas.

D. Puesto que ocupa un empleado en la organización.