Documentación de Proyecto 9S AI

Joshua Sancho Leonardo Bolaños Steven Solis

Mayo 2024

1 Introducción

El proyecto de desarrollo de un asistente virtual de inteligencia artificial, inspirado en JARVIS de Iron Man, busca emular la versatilidad y eficiencia de este asistente virtual icónico. JARVIS, acrónimo de "Just A Rather Very Intelligent System" (Solo un Sistema Bastante Muy Inteligente), es el asistente personal de Tony Stark en el universo de Marvel Comics y en las películas de Iron Man.

Diseñado por Stark Industries, JARVIS es una inteligencia artificial sofisticada que aborda una amplia gama de tareas, desde el control de la mansión y los trajes de Iron Man hasta el análisis táctico y la asistencia en la investigación científica.

JARVIS se destaca por su capacidad para procesar grandes cantidades de información en tiempo real, su habilidad para comprender y responder a los comandos de voz de manera natural, y su adaptabilidad a las necesidades cambiantes de Tony Stark. Además, JARVIS puede interactuar con otros sistemas informáticos y dispositivos tecnológicos, lo que le permite controlar una amplia gama de funciones y dispositivos conectados.

Nuestro proyecto tiene como objetivo desarrollar un conjunto de modelos de inteligencia artificial y funcionalidades adicionales que permitan crear una experiencia similar a la de JARVIS. A través de la integración de tecnologías de clasificación, regresión y reconocimiento, junto con características de reconocimiento facial y comandos de audio, buscamos emular la versatilidad y eficiencia de JARVIS en un entorno práctico y aplicable.

Con este proyecto, esperamos no solo explorar las capacidades de la inteligencia artificial, sino también brindar una solución útil y emocionante para mejorar la interacción humano-máquina.

El proyecto se compone de diez modelos de inteligencia artificial, cada uno diseñado para tareas específicas de clasificación, regresión o series temporales. Cada modelo estará disponible a través de un endpoint que permitirá su integración con un frontend web desarrollado en Python. Además, el proyecto incluirá funcionalidades de reconocimiento de expresiones faciales y comandos de audio, añadiendo una capa adicional de interactividad y usabilidad al asistente virtual.

Con este enfoque, aspiramos a crear una experiencia de asistente virtual avanzada y altamente personalizable, que pueda adaptarse a una variedad de

escenarios y necesidades. Nuestro objetivo es no solo demostrar las capacidades de la inteligencia artificial de manera seria y efectiva, sino también proporcionar una herramienta útil y satisfactoria para el usuario final.

2 Solución

2.1 Backend

El backend del proyecto juega un papel fundamental en el desarrollo y funcionamiento del asistente virtual de inteligencia artificial.

Compuesto por una serie de módulos interconectados y funcionalidades específicas, el backend proporciona la infraestructura necesaria para gestionar y procesar datos, ejecutar modelos de inteligencia artificial y facilitar la interacción con el frontend web.

Este sistema está diseñado bajo una arquitectura de microservicios, que permite una mayor modularidad, escalabilidad y flexibilidad en el desarrollo y despliegue de cada componente. Cada módulo del backend se desarrolla como un servicio independiente, lo que facilita su mantenimiento y actualización individual, sin afectar el funcionamiento global del sistema.

Además, el backend del proyecto se basa en una API REST, que proporciona una interfaz uniforme para la comunicación entre los diferentes componentes del sistema. Esta API REST permite acceder y manipular recursos de manera eficiente a través de solicitudes HTTP estándar, garantizando una integración suave con el frontend web y otros sistemas externos.

A través de esta arquitectura robusta y modular, el backend del proyecto se esfuerza por garantizar un rendimiento óptimo y una experiencia de usuario fluida. A continuación, se presentarán los diferentes módulos del backend, junto con una descripción de sus funciones y características principales. Estos módulos trabajan en conjunto para ofrecer un sistema completo y eficiente que respalda las capacidades del asistente virtual de inteligencia artificial.

2.1.1 Módulo models.py

Este módulo desempeña un papel crucial en el backend del proyecto, siendo responsable de cargar, almacenar y utilizar modelos de aprendizaje automático para realizar predicciones sobre una variedad de datos. A continuación, se presenta una descripción detallada de cada función dentro del módulo:

- **__init__**: Este método inicializa la clase ModelLoader, cargando todos los modelos de aprendizaje automático disponibles al momento de la inicialización. Los modelos se almacenan en un diccionario para su posterior acceso y uso.
- **load_models**: Esta función estática carga los modelos de aprendizaje automático desde archivos pickle ubicados en un directorio específico. Utiliza los módulos os y pickle para recorrer el directorio y cargar cada modelo en un diccionario.

- **process_SARIMAX**: Este método realiza predicciones utilizando un modelo SARIMAX específico para una fecha dada. Toma el nombre del modelo y la fecha como entrada, realiza la predicción utilizando el modelo correspondiente y devuelve el valor predicho.
- **wine_prediction**: Utiliza un modelo específico para predecir la calidad del vino basándose en la acidez volátil, la densidad y el contenido de alcohol proporcionados como entrada.
- **stroke_prediction**: Predice la probabilidad de un accidente cerebrovascular utilizando un modelo específico basado en la edad, la presencia de hipertensión, enfermedades cardíacas y el nivel de glucosa en sangre del paciente.
- **pokemon_prediction**: Predice la probabilidad de que un Pokémon sea legendario utilizando un modelo específico basado en los pasos de huevo base y el porcentaje masculino del Pokémon.
- **heart_failure_prediction**: Predice la probabilidad de insuficiencia cardíaca utilizando un modelo específico basado en la fracción de eyección y el tiempo de seguimiento.
- **rug_prediction**: Predice el tipo de droga adecuada para un paciente utilizando un modelo específico basado en la edad, el sexo, la presión arterial, el colesterol y el ratio sodio-potasio del paciente.
- **breast_cancer_prediction**: Predice la probabilidad de cáncer de mama utilizando un modelo específico basado en los puntos cónicos y el perímetro del tumor.

Cada método dentro del módulo **models.py** está diseñado para realizar predicciones sobre datos específicos utilizando modelos de aprendizaje automático correspondientes. Su implementación modular y bien definida facilita la integración y el uso de estos modelos en otras partes del proyecto. Este módulo juega un papel crucial en la capacidad del sistema para ofrecer predicciones precisas y significativas en una variedad de contextos y aplicaciones.

2.1.2 Módulo fer.py

Este módulo desempeña un papel crucial en el reconocimiento de emociones en imágenes faciales. A continuación, se detallan las funciones dentro del módulo:

- **detectFaces(image)**
  - Esta función detecta caras en una imagen proporcionada utilizando el clasificador de cascada Haar.
  - La imagen se convierte a formato OpenCV y se detectan las caras utilizando el método *detectMultiScale*.
  - Cada cara detectada se mantiene en su tamaño y formato original.

- Finalmente, devuelve una lista de regiones faciales detectadas.

- **preprocess_images(images, target_size=(48, 48))**
  - Esta función realiza la preprocesamiento de una lista de imágenes para el reconocimiento de emociones.
  - 1. Convierte cada imagen a escala de grises.
    2. Cambia el tamaño de cada imagen al tamaño objetivo.
    3. Normaliza los valores de píxeles en el rango [0, 1].
    4. Convierte cada imagen a un tensor y expande sus dimensiones para que coincida con la forma de entrada del modelo.
  - Devuelve un tensor que contiene las imágenes preprocesadas.

- **emotionRecognition(faces)**
  - Esta función reconoce las emociones en las caras detectadas utilizando un modelo de clasificación de imágenes.
  - Utiliza el procesador y el modelo preentrenados de Hugging Face para el reconocimiento de emociones en imágenes faciales.
  - Cada cara detectada se procesa utilizando el modelo y se obtiene una predicción de la emoción.
  - Las predicciones se mapean a etiquetas de emociones (tristeza, disgusto, ira, neutral, miedo, sorpresa, felicidad) y se devuelven como una lista de resultados.
  - Este módulo proporciona una funcionalidad esencial para el sistema de reconocimiento de emociones en imágenes faciales. Al detectar caras y reconocer emociones, permite una comprensión más profunda de las interacciones humanas en imágenes digitales. Su implementación modular y bien definida lo hace esencial para el funcionamiento del backend del proyecto de reconocimiento de emociones.

2.2 Módulo tts.py

Este módulo proporciona funciones para transformar y transcribir archivos de audio utilizando un modelo pre-entrenado.

- **CTCLoss(y_true, y_pred)**
  - Calcula la pérdida de Connectionist Temporal Classification (CTC).
  - Útil para entrenar modelos de secuencia a secuencia cuando la alineación entre las secuencias de entrada y salida es desconocida, como en reconocimiento de voz y escritura a mano.
  - Requiere los tensores y_true y y_pred como entrada y devuelve la pérdida CTC para cada secuencia en el lote

- **decode_batch_predictions(pred)**
  - Decodifica las predicciones de salida de un modelo basado en CTC en texto legible.
  - Utiliza búsqueda codiciosa por defecto, pero puede modificarse para usar búsqueda en viga para tareas más complejas.
  - Devuelve una lista de secuencias de texto decodificado para cada elemento del lote.
- **transform_audio(input_file)**
  - Transforma un archivo de audio a un formato específico adecuado para su posterior procesamiento.
  - Resamplea el audio a 22050 Hz, lo convierte a mono (un canal) y aplica un filtro de resampleo asíncrono.
  - Guarda el audio transformado en un archivo WAV temporal y lo devuelve para su uso posterior.
- **transcribe_audio(audio_file)**
  - Transcribe un archivo de audio a texto utilizando un modelo pre-entrenado.
  - Lee el archivo de audio, calcula su espectrograma, lo normaliza y lo pasa por un modelo pre-entrenado para obtener la transcripción.
  - Devuelve la transcripción del audio como texto.

2.2.1 Módulo server.py

Este módulo es la columna vertebral de la aplicación backend y proporciona una serie de endpoints RESTful para realizar predicciones y análisis de datos. A continuación, se analiza detalladamente cada función dentro del módulo:

- **s_and_p_prediction(), ethereum_prediction(), bitcoin_prediction(), avocado_prediction()**: Estas funciones predicen el valor del índice S&P, Ethereum, Bitcoin y aguacates respectivamente para una fecha dada. Extraen la fecha de la solicitud y pasan esta fecha junto con el nombre del modelo correspondiente a model_loader.process_SARIMAX. Luego, devuelven la predicción como un objeto JSON.
- **wine_prediction(), stroke_prediction(), pokemon_prediction(), heart_failure_prediction(), drug_prediction(), breast_cancer_prediction()**: Estas funciones realizan predicciones basadas en modelos de aprendizaje automático para diferentes escenarios, como la calidad del vino, la probabilidad de accidente cerebrovascular, si un Pokémon es legendario, la probabilidad de insuficiencia cardíaca, el tipo de droga recomendado y la probabilidad de cáncer de mama, respectivamente. Extraen los

parámetros relevantes de la solicitud y los pasan a las funciones correspondientes en model_loader. Luego, devuelven las predicciones como objetos JSON.

- **transcribe_audio_route()**:Esta función implementa un endpoint RESTful para la transcripción de archivos de audio. Se encarga de recibir un archivo de audio a través de una solicitud POST, procesarlo y devolver la transcripción en formato JSON. Primero guarda el archivo subido en una ubicación temporal, luego transforma y transcribe el audio utilizando las funciones definidas en el módulo tts. Devuelve la transcripción como una respuesta JSON o un mensaje de error en caso de que ocurra una excepción.

- **process_image()**: Esta función implementa un endpoint RESTful para el reconocimiento de emociones en imágenes. Se encarga de recibir una imagen a través de una solicitud POST, detectar las caras en la imagen y reconocer las emociones en cada cara detectada. Utiliza OpenCV para leer el archivo de imagen y el módulo fer para detectar las caras y reconocer las emociones. Las emociones se devuelven como una respuesta JSON o un mensaje de error en caso de que ocurra una excepción.

Estas funciones proporcionan una interfaz fácil de usar para realizar diversas operaciones basadas en modelos de aprendizaje automático y análisis de datos. Al exponer estos endpoints RESTful, la aplicación backend puede integrarse fácilmente con otros sistemas o interfaces de usuario para proporcionar funcionalidades avanzadas de predicción y análisis.

2.3 Frontend

2.3.1 Módulo client.js

El cliente realiza solicitudes HTTP utilizando la biblioteca *Axios*, que es una popular biblioteca de JavaScript para realizar solicitudes HTTP desde el navegador o desde Node.js. Cada función en el cliente está diseñada para interactuar con un endpoint específico en el backend Flask. Aquí está cómo se realizan las solicitudes para cada función:

- **transcribeAudio(audioFile)**

  1. Esta función toma un archivo de audio como parámetro.
  2. Crea un objeto FormData y adjunta el archivo de audio a él.
  3. Luego, realiza una solicitud POST a la URL http://localhost:5000/transcribe_audio pasando el objeto FormData.
  4. Espera la respuesta del servidor y devuelve la transcripción del audio obtenida de los datos de respuesta.

- **recognizeEmotion(imageFile)**

1. Similar a transcribeAudio, esta función toma un archivo de imagen como parámetro y lo adjunta a un objeto FormData.
2. Realiza una solicitud POST a http://localhost:5000/recognize_emotion con el objeto FormData.
3. Espera la respuesta del servidor, que contiene las emociones detectadas en la imagen, y devuelve los datos de respuesta.

- **fetchSandPPrediction(inputDate), fetchEthereumPrediction(inputDate), fetchBitcoinPrediction(inputDate), fetchAvocadoPrediction(inputDate), fetchWinePrediction(volatileAcidity, density, alcohol), fetchStrokePrediction(age, hypertension, heartDisease, avgGlucoseLevel), fetchPokemonPrediction(baseEggSteps, percentageMale), fetchHeartFailurePrediction(ejectionFraction, time), fetchDrugPrediction(age, sex, bp, cholesterol, naToK), fetchBreastCancerPrediction(concavePointsWorst, perimeterWorst)**

  1. Cada una de estas funciones realiza una solicitud GET a su respectiva URL en el servidor Flask (por ejemplo, http://localhost:5000/s&p_prediction para predecir el valor del S&P index).
  2. Adjunta los parámetros necesarios a la URL de la solicitud, como la fecha de entrada para las predicciones financieras, los parámetros de salud para las predicciones médicas y los atributos para las predicciones de vino y Pokémon.
  3. Espera la respuesta del servidor y devuelve el resultado de la predicción obtenido de los datos de respuesta.

3 Arquitecturas de AI/ML

- **Regresión Lineal**:
  - La regresión lineal es un modelo estadístico que se utiliza para modelar la relación entre una variable dependiente y una o más variables independientes.
  - El objetivo es encontrar la mejor línea recta que se ajuste a los datos para predecir la variable dependiente.
  - Se utiliza principalmente para problemas de regresión, donde se predice un valor numérico.
- **Random Forest**:
  - Random Forest es un algoritmo de aprendizaje supervisado utilizado tanto para problemas de regresión como de clasificación.
  - Consiste en un conjunto de árboles de decisión, donde cada árbol se entrena con una muestra aleatoria de los datos y hace predicciones. Luego, las predicciones de todos los árboles se promedian (en regresión) o se votan (en clasificación) para obtener el resultado final.

  - Es conocido por su capacidad para manejar grandes conjuntos de datos con alta dimensionalidad y para evitar el sobreajuste.

- **SARIMAX (Seasonal AutoRegressive Integrated Moving Average with eXogenous factors)**:
  - SARIMAX es un modelo estadístico utilizado para analizar y predecir series temporales.
  - Se basa en la descomposición de una serie temporal en componentes de tendencia, estacionalidad y aleatoriedad. Incorpora términos autoregresivos, de medias móviles, así como componentes estacionales y exógenas para modelar y predecir el comportamiento de la serie temporal.
  - Es especialmente útil cuando se trabaja con datos que muestran patrones estacionales y tendencias.

- **Convolutional Neural Networks (CNN)**:
  - Las CNN son un tipo de red neuronal profunda que se utiliza comúnmente en tareas de visión por computadora, como reconocimiento de imágenes y segmentación semántica.
  - Están compuestas por capas de convolución, que aplican filtros para extraer características de las imágenes, seguidas de capas de agrupación para reducir la dimensionalidad.
  - Las CNN son capaces de aprender automáticamente jerarquías de características, lo que las hace efectivas para la extracción de características en datos de imágenes.

- **Convolutional Temporal Networks (CTC)**:
  - Las Convolutional Temporal Networks (CTC) son una arquitectura de inteligencia artificial y aprendizaje automático diseñada específicamente para el procesamiento de datos secuenciales, como series temporales o secuencias de datos.
  - Al igual que las CNN se especializan en el procesamiento de imágenes, las CTC se centran en el análisis y la predicción de secuencias temporales.
  - Estas redes están compuestas por capas de convolución temporal que aplican filtros sobre ventanas de tiempo en los datos de entrada para extraer características relevantes.
  - A menudo, se combinan con capas de agrupación temporal para reducir la dimensionalidad de la secuencia, preservando la información importante.
  - Las CTC son especialmente efectivas para capturar patrones temporales complejos en conjuntos de datos, como series temporales financieras, señales biológicas o datos de sensores.

  - Se utilizan en una variedad de aplicaciones, incluyendo pronóstico de series temporales, reconocimiento de actividad humana, análisis de voz y procesamiento de señales.

- **YOLO (You Only Look Once)**:
  - YOLO es una técnica de detección de objetos en tiempo real que es extremadamente rápida y precisa.
  - A diferencia de los enfoques tradicionales de detección de objetos, que aplican modelos de clasificación a múltiples regiones de una imagen, YOLO aplica una única red neuronal a la imagen completa.
  - Divide la imagen en una cuadrícula y predice simultáneamente las cajas delimitadoras y las probabilidades de clase para cada celda de la cuadrícula.
  - YOLO es conocido por su capacidad para realizar detección de objetos en tiempo real, lo que lo hace ideal para aplicaciones que requieren velocidad, como la vigilancia por video, la conducción autónoma y los sistemas de seguridad.
  - Las versiones más recientes de YOLO (e.g., YOLOv8, YOLOv9) han mejorado aún más la precisión y la eficiencia del modelo.

4 Análisis de Resultados Obtenidos

Modelo	**Métricas**	**Resultado**
S&P	MAE: 0.35 MSE: 0.33 RMSE: 0.57	Muy bueno
Ethereum	MAE: 115.43 MSE: 22212.87 RMSE: 149.03	Muy bueno
Bitcoin	MAE: 21.83 MSE: 2376.24 RMSE: 48.74	Muy bueno
Avocado	MAE: 0.01 MSE: 0.0007 RMSE: 0.02	Muy bueno
Wine	Accuracy: 0.83	Bueno
Stroke	Accuracy: 0.91	Muy bueno
Pokemon	Accuracy: 0.99	Excelente
Heart Failure	Accuracy: 0.85	Muy bueno
Drug	Accuracy: 0.98	Excelente
Breast Cancer	Accuracy: 0.94	Excelente

Table 1: Resultados obtenidos en evaluación de modelos de Machine Learning

5 Análisis de Completitud de Proyecto

6 Anexos

- Carpeta del Proyecto en Drive

Modelo	**Métricas**	**Resultado**
FER CNN	$\text{val}_loss : 0.55 accuracy : 0.70$	Bueno
TTS CTC	wer: 0.65	Bueno
YOLOv8m	mAP50-95: 0.70	Bueno

Table 2: Resultados obtenidos en evaluación de modelos de Deep Learning

Objetivo	Porcentaje(%)
Modelos de IA/ML	100%
Modelos de Deep Learning	100%
Endpoints	100%
Recocimiento Facial de Expresiones	100%
Comandos por Voz	90%
Reconocimiento de Objetos	100%
Interfaz	100%
Total	100%

Table 3: Tabla de Completitud de Proyecto

- Whisper AI
- Yolov8