OSINT InfoHound

Síntesis de datos de fuentes abiertas por medio de modelos de lenguaje de gran tamaño (LLM)

Marcos Casado Herrero

Máster en Ciberseguridad y Privacidad
Seguridad Empresarial

Nombre Tutor de TFM
Jordi Guijarro Olivares
Profesor Responsable de la Asignatura
Víctor García Font

Universitat Oberta de Catalunya

09/01/2024

FICHA DEL TRABAJO FINAL

Título del trabajo:	*OSINT Infohound – Síntesis de datos de fuentes abiertas por medio de modelos de lenguaje de gran tamaño (LLM)*
Nombre del autor:	*Marcos Casado Herrero*
Nombre del consultor/a:	*Jordi Guijarro Olivares*
Nombre del PRA:	*Víctor García Font*
Fecha de entrega (mm/aaaa):	*01/2024*
Titulación o programa:	Máster en Ciberseguridad y Privacidad
Área del Trabajo Final:	*Seguridad Empresarial*
Idioma del trabajo:	*Castellano*
Palabras clave	*OSINT, AI, LLM*

Resumen del Trabajo

En este trabajo se analiza la viabilidad de la integración de modelos de lenguaje de gran tamaño (LLM) en plataformas de recolección de información de fuentes abiertas (OSINT) con el objetivo de mejorar la eficiencia de búsqueda y análisis de este tipo de información. Se argumenta que los LLM pueden aportar versatilidad, flexibilidad y valor añadido a estas plataformas, permitiendo buscar, analizar y sintetizar grandes cantidades de datos.

Para ilustrar esta idea, se presenta un caso práctico de integración de un LLM en la plataforma InfoHound. InfoHound es una herramienta del instituto de investigación e innovación i2cat de Catalunya que permite, entre otras cosas, a las organizaciones realizar el análisis inverso de la información que se ha indexado sobre ellas en Internet. La integración de un LLM en esta plataforma abriría un gran abanico de oportunidades, permitiendo, por ejemplo, sintetizar los curriculum vitae de las personas asociadas a una organización, o clasificar personas por su pensamiento político derivado de la información de redes sociales. El ejemplo de caso práctico presentado durante este trabajo y demostrado sobre InfoHound consiste en recopilar perfiles de usuarios de fuentes abiertas y guardar los datos de estos tal y como se encuentran en la red, con distintos formatos. De tal manera que, a partir de esta información desorganizada, se le pueda pedir a un contenedor de modelos LLM que busque, filtre y sintetice la información generando un resumen profesional para cada persona recopilada.

Abstract

This thesis analyses the feasibility of integrating large language models (LLMs) into open-source intelligence (OSINT) collection platforms, with the aim of improving the efficiency of open-source intelligence analysis. It is argued that LLMs can add versatility, flexibility, and value to these platforms, enabling the search, analysis, and synthesis of large amounts of data.

To illustrate this idea, a practical case of integrating an LLM into the InfoHound platform is presented. InfoHound is a tool from the research and innovation institute i2cat of Catalunya that allows organisations to perform reverse analysis on information indexed about them. Integrating an LLM into this platform would open up a wide range of possibilities, such as synthesising the CVs of individuals associated with an organisation, or classifying individuals based on their political thinking derived from social media information. The practical case study applied to InfoHound consists of collecting user profiles from open-sources and storing their data in a disorganised way, with different formats or sources, so that later an LLM model container can be asked to analyse the information and generate a professional summary for each person collected.

Índice

Lista de tablas

Lista de figuras

1. Introducción

Uno de los principales vectores de ataques y por lo tanto vulnerabilidad de un entorno empresarial es a través de las personas que lo conforman y la información que se conoce de este. Los componentes de una organización publican tanto de manera voluntaria como involuntaria información personal y profesional en Internet que de ser analizada de la manera correcta podría ser de utilidad para un atacante.

La inteligencia de fuentes abiertas (también conocida como OSINT por sus siglas en inglés) nos permite recopilar y analizar toda esa información que en algún momento se ha indexado en la web.

1.1. Contexto y justificación del trabajo

Este proyecto pretende contribuir y analizar la integración de tecnologías emergentes, en concreto lenguajes de gran tamaño (LLMs), en la plataforma InfoHound.

InfoHound es un proyecto en desarrollo del instituto de investigación e innovación i2cat que permite a las organizaciones realizar el análisis inverso de la información que se ha indexado sobre ellas y así poder prepararse y protegerse frente a posibles amenazas. Esta herramienta cuenta con un módulo de adquisición que permite, de manera pasiva, obtener información relacionada con un dominio dado (subdominios, e-mails, personas, documentos, etc.). Una vez recopilada la información, además, InfoHound permite hacer uso de sus módulos internos de explotación y análisis de la información con el fin de aplicar funciones de análisis a medida o buscar más información relacionada con la ya adquirida.

InfoHound recopila gran cantidad de información, lo cual es un aspecto muy positivo en este tipo de plataformas OSINT, sin embargo, esta información no se encuentra en su mayoría de forma organizada o es demasiado extensa para que un analista pueda sacar conclusiones preliminares. Por ello, este proyecto pretende analizar cómo podrían ayudar las últimas tendencias en modelos de inteligencia artificial, también conocidos como modelos de lenguaje de gran tamaño (LLM) dentro de los procesos de análisis de la información propios de una herramienta OSINT como InfoHound.

A modo de ejemplo, si quisiéramos desarrollar un modelo a medida que analice e interprete el currículum vitae de las personas encontradas, asociadas a una organización, resumiendo en un párrafo su puesto actual y actividad, sería una tarea compleja y costosa. Mientras que, pedirle a un LLM que lo haga se convierte en una tarea relativamente sencilla.

De esta manera, este proyecto nace de la idea de que la integración de un LLM en InfoHound aportaría flexibilidad y valor a la plataforma, permitiendo sintetizar grandes cantidades de datos para facilitar el análisis de estos a los usuarios. Así mismo, se es consciente de que una vez integrado un LLM el abanico de posibilidades que se abren

va mucho más lejos de lo anteriormente expuesto y por ello también se analizaran otras aplicaciones.

1.2. Antecedentes

Este trabajo complementa el trabajo de final de máster de la Universitat Oberta de Catalunya realizado por Xavier Marrugat Plaza y supervisado por Jordi Guijarro Olivares con título "*InfoHound tool - Improving OSINT open source CyberArsenal for Good*". Se recomienda la lectura de este documento incluido en la bibliografía para una completa comprensión del funcionamiento de la herramienta InfoHound.

1.3. Objetivos del trabajo

El objetivo principal de InfoHound surge de la necesidad de responder a la siguiente pregunta: "*Si no podemos recopilar y analizar toda la información que hay en la red sobre una organización, ¿Cómo podemos determinar su nivel de exposición?*" De esta manera, InfoHound se ofrece como una herramienta alternativa para que las organizaciones puedan conocer mejor su nivel de exposición en la red. Además de contribuir a este objetivo principal, este proyecto de final de máster pretende:

1. Presentar las distintas opciones y tendencias en modelos de lenguaje de gran tamaño (LLM) y sus características.
2. Analizar las posibles aplicaciones de los modelos LLM dentro de InfoHound tomando como caso la síntesis de datos de fuentes abiertas (OSINT).
3. Integrar y validar un modelo LLM contribuyendo al desarrollando de la herramienta InfoHound de la Fundación i2CAT.

1.4. Impacto en sostenibilidad, ético-social y de diversidad

Casi cualquier información que se desee se puede encontrar en internet si se busca con tiempo y en detalle. Esta tarea se puede simplificar mediante el uso de herramientas de búsqueda de información de fuentes abiertas que automatizan los procesos de búsqueda.

Existen herramientas especializadas en un tipo de adquisición de información o de una única fuente, mientras que InfoHound pretende distinguirse por ser una herramienta que adquiera distintos tipos de información de distintas fuentes, fusionando en interrelacionando los datos para facilitar su posterior explotación.

A nivel social InfoHound puede tener un impacto importante:

- Los usuarios podrán investigar y concienciarse sobre lo que está expuesto en internet sobre ellos y sus organizaciones.
- Los usuarios tendrán más cuidado con lo que suben a la red y la gestión de sus perfiles en RRSS.
- Las organizaciones podrán controlar la información que los expone.

- Las organizaciones reducirán el número de brechas de seguridad y ataques.
- Los datos de los usuarios estarán más seguros.

Por otro lado, y no menos importante, tampoco se debe dejar pasar por alto que este tipo de herramientas podrían convertirse todo lo contrario de para lo que se han diseñado. Bien usadas por las organizaciones supondrían un avance en seguridad, pero, también podrían ayudar a un usuario con malas intenciones. Este debate ético existe aplicado a muchos ámbitos y hay que confiar en el buen uso de las herramientas.

1.5. Enfoque y estructura del documento

El documento se divide en cuatro grandes bloques con el objetivo de guiar al lector a lo largo del análisis y la integración que se proponen en este trabajo de final de máster.

1. El primer bloque, del que es parte esta sección, introduce los aspectos más generales del trabajo tratando temas como la motivación y los objetivos que se buscan conseguir.
2. El segundo bloque es el que contiene la mayoría de los aspectos técnicos. Introduce la plataforma InfoHound, la arquitectura y despliegue de la plataforma, los LLMs y otras tecnologías a integrar, así como las funcionalidades desarrolladas.
3. El tercer bloque presentara de manera resumida los resultados que se han obtenido durante el análisis realizado en el bloque anterior.
4. Finalmente, el cuarto bloque tratará tanto las conclusiones del trabajo como las oportunidades de mejora o posibles trabajos a futuro.

1.6. Breve sumario de productos obtenidos

Este trabajo de final de máster abarca el desarrollo e integración de un módulo LLM y su aplicación en un caso práctico en la existente herramienta InfoHound del instituto de investigación e innovación i2cat. La contribución se hará efectiva en el repositorio de GitHub de la propia herramienta.

1.7. Metodología

La metodología seleccionada para la ejecución del proyecto es una metodología ágil. El motivo de selección de esta metodología se debe a la necesidad de un marco de trabajo flexible que permita adaptarse a los cambios y necesidades del proyecto. Se deberá establecer un seguimiento continuo entre las partes interesadas y mediante iteraciones reconducir los esfuerzos para la consecución de la solución que se adapte mejor a las necesidades del proyecto.

Las metodologías ágiles se apoyan en priorizar la respuesta ante el cambio sobre seguir un plan. Hay un objetivo, pero tanto este como la manera de conseguirlo pueden ir cambiando durante el desarrollo del proyecto.

Para lograr el funcionamiento de esta metodología es importante establecer un marco de trabajo y también establecer unas fechas, así como hitos de revisión y entregas. La siguiente sección introduce la planificación acordada entre las partes interesadas.

1.8. Planificación del proyecto

El proyecto se desarrolla en las siguientes fases, las cuales por su naturaleza se tienen que ceñir a unos tiempos y a unas entregas concretas:

1. Planificación.
2. Desarrollo de la idea.
3. Análisis e implementación.
4. Documentación, redacción y revisión.
5. Presentación de resultados.

Durante estas fases se establecen tanto hitos oficiales de entrega como revisiones periódicas con las partes interesadas a modo de seguimiento y con la idea de recibir opiniones sobre el desarrollo. Estas reuniones de seguimiento se irán fijando a demanda en función de las necesidades de cada momento.

La siguiente figura sintetiza la planificación del proyecto a modo de cronograma simplificado donde se marcan los hitos entregables como estrellas y se da una visión temporal de las fechas importantes, así como las principales fases del proyecto anteriormente expuestas.

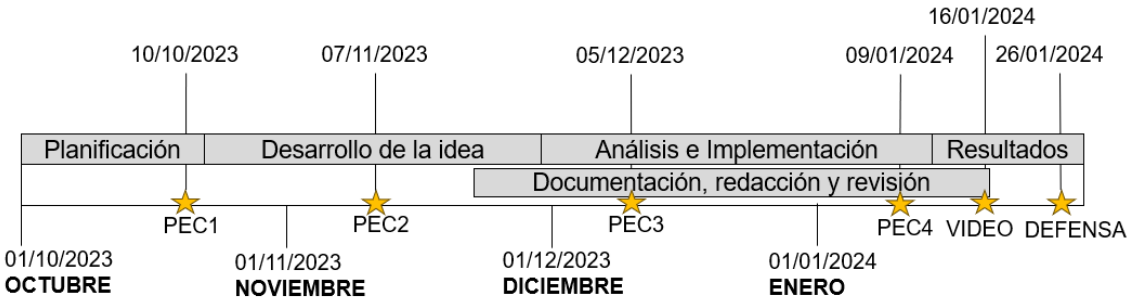

Ilustración 1 – Conograma del proyecto

Se proporciona más detalle de las actividades en un diagrama de Gantt adjunto en el Anexo II de este documento.

2.1. Riesgos

Los riesgos de proyecto se analizarán dentro de las siguientes categorías:

- Técnicos (relacionados con la tecnología):
    - Fallos de hardware sobre las máquinas de prueba.
    - Fallos del software desarrollado o librerías de terceros.
    - Falta de madurez de las tecnologías empleadas.
    - Falta de documentación.
- De gestión (relacionados con la gestión del proyecto):
    - Planificación inadecuada.
    - Falta de comunicación.
    - No definir un objetivo realista y asequible en tiempo.
    - Retrasos por materialización de riesgo.
- Externos (fuera del control del equipo del proyecto)
    - Cambios de legislación.
    - Cambios en las políticas de las librerías de terceros empleadas.

Estos riesgos deben de ser monitorizados constantemente a lo largo del proyecto y en un caso ideal se debería tener un plan de contingencia para cada uno de ellos por si se materializasen. Durante las reuniones de seguimiento con las partes interesadas se revisará el correcto avance del proyecto y el estado de los riesgos.

2. Métodos y desarrollo

Este proyecto busca analizar la integración de los lenguajes de gran tamaño (LLM) dentro de la plataforma InfoHound del instituto de investigación e innovación i2cat de Catalunya. Para ello, este capítulo de análisis de los métodos y desarrollo se divide de la siguiente manera:

1. Modelo OSINT y su aplicación al proyecto (capítulo 2.2): Introducirá el modelo OSINT en el que se basa la herramienta a modo de presentación y justificación de este tipo de herramientas.
2. InfoHound (capítulo 2.3): Introducirá los aspectos fundamentales de la propia herramienta InfoHound (Tecnologías empleadas, arquitectura, etc.).
3. Análisis, integración y despliegue del LLM en InfoHound (capítulo 2.4): Analizarán las distintas opciones de LLMs disponibles para integrar dentro de la plataforma, considerando los aspectos positivos y negativos de cada uno de ellos. De este análisis saldrá un LLM candidato a ser integrado dentro de InfoHound y se presentará la metodología empleada para el despliegue e integración del LLM dentro de InfoHound.
4. Búsqueda de información de fuentes abiertas (Google API JSON) (capítulo 2.5): Presentará la función de adquisición de información de Google implementada para recuperar los perfiles que posteriormente serán analizados.
5. Síntesis de datos de fuentes abiertas por medio del LLM (capítulo 2.6): Presentará las llamadas al LLM para analizar los datos en crudo rescatados de Google, así como los resultados de estas funciones.

Todos estos capítulos pretenden guiar a lo largo del análisis realizado con el objetivo comprender y dar a conocer el ámbito de la inteligencia de fuentes abiertas, las tecnologías existentes y aplicables, analizar las alternativas, así como justificar su elección y argumentar las decisiones tomadas.

2.2. Modelo OSINT y su aplicación al proyecto

El Instituto Nacional de Ciberseguridad define OSINT como el "*Conocimiento recopilado a partir de fuentes de acceso público. El proceso incluye la búsqueda, selección y adquisición de la información, así como un posterior procesado y análisis de la misma con el fin de obtener conocimiento útil y aplicable en distintos ámbitos.*" [2] Existen muchas fuentes para la recolección de información como pueden ser revistas, páginas web, redes sociales, blogs, etc., y los objetivos perseguidos comúnmente son:

- Estudios de investigación (tendencias de mercados, sociológicos, etc.).
- Auditoria de organismos con el fin de evaluar el nivel de exposición tanto de su infraestructura como de sus empleados.
- Reputación online de usuarios y empresas.

- Prevención de posibles amenazas en el ámbito militar o de la seguridad nacional.

Así mismo, la metodología OSINT sigue el ciclo de vida de elaboración de inteligencia. Este ciclo representa la inteligencia como proceso que se puede modelar de manera simplificada en las siguientes cuatro fases [2]:

1. Dirección: Esta fase engloba la planificación, definición de fuentes de información y estudio de requisitos.
2. Obtención: Esta fase en la que se obtiene la información de las fuentes definidas en el punto anterior en este caso por medio de técnicas OSINT.
3. Elaboración: Esta fase consiste en dar formato estructurado a la información. Se pueden aplicar técnicas de análisis automatizadas para cuantificar datos, generar estadísticas y extraer conclusiones.
4. Difusión: Esta fase consiste finalmente en la presentación clara y ordenada de los productos generados a las partes interesadas.

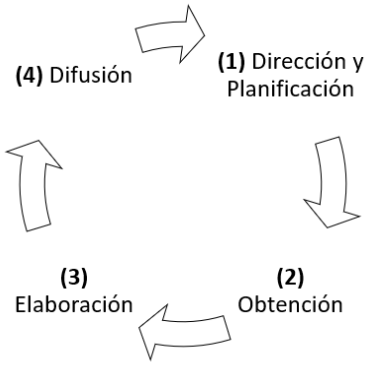

Ilustración 2 - Ciclo de inteligencia

La herramienta InfoHound, permite dar cobertura a todas las fases del ciclo de inteligencia. Así mismo los módulos a desarrollar cumplen con este modelo de ciclo:

- La fase de dirección la ejecutará el usuario indicando al módulo desde el panel de control el dominio a analizar.
- La fase de obtención constará de un análisis de perfiles de personas mediante el uso de la API JSON de Google.
- La fase de elaboración se hará por medio del LLM integrado.
- Y finalmente la fase de difusión se cubre en el mismo panel de InfoHound permitiendo adicionalmente la extracción de los datos.

Dado que se cuenta ya con una extensa base desarrollada en InfoHound, este proyecto se centra en la parte de análisis de datos y elaboración de inteligencia por medio de la integración del LLM. Además de contribuir con una nueva función de obtención.

2.3. InfoHound

Este capítulo da a conocer la herramienta InfoHound, realizando una breve descripción de sus tecnologías, arquitectura, datos y posibilidades, con la idea de que el lector comprenda la base de la integración que se realiza en este trabajo.

2.3.1. ¿Qué es InfoHound?

InfoHound es una herramienta para la recolección y el análisis de datos de fuentes abiertas desarrollada por el instituto de investigación e innovación i2cat de Catalunya. Esta herramienta ha sido diseñada para ser desplegada de manera sencilla y proporciona una interfaz web para facilitar la interacción con los usuarios. Su código e instrucciones pueden ser encontrados en el repositorio de GitHub.

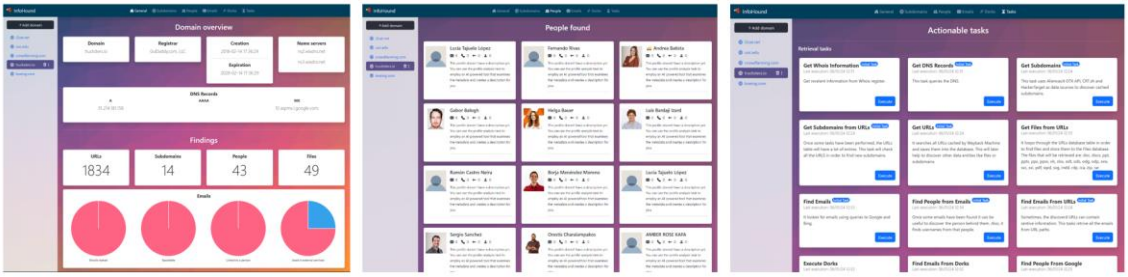

Ilustración 3 – Vistas generales de aplicación web InfoHound

InfoHound responde al ciclo de inteligencia expuesto en el anterior capítulo y por ello cuenta tanto con módulos de recolección como de explotación de la información.

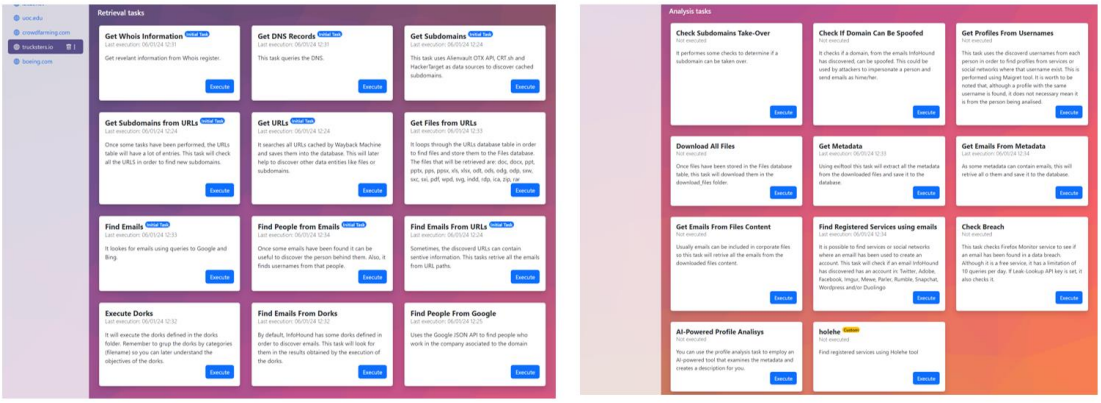

Ilustración 4 – Vistas tareas ejecutables

Las vistas de tareas ejecutables estan diseñadas para ser el panel de control desde el que lanzar las tareas de recolección o análisis sobre un dominio previamente definido (por ejemplo, “i2cat.net”).

Las tareas de recolección permiten encontrar fragmentos de información en la red tales como direcciones de correo asociadas al dominio, información del propio dominio

(subdominios, dns, etc.), direcciones URL del dominio, personas asociadas a las organización y correos electronicos o archivos.

Por otro lado las tareas de análisis permiten ejecutar funciones con el objetivo de extraer conclusiones tales como si un email recolectado ha sido comprometido, buscar nombres de usuarios de redes sociales asociados a emails encontrados, extraer los metadatos o descargar archivos.

De esta manera se podría visualizar la información recopilada en la plataforma web para realizar un análisis preliminar.

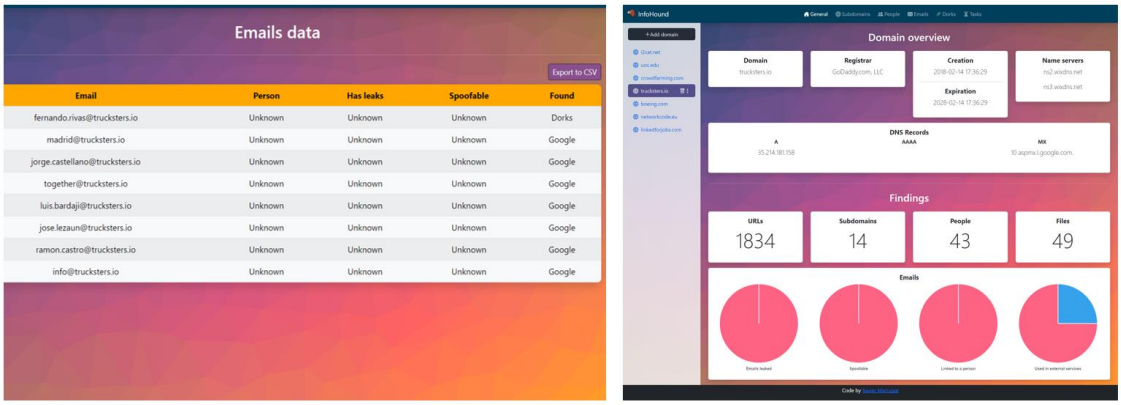

Ilustración 5 – Vistas de emails y dashboard

Adicionalmente InfoHound también permite extraer los datos en múltiples formatos para analizarlos o para que sirvan de entrada a otras plataformas OSINT.

2.3.2. Tipos de datos manejados por InfoHound

A la hora de desarrollar InfoHound se pensó en los datos que podrían ser útiles desde el punto de vista de una auditoría de seguridad y por ello se crearon los siguientes modelos de datos:

- Domain
- Subdomain
- IPs
- People
- Files
- Emails
- URLs
- Credentials

Todos estos modelos interrelacionados y dependientes del modelo principal del que parte todo el proceso que es el dominio (“Domain”).

2.3.3. Tecnologías de InfoHound

InfoHound está preparado para ser desplegado con una arquitectura de contenedores en Docker. Cada uno de estos contenedores alberga una aplicación que cumple con un propósito especifico.

2.3.3.1. Docker

InfoHound se despliega sobre Docker, una plataforma diseñada para facilitar la creación, implementación y ejecución de aplicaciones en contenedores [3].

Los contenedores son entornos ligeros y portátiles que encapsulan una aplicación y sus dependencias, asegurando consistencia entre diferentes entornos de desarrollo, pruebas y producción. Docker utiliza tecnologías de virtualización a nivel de sistema operativo para crear estos contenedores, permitiendo así que las aplicaciones se ejecuten de manera eficiente y consistente en cualquier entorno, independientemente de las diferencias de infraestructura subyacente [3]. Esto simplifica el desarrollo y despliegue de aplicaciones al proporcionar un entorno aislado y autocontenido que facilita la gestión de dependencias y la escalabilidad de aplicaciones [3].

2.3.3.1.1. Despliegue de InfoHound en contenedores Docker

Para el despliegue de la aplicación bastaría con tener instalado Docker, descargarse el repositorio de GitHub y ejecutar el comando de generación de los contenedores de Docker en la raíz de la aplicación, el cual buscará el archivo *docker-compose.yml* para instanciar los contenedores de aplicaciones definidos a partir de las imágenes referenciadas.

```
docker-compose up -d
```

La siguiente figura resume la arquitectura de los contenedores una vez creados con la configuración definida en el archivo *docker-compose.yml.*

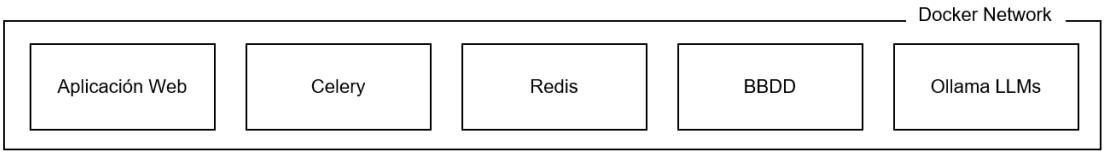

Ilustración 6 – Arquitectura de contenedores

Durante la creación de los contenedores se creará red virtual cuyo propósito es proporcionar una forma de comunicación entre ellos permitiendo así que estos puedan ejecutar peticiones cruzadas tales como ejecutar funciones remotas para el procesamiento de datos.

La aplicación desplegará los siguientes contenedores explicados a continuación, cada uno de ellos con una configuración especifica:

- web:
    - Propósito: Este contenedor ejecuta la aplicación web basada en Django. Django es un marco de desarrollo web de alto nivel y de código abierto en Python que facilita la creación de aplicaciones web robustas y escalables. Usa el modelo MVC (Modelo-Vista-Controlador).
    - Configuración Docker:
        - Expone el puerto 8000 para acceder a la aplicación.
        - Compila y ejecuta el código fuente.
        - Depende de otros servicios como la base de datos (db), Redis, el trabajador Celery, y el contenedor de IA "ollama" montado en el contexto de este trabajo.
        - Configura variables de entorno relacionadas con la base de datos (PostgreSQL) y Redis.
        - Ejecuta comandos para realizar migraciones, configurar la base de datos y ejecutar el servidor Django.
- celery_worker:
    - Propósito: Este contenedor ejecuta un trabajador Celery para procesamiento de tareas en segundo plano. Celery es una herramienta de procesamiento en segundo plano en Python que se utiliza para ejecutar tareas de manera distribuida y asíncrona. Su propósito principal es permitir la ejecución de operaciones que toman tiempo de manera eficiente y en paralelo, sin bloquear la ejecución del programa principal.
    - Configuración Docker:
        - Dependiente de la base de datos (db) y Redis.
        - Configura variables de entorno relacionadas con la base de datos (PostgreSQL) y Redis.
        - Ejecuta el comando Celery para iniciar un trabajador que escucha eventos y realiza tareas en segundo plano.
- redis:
    - Propósito: Contenedor que ejecuta el servicio Redis. Redis es un sistema de almacenamiento en memoria de código abierto que funciona como un almacén de estructuras de datos clave-valor. En el contexto de una integración con Celery, Redis se utiliza como un broker o canal de mensajería para la comunicación entre la aplicación principal y los trabajadores Celery.
    - Configuración Docker:
        - Utiliza la imagen oficial de Redis.
        - Expone el puerto 6378 para acceder a Redis.

- db:
    - Propósito: Contenedor que ejecuta el servicio PostgreSQL. PostgreSQL es un sistema de gestión de bases de datos relacional de código abierto y altamente extensible.
    - Configuración Docker:
        - Utiliza la imagen oficial de PostgreSQL versión 12.
        - Configura variables de entorno relacionadas con la base de datos (usuario, contraseña, nombre de la base de datos).
        - Utiliza un volumen llamado "postgres_data" para persistir los datos de PostgreSQL.
- ollama (new):
    - Propósito: Contenedor que ejecuta la imagen ollama/ollama. Ollama es una herramienta que permite a los utilizar modelos de lenguaje de gran tamaño (LLMs) en las aplicaciones. La librería proporciona una interfaz sencilla y fácil de usar para acceder a los LLMs, lo que permite a los desarrolladores centrarse en sus aplicaciones sin tener que preocuparse por la complejidad de los LLMs.
    - Configuración Docker:
        - Utiliza la imagen "ollama/ollama:latest".
        - Expone el puerto 11434 para acceder a el contenedor de LLMs para peticiones externas.
    - Información: Este contenedor ha sido añadido en el contexto de este trabajo.

Una vez arrancados los contenedores, la interface de Dockers permite monitorizar sus salidas desde su consola simplificando las tareas de desarrollo y debug.

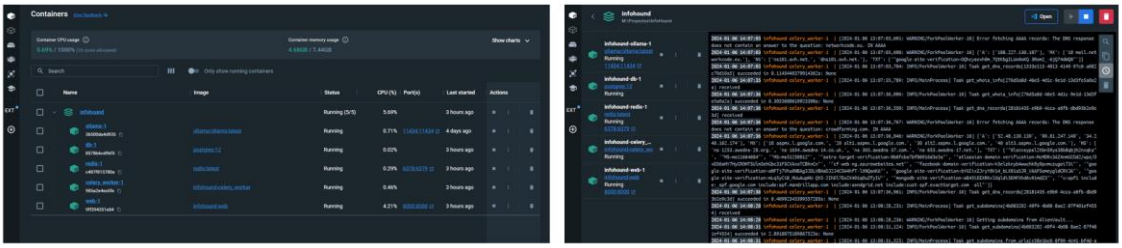

Ilustración 7 – Entorno Docker con contenedores arrancados

Desde el punto de vista de la ciberseguridad Docker proporciona grandes ventajas a los desarrollos tales como el aislamiento de recursos, el monitoreo y detección de vulnerabilidades o el uso de imagenes oficiales con actualizaciones de seguridad y soporte continuo.

2.3.3.2. Aplicación web

La aplicación web de InfoHound es el cerebro de la herramienta donde se define la lógica central que hace uso del resto de recursos. Esta aplicación se ha creado sobre el framework web Django dependiente del lenguaje de programación Python.

2.3.3.2.1. Framework Django

Django es un framework de desarrollo web gratuito y de código abierto escrito en Python. Está basado en el patrón de arquitectura MVC (Modelo-Vista-Controlador) y proporciona una serie de características y herramientas que facilitan el desarrollo de aplicaciones web complejas [4]. Un ejemplo de su potencia es que Instagram, Spotify, Pinterest y Facebook están parcial o totalmente desarrolladas en Django.

Django es una herramienta útil para aplicaciones web de OSINT como lo es InfoHound por varias razones. En primer lugar, es un framework robusto y escalable que puede manejar grandes cantidades de datos. Y, en segundo lugar, proporciona una serie de herramientas y bibliotecas que pueden ayudar a los desarrolladores a recopilar, almacenar y analizar datos de fuentes abiertas [4].

Django usa modelos para almacenar los datos y estos se gestionan de una manera muy transparente para los desarrolladores, proporcionando una capa de alto nivel que permite abstraerse de las interacciones típicas con la base de datos. Esto permite desarrollar de manera muy rápida dado que una vez integrada la base de datos (en este caso PostgreSQL) solo se tendrían que ejecutar funciones de alto nivel para filtrar, actualizar, guardar, crear y borrar modelos.

Adicionalmente este entorno de trabajo permite enlazar de manera muy sencilla las peticiones web con las vistas creadas para retornar al usuario la información requerida y a su vez hacer las llamadas necesarias a los controladores que permiten tanto la recolección como la manipulación y el análisis de los datos.

A pesar de sea la aplicación web el centro de la lógica de la herramienta, esta necesita hacer uso de otros recursos como la base de datos o el planificador de tareas dando como resultado una arquitectura de software integrada, la cual se explica en el siguiente capítulo.

2.3.3.2.2. Arquitectura software

Como ya se ha mencionado anteriormente InfoHound funciona sobre unos cimientos en Django sobre los cuales emergen varios pilares claves para el funcionamiento de la aplicación, estos son:

1. Celery con Redis para ejecutar tareas de recolección de datos y análisis de manera asíncrona en segundo plano.
2. PostgreSQL para almacenar y gestionar la información.

3. Un contenedor para realizar consultas a un modelo de lenguajes de gran tamaño a modo de sistema caja negra.

Cada uno de los pilares sirve a un proposito para el funcionamiento de la aplicación definiendo el siguiente concepto de arquitectura software:

- Arquitectura General:
    - La aplicación web está construida con Django, que maneja las solicitudes del usuario, gestiona la autenticación y proporciona la interfaz de usuario.
        - Los usuarios pueden monitorear el progreso de sus tareas a través de la interfaz web de Django.
        - Los resultados de las tareas, una vez completadas, están disponibles para su visualización a través de la interfaz.
    - Celery se utiliza para ejecutar tareas en segundo plano de forma asíncrona. Django es el responsable de lanzar las tareas a Celery a través de Redis.
    - Redis se utiliza como broker de mensajes para la comunicación entre la aplicación Django y Celery.
    - Los resultados de las tareas de recolección y análisis se almacenan en una base de datos PostgreSQL. Django se integra con PostgreSQL para realizar operaciones de lectura y escritura en la base de datos.

Así mismo, a continuación se muestran algunos ejemplos de como la arquitectura sirve a las funcionalides de la herramienta para llevar a cabo las tareas:

- Tareas de recolección de datos:
    - Los usuarios interactúan con la interfaz web para iniciar tareas de recolección de datos. Estas tareas podrían incluir la búsqueda de información en fuentes abiertas.
    - Cuando se inicia una tarea, la aplicación Django encola la tarea en Celery utilizando el sistema de mensajes proporcionado por Redis.
    - Celery ejecuta las tareas en segundo plano, permitiendo que la aplicación web siga siendo receptiva para los usuarios.
- Tareas de análisis:
    - Las tareas de análisis pueden incluir procesamiento de datos recopilados, identificación de patrones, busqueda de datos, etc.
    - Estas tareas también se encolan en Celery para su ejecución asíncrona, evitando bloquear la aplicación principal durante operaciones intensivas.

Uno de los puntos fuertes de la arquitectura es su escalabilidad gracias a la capacidad de Celery para definir nuevas tareas y distribuirlas en nodos de trabajo (si fuese necesario) y a la base de datos PostgreSQL, que puede gestionar grandes cantidades de datos.

Finalmente, el siguiente diagrama ilustra tanto la arquitectura software establecida en InfoHound como las interacciones entre los elementos.

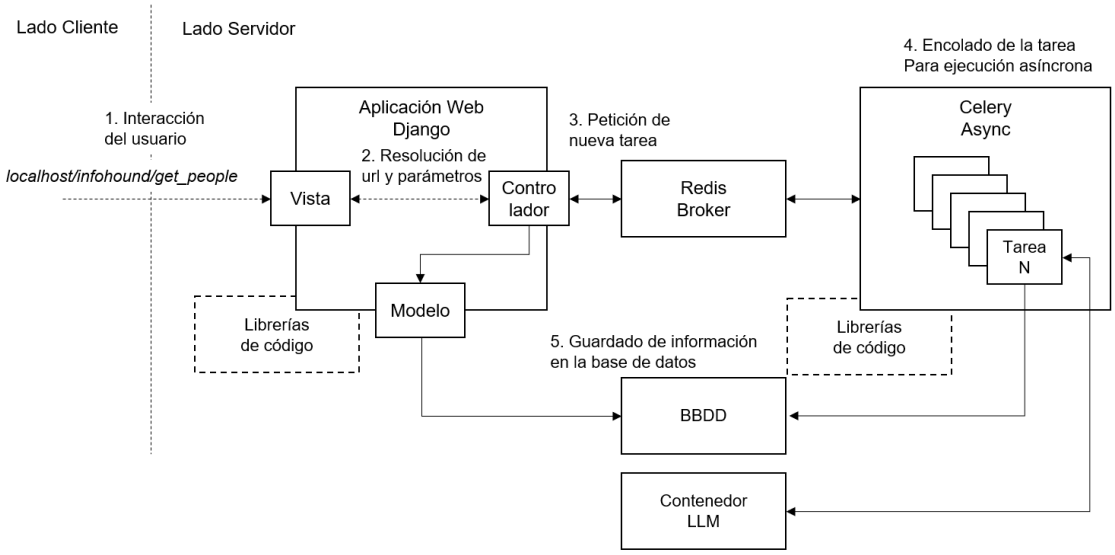

Ilustración 8 – Arquitectura InfoHound

2.3.4. ¿Dónde puedo encontrar más información de InfoHound?

Para información más detallada sobre el funcionamiento de la herramienta ver el capítulo 1.2 Antecedentes o visitar su repositorio de GitHub:

- InfoHound I2CAT: https://github.com/Fundacio-i2CAT/InfoHound

2.4. Análisis, integración y despliegue del LLM en InfoHound

Una vez introducida la herramienta InfoHound se va a analizar la incorporación de capacidades de inteligencia artificial por medio de la integración de lenguajes de gran tamaño (LLMs). Cabe destacar que, aunque se haya mencionado en alguno de los capítulos o diagramas anteriores, esta es una capacidad con la que InfoHound no contaba anteriormente.

Para la integración de capacidades LLM se ha analizado tanto la opción de hacerlo de manera remota como local. Llegando a unas conclusiones que se expondrán en los consiguientes capítulos. Esta capacidad por definir e integrar es la que se ha representado en la arquitectura como “Contenedor LLM”.

2.4.1. ¿Qué es un LLM y cómo funciona?

Un LLM, o modelo de lenguaje de gran tamaño, es un modelo de aprendizaje automático que se ha entrenado en un conjunto de datos masivo de texto y código. Los LLMs pueden verse como una rama de la inteligencia artificial que combina diversas técnicas tales como [5]:

- Aprendizaje automático (o Machine Learning): El aprendizaje automático permite a los LLM aprender las relaciones entre las palabras y los conceptos. Para este cometido, los LLMs se entrenan en un conjunto de datos masivo de texto y código.
- Procesamiento del lenguaje natural (NLP): Utilizando técnicas de NLP para interpretar texto, generar texto, traducir idiomas, escribir diferentes tipos de contenido creativo y responder a preguntas de forma informativa.
- Redes neuronales: Los LLM se basan en una arquitectura de red neuronal llamada transformador. Los transformadores son capaces de aprender relaciones complejas entre las palabras, lo que los hace muy eficientes para las tareas de NLP.

Los LLM son modelos en auge de gran complejidad. Para el alcance de este trabajo nos interesaría modelarlo como un sistema de caja negra al cual le entregamos cierta información y nos devuelve una respuesta a lo que le hemos pedido. Estas peticiones se hacen por medio de instrucciones (también conocidas como “prompts”) para guiar su generación de respuestas o resultados [5] [6]. Al proporcionar un *prompt*, se condiciona al modelo de IA para que genere una salida coherente y relevante en función de lo que el usuario necesita.

Hoy en día los LLM se están utilizando en una gran variedad de aplicaciones, entre las que se incluyen [5] [6]:

- Generación de texto creativo, código, guiones y piezas musicales.
- Traducción de idiomas de forma precisa y fluida.
- Respuesta a pregunta de cualquier tipo.
- Análisis de texto para extraer información, como temas, entidades y relaciones.

Las aplicaciones dentro de InfoHound se analizarán dentro de un capítulo dedicado e igualmente son diversas, de gran relevancia y valor añadido.

2.4.2. Alternativas de LLMs

Existen diferentes opciones de LLMs de uso libre en el mercado entre los que destaca el conocido chat-GPT con sus modelos GPT-3.5 y GPT-4 de la empresa OpenAI por ser pioneros en el segmento. Sin embargo, con el tiempo el abanico de posibilidades se ha diversificado. Actualmente se pueden encontrar modelos de LLM de diversos

proveedores entrenados para multitud de fines, algunos enfocados en texto, otros en imágenes e incluso en audio.

Para este análisis se van a considerar varios criterios por los que seleccionar un modelo u otro y vamos a asignar un peso a cada uno de ellos para posteriormente poder determinar qué modelo cumple mejor con los requisitos del sistema:

- (M) Madurez (peso = 5)
- (V) Velocidad (peso = 10)
- (D) Documentación (peso = 10)
- (F) Facilidad de integración (peso = 10)
- (I) Tiene imagen oficial en Docker (peso = 10)
- (G) Consultas gratuitas (peso = 20)
- (L) Se puede ejecutar en local (peso = 20)

Los LLMs que se han seleccionado para evaluar son los siguientes:

- Locales:
    - PrivateGPT: Es un LLM diseñado para poder ejecutarse en local de forma privada por medio de una API y está orientado al análisis de documentos como contexto de la consulta.
    - Ollama: Es un contenedor de LLM diseñado para ejecutar diversos modelos (Llama2, Mistra, etc) de código abierto en local. Tiene una imagen en Docker oficial.
    - GPT4All: Es un LLM gratuito optimizado para ser instanciado localmente y ejecutar operaciones únicamente a coste de CPU.
- Remotos:
    - GPT-3.5: Es una versión actualizada de GPT-3 de OpenAI. Es de pago y se limita su uso por Tokens.
    - Bard: Es un LLM desarrollado por Google AI. Bard se entrena en un conjunto de datos masivo de texto y código. Es una buena alternativa, pero es de pago a partir de cierta cantidad de caracteres.

A continuación, se asigna una puntuación del 1 al 10 para cada uno de los parámetros definidos en los LLMs a ser evaluados.

	M (5)	V (10)	D (10)	F (10)	I (10)	G (20)	L (20)	**Total**
PrivateGPT	5	3	6	5	1	10	10	575
Ollama	**7**	**3**	**5**	**8**	**10**	**10**	**10**	**695**
GPT4All	5	3	6	5	1	10	10	575
GPT-3.5	9	9	7	8	NA (10)	3	4	525
Bard	8	9	7	8	NA (10)	3	4	520

Tabla 1 - Comparativa de LLMs

A raíz de este análisis preliminar se puede determinar que Ollama es el contenedor de LLMs que probablemente mejor se ajuste a los requisitos. Como puntos determinantes de este contenedor de LLMs tenemos su facilidad para ser desplegado en local por medio de un contenedor Docker oficial, lo cual se ajusta perfectamente a la arquitectura software de InfoHound, además de cumplir con ser de carácter gratuito e ilimitado. Un punto preocupante de Ollama, en el que destacan más los LLMs remotos, es la velocidad, dado que en los locales dependerá de la máquina donde se ejecute y la configuración que se aplique.

Ollama permite ejecutar multitud de modelos pre-entrenados de forma local. Cuenta con una API REST, integración con librerías de terceros y permite una amplia personalización de las consultas pudiendo modificar el contexto añadiendo documentos o condiciones además de permitir modificar los formatos de salida, por ejemplo, en JSON [7].

2.4.3. Aplicaciones de un LLM dentro de InfoHound

Dentro de InfoHound se abre un abanico inmenso de posibilidades una vez integrado un contenedor de modelos LLM local. Una aplicación OSINT maneja gran cantidad de datos en distintos idiomas, contextos y/o formatos. En ocasiones estos datos se encuentran dentro de documentos, en metadatos, buscadores o en redes sociales. Por ello sería de gran complejidad ordenarlos todos y presentárselos a un analista. Un LLM podría ayudar dentro de InfoHound en campos como:

- Reconocimiento de entidades (organizaciones):
    - Identificar automáticamente nombres de personas, organizaciones, ubicaciones u otros elementos relevantes en grandes cantidades de datos, o documentos facilitando la extracción eficiente de información desorganizada.
- Extracción de información:
    - Analizar documentos, artículos o sitios web para extraer información relevante, como eventos, fechas, relaciones y otros datos valiosos, agilizando la tarea de recopilación de datos.
- Análisis de sentimientos y opiniones:
    - Evaluar el tono emocional de textos, comentarios o publicaciones que puedan haber sido recopiladas de las redes de un usuario (o foros) para obtener percepciones sobre actitudes y opiniones asociadas con ciertos temas (por ejemplo, la afinidad política).
- Resúmenes automáticos:
    - Generar resúmenes automáticos de documentos largos o metadatos para proporcionar una visión rápida y concisa de la información esencial, facilitando la revisión y comprensión eficiente de grandes volúmenes de texto.

- Detección de patrones y relaciones:
    - Identificar patrones, conexiones y relaciones dentro de datos dispersos para revelar información oculta, como la vinculación entre entidades o la detección de tendencias en eventos.
- Clasificación de temas:
    - Clasificar documentos o publicaciones en categorías específicas para organizar y estructurar la información, permitiendo una fácil recuperación y análisis por parte de los analistas.
- Traducción automática:
    - Facilitar la comprensión de información en diferentes idiomas mediante la traducción automática, permitiendo a los analistas acceder a datos internacionales sin barreras lingüísticas.
- Detección de desinformación:
    - Identificar posibles señales de desinformación o noticias falsas al analizar patrones lingüísticos y contextualizar la información, mejorando la capacidad de evaluación de la autenticidad de los datos recopilados.
- Asistente OSINT:
    - Proveer de un chat interactivo con el que poder conversar y realizar peticiones sobre los datos extraídos.

Expuestos los anteriores puntos, es evidente la potencia y el valor que puede añadir la integración de un LLM local dentro de InfoHound. Sin duda, hoy en día, implementar todas las funciones anteriormente descritas con una completa contextualización de los datos extraídos por InfoHound se trataría de un desarrollo muy costoso que necesitarían de un conocimiento profundo de implementaciones sobre LLMs, así como posibles grandes modificaciones de la arquitectura de InfoHound y un equipo especializado. Es por ello por lo que este proyecto como demostrador de la integración de la tecnología no pretende implementar todos esos puntos, sino que se va a centrar exclusivamente en el análisis de datos y resúmenes de perfiles.

2.4.4. Caso de uso de LLM aplicado a InfoHound

Dentro de todas las posibilidades que aporta un LLM, el caso de uso elegido a resolver a modo de demostrador de la tecnología para este trabajo es el análisis de datos y resúmenes de perfiles. Se llevará a cabo de la siguiente manera:

1. Se añadirá un campo (*raw_info*) al modelo de personas (“People”) donde se guardarán en texto plano y desorganizado (distintos formatos) los metadatos encontrados por distintos medios sobre una persona.
2. Se creará una función (tarea asíncrona) de adquisición, complementaria a las ya existentes, que permita recuperar información de las personas de una organización basada en una búsqueda en Google con la API JSON.
3. Los perfiles recuperados se mostrarán en la vista de personas sin una descripción de la ocupación de la persona dado que no se contará con ella en este punto.

4. Se creará una función (tarea asíncrona) de análisis, complementaria a las ya existentes, que permitirá generar una descripción para cada uno de los perfiles de las personas encontradas. La generación de esta descripción se hará por medio del resumen de los metadatos desorganizados haciendo uso del contenedor de modelos LLM como un sistema caja negra.

Con todo esto el analista pasará de tener un conjunto desorganizado de datos a un resumen fácilmente interpretable.

2.4.5. Integración de Ollama como LLM en entorno local

Ollama se integra de manera muy sencilla dentro de la plataforma InfoHound. En primer lugar, se añadirá al archivo *docker-compose.yml* un servicio referenciando a la imagen oficial de Ollama con cierta configuración específica.

```yaml
ollama:
  image: ollama/ollama:latest
  ports:
    - '11434:11434'
  environment:
    OLLAMA_COMMAND: "ollama run llama2"
```

Ilustración 9 – Configuración docker-compose para añadir Ollama

Al componer los contenedores este archivo lanzará la creación de un nuevo contenedor a partir de la última imagen de Ollama y expondrá el puerto 11434 para la API REST. Finalmente ejecutará el comando que permitirá cargar en el contenedor el modelo que deseemos usar, en este caso el modelo “llama2”. Hay que destacar que podrían instalarse varios modelos en el contenedor entre los que están disponibles para Ollama. Por ejemplo, se podría instalar un modelo de texto y otro modelo orientado a imágenes. Algunos de los modelos ofrecidos son:

Modelo	Parámetros	Tamaño	Comando
Llama 2	7B	3.8GB	*ollama run llama2*
Mistral	7B	4.1GB	*ollama run mistral*
Dolphin Phi	2.7B	1.6GB	*ollama run dolphin-phi*
Phi-2	2.7B	1.7GB	*ollama run phi*
Neural Chat	7B	4.1GB	*ollama run neural-chat*
Starling	7B	4.1GB	*ollama run starling-lm*
Llama 2 13B	13B	7.3GB	*ollama run llama2:13b*
Llama 2 70B	70B	39GB	*ollama run llama2:70b*
Orca Mini	3B	1.9GB	*ollama run orca-mini*
LLaVA	7B	4.5GB	*ollama run llava*

Tabla 2 – Modelos disponible en Ollama

La columna “parámetros” de la tabla indica la cantidad de datos, en billones, que se han usado para entrenar el modelo. Es importante mencionar que el desarrollador de Ollama indica que se necesitarán al menos 8 GB de RAM para ejecutar los modelos 7M, 16 GB para los 13B y así incrementalmente para los modelos de mayor tamaño. Para este proyecto por la capacidad del hardware local (16GB) no podremos ejecutar modelos mayores de 7B [7].

Así mismo Ollama se puede lanzar en modo aceleración GPU o solo CPU. Para activar la aceleración por GPU tendríamos que configurar el contenedor.

Una vez el contenedor está iniciado y el modelo cargado, Ollama tendrá el servicio activo en localhost:11434 o ollama://ollama:11434 en formato Docker. Se puede comprobar el correcto funcionamiento de la siguiente manera.

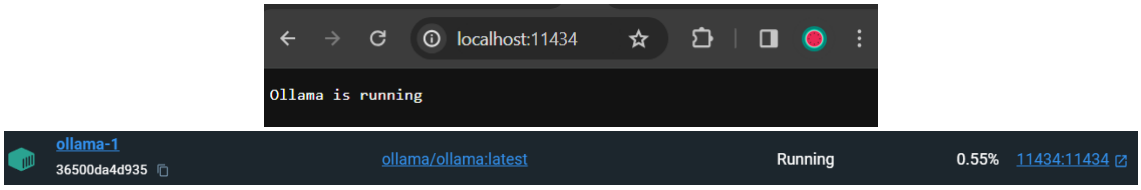

Ilustración 10 – Contenedor Ollama LLM activo

La forma de interactuar con Ollama desde la aplicación web Django será por medio de la API REST, sin embargo, para simplificar la interacción con esta API podemos usar la librería de alto nivel de interacción con Ollama de Langchain. Es librería es fácilmente integrable con Django y permite una sencilla configuración y uso.

```python
from langchain.llms import Ollama
from infohound_project import settings

def ollama_flexible_prompt(in_prompt):
    BASE_URL = settings.OLLAMA_URL
    MODEL = "llama2"
    ollama = Ollama(base_url=BASE_URL, model=MODEL)
    try:
        res = ollama(in_prompt)
        return res
    except Exception as e:
        print(f"Error en la llamada a Ollama: {e}")
        return None
```

Ilustración 11 – Funcion de interacción con contenedor Ollama

Langchain permite de manera muy sencilla definir una interfaz de consultas a Ollama desde Python tal y como se puede ver en la ilustración superior donde se muestra el

código desarrollado en el archivo *ollama.py* dentro de módulo *ai_assistant* de Django. Esta función se llamará desde las tareas que requieran hacer uso del contenedor de modelos LLM generando una *prompt*.

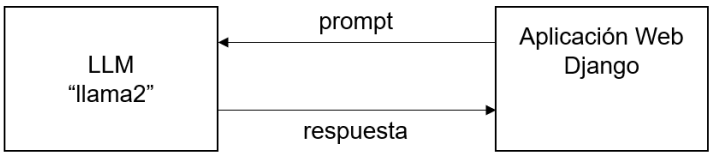

Ilustración 12 – Integración de Ollama con Django por medio de Langchain

De esta manera ya tenemos el contenedor local en Docker con el LLM (modelo “llama2”) funcionando y una función en Django para interactuar con este.

2.5. Búsqueda de información de fuentes abiertas (Google API JSON)

Una vez que se tiene el contenedor de modelos LLM integrado dentro de la arquitectura necesitamos una fuente de datos que analizar. Se podrían haber usado las ya existentes, pero con la idea de contribuir al crecimiento de la plataforma se ha desarrollado una función nueva.

La idea de esta función de búsqueda pasa por recopilar el mayor número de perfiles de personas trabajando en una organización. Para esto se va a usar la API de Google con el objetivo de encontrar perfiles de la red social LinkedIn.

2.5.1. Google Custom Search JSON

Google Custom Search JSON es una API proporcionada por Google que permite integrar la funcionalidad de búsqueda personalizada dentro de aplicaciones o sitios web. A través de esta API, se pueden realizar consultas de búsqueda específicas y recibir resultados en formato JSON.

De esta manera Google Custom Search JSON API permite a InfoHound:

- Realizar búsquedas en Google:
    - Permite a los desarrolladores definir y personalizar los motores de búsqueda para adaptarse a sus necesidades específicas, centrándose en dominios o sitios web particulares.
- Recibir los resultados en formato JSON:
    - Los resultados de las consultas de búsqueda se devuelven en formato JSON, lo que facilita el análisis y procesamiento de datos en aplicaciones web.
- Controlar y filtrar los resultados:
    - Proporciona opciones para filtrar y ordenar los resultados de búsqueda, así como limitar el conjunto de resultados según criterios específicos.

Es importante destacar que Google Custom Search JSON API es parte de la oferta de Google Cloud Platform y aunque tenga una cuota de solicitudes gratuitas, una vez agotadas estas, se requiere pagar por seguir usándola. Para controlar esto, Google requieres configurar un proyecto en la consola de desarrolladores de Google y obtener una clave de API para acceder a la API REST y así gestionar las cuotas de uso. En el caso de InfoHound al estar usando solo la parte gratuita, se tendrá un número limitado de búsquedas cada cierto tiempo y una vez terminadas se avisará de esto al usuario con el mensaje de error “RESOURCE_EXHAUSTED”. Es decisión de cada analista pagar o no este servicio.

2.5.2. Integración de Google Custom Search JSON

Existen diversas librerías para la integración de una aplicación en Python con esta API de Google, pero en este caso se ha optado por realizar directamente las consultas a la url objetivo de la API.

```
payload = {"key":API_KEY,"cx":ID,"start":start,"q":query}
res = requests.get(GOOGLE_API_URL,params=payload)
data = json.loads(res.text)
```

Ilustración 13 – Consulta a Google API

Es importante mencionar que la API te devolverá las respuestas de 10 en 10 por lo que es necesario implementar una función que lea la primera respuesta para saber el nuemero de respeustas totales y así iterar las peticiones para recoger más resultados.

La función de adquisición desarrolalda con nombre *discoverpeople(query)* dentro de *google_data.py* disponible en el Anexo I – Archivos Python se utiliza para buscar personas en Google Search y luego devolver una lista de personas con sus nombres, enlaces, información y URL de imagen. La función funciona de la siguiente manera:

1. Comienza creando un objeto payload con los parámetros de la solicitud de búsqueda, como la clave API, la ID de búsqueda personalizada, el número de inicio y la consulta de búsqueda. La query que se ha aplicado para este demostrador es “query = f'intitle:"{company}" site:"linkedin.com"”.
2. Realiza una solicitud GET a la API de Custom Search de Google y recupera los resultados de la búsqueda en formato JSON.
3. Itera sobre los resultados de la búsqueda y extrae la información de cada persona, incluyendo su nombre, enlace, información y URL de imagen. Para distinguir falsos positivos, si la estructura del objteo JSON devuelto contiene ni el parámetro “profile:first_name” ni “profile:last_name” se considera que no pertenece a un perfil de usuario y se lanza una excepción descartando ese resultado. La función detiene la búsqueda si se encuentra con un error o si se ha alcanzado el límite de 100 resultados.
4. Agrega la información de cada persona a una lista de personas.
5. La función devuelve la lista de personas al final de la ejecución para que otra funciónd e nivel superior guarde lso datos en lso modelos.

Es importante destacar que dentro de la información que devuelve la función se guarda el objeto JSON al completo dentro de un campo en el modelo. Esto servirá para que mas adelante se genere de ahí la descripción.

2.5.3. Nueva tarea de adquisición de personas

Una vez se dispone de la funcionalidad para realizar una búsqueda de personas por medio de la API JSON de Google es necesario proporcionar al analista de un medio por el cual ejecutar esta nueva tarea de adquisición.

InfoHound está pensado para que de forma sencilla se puedan añadir estas tareas por medio de Clery. Sería tan sencillo como una vez que tengamos desarrollada la función que queremos ejecutar creemos una llamada a esta en el archivo *tasks.py* usando la nomenclatura de Celery tal y como se muestra a continuación.

```python
@shared_task(bind=True, name="find_people_from_google")
def findPeopleFromGoogleTask(self, domain):
    people.findPeopleFromGoogle(domain)
```

Ilustración 14 – Tarea “Find People From Google” en Celery

Además de esto también habría que añadir esta tarea a una lista del archivo *utils.py* donde se generan e inicializan las tareas de dicha lista al arrancar la aplicación web.

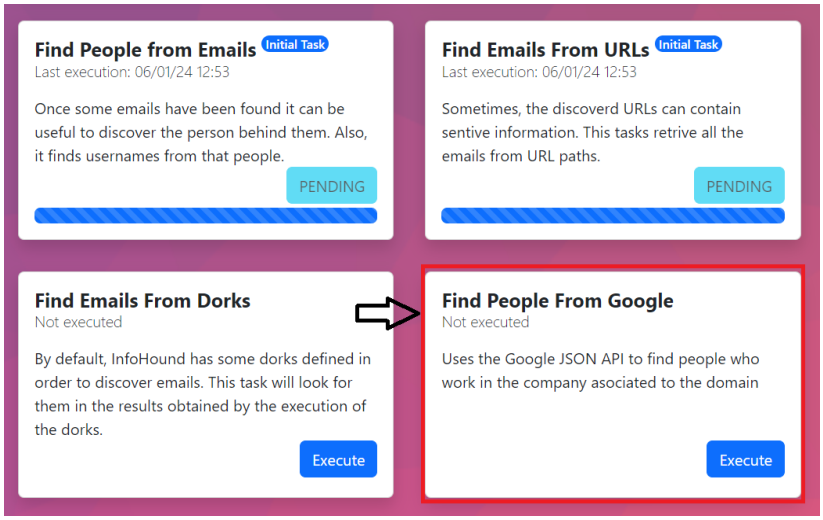

Ilustración 15 – Tarea “Find People From Google” añadida a InfoHound

Una vez añadida la tarea a la lista ya aparecería en el menú y pulsando sobre el botón de ejecución se podría lanzar la función asociada de manera asíncrona.

2.5.4. Ejecución de la tarea “Find People From Google”

Al pulsar sobre el botón de ejecución de la tarea se crea una tarea software en segundo plano gestionada por Celery que ejecuta la función de búsqueda. Se puede ver entonces en la consola de Celery que se empiezan a recopilar resultados.

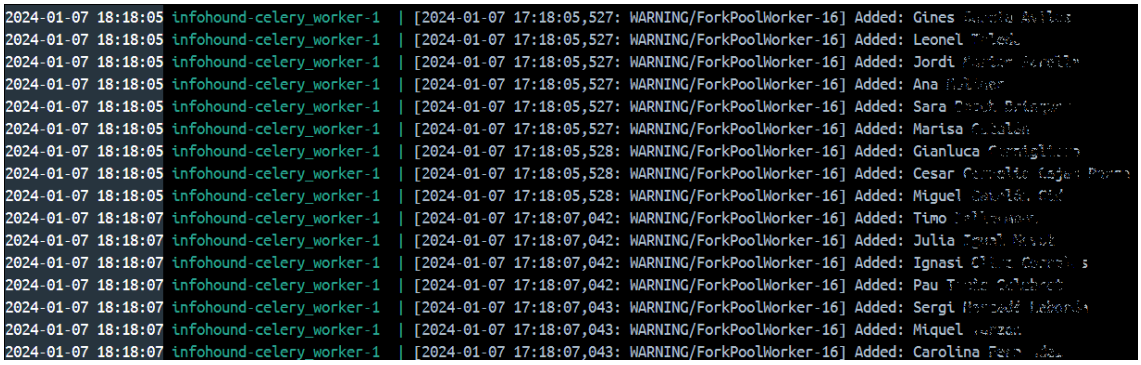

Ilustración 16 – Pocesamiento Celery de tarea “Find People From Google”

De igual manera si se accede a la vista de Personas apareceran esos objetos representando a los miembros de una organización encontrados. En este punto aun no contarán con una descripción.

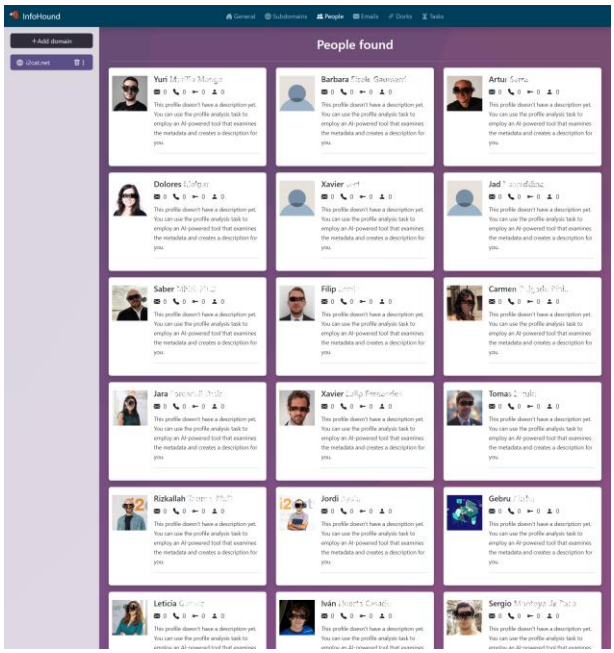

Ilustración 17 – Resultados tarea “Find People From Google”

Para la fundación i2cat se han recuperado unos 87 perfiles, lo cual se considera un éxito teniendo en cuenta que se está usando la versión gratuita de la API que limita a 100 respuestas por búsqueda. Usando la versión de pago y optimizando la consulta se podrían sacar prácticamente todos los miembros de la organización activos en LinkedIn.

2.6. Síntesis de datos de fuentes abiertas por medio del LLM

Como última parte del proceso, una vez adquiridos los datos de los perfiles de manera desorganizada, faltaría poder analizarlos para generar una descripción de cada perfil que sirva a los analistas. Para esto no solo hay que buscar en la información, sino que también hay que resumirla para que no sea demasiado extensa. Aquí es donde el LLM nos va a permitir sin necesidad de programar una función compleja, demostrar que por medio de una *prompt* este va a ser capaz de llevar a cabo esas tareas:

1. Buscar la información que necesitamos dentro de los datos
2. Resumir esa información en un párrafo

Si esto se cumple, se podría validar así la versatilidad que proporciona un LLM dentro de InfoHound.

2.6.1. Nueva tarea de análisis de perfiles

Para que un analista pueda lanzar la tarea de análisis de los perfiles ya recolectados y generación de resúmenes, se debe crear y añadir una tarea haciendo uso de Celery para que esta pueda ser ejecutada en segundo plano de manera asíncrona.

```python
@shared_task(bind=True, name="summarize_profile")
def summarize_profile(self, domain):
    people_analisys.summarize_profile(domain)
```

Ilustración 18 – Tarea “AI-Powered Profile Analisys” en Celery

De esta manera y añadiendola a la lista de tareas de la aplicación, esta dse precargará en el menú de tareas al arrancar la aplicación.

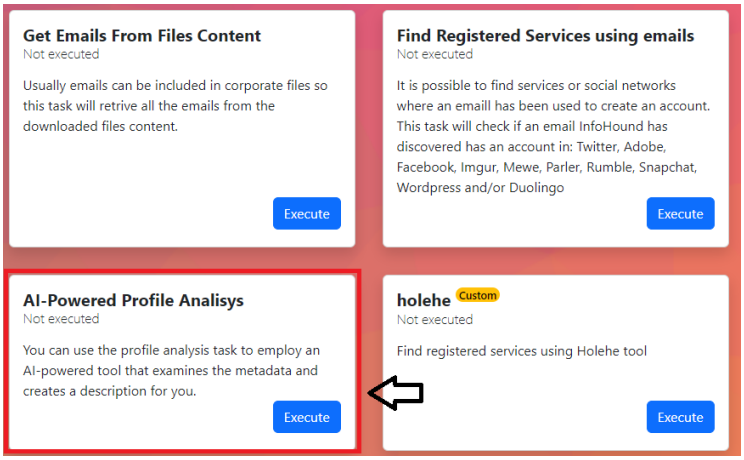

Ilustración 19 – Tarea “AI-Powered Profile Analisys” añadida a InfoHound

Una vez añadida la tarea a la lista ya aparecería en el menú y pulsando sobre el botón de ejecución se podría lanzar la función asociada de manera asíncrona.

2.6.2. Ejecución de la tarea “AI-Powered Profile Analisys”

Al pulsar sobre el botón de ejecución de la tarea se crea una tarea software en segundo plano gestionada por Celery que ejecuta la función de análisis. Esta función toma como parámetro un identificador de dominio. Utiliza este identificador para filtrar las personas en una base de datos que pertenecen a ese dominio. Luego, itera sobre los resultados de esa consulta y realiza un análisis de resumen de ocupación para cada persona lanzando una prompt a Ollama. Actualiza el campo ocupation_summary en la entrada de la base de datos con el resultado del resumen y guarda la entrada actualizada en la base de datos.

```
def summarize_profile(domain_id):
    queryset = People.objects.filter(domain_id=domain_id)

    for entry in queryset.iterator():
        try:
            summarize_prompt = "Summarize the occupation of the person in
just one pharagraph given the following data: "
            raw_data = entry.raw_metadata
            entry.ocupation_summary =
ollama.ollama_flexible_prompt(summarize_prompt + raw_data)
            entry.save()
        except Exception as e:
            print(f"Error inesperado: {str(e)}")
```

Ilustración 20 – Función de analisis mediante “prompts” al contenedor Ollama

Analizando los logs del contenedor de Ollama se puede observar que las *prompts* están llegando correctamente y se están resolviendo.

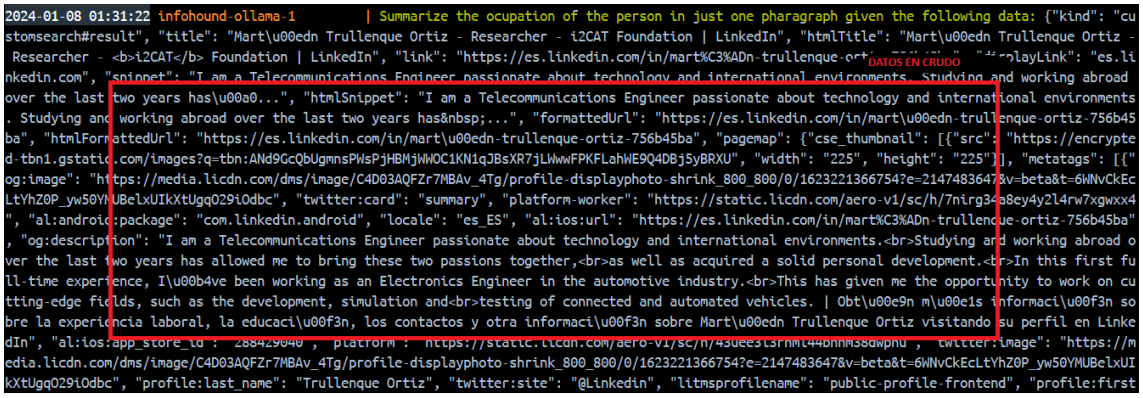

Ilustración 21 – Recepción de prompt en contenedor Docker

Los datos que se muestran en la anterior ilustración en crudo son los datos extraidos directamente de la consulta a la API JSON y de donde el LLM buscará y generara el resumen de perfil.

También se puede observar que la tarea en segundo plano poco a poco va generando los resumenes en la vista de personas y de esta manera se van rellenando los perfiles, tal y como se puede observar en los dos perfiles de la siguiente imagen que aun no han sido analizados aun mientras que el tercero ya cuenta con una descripción.

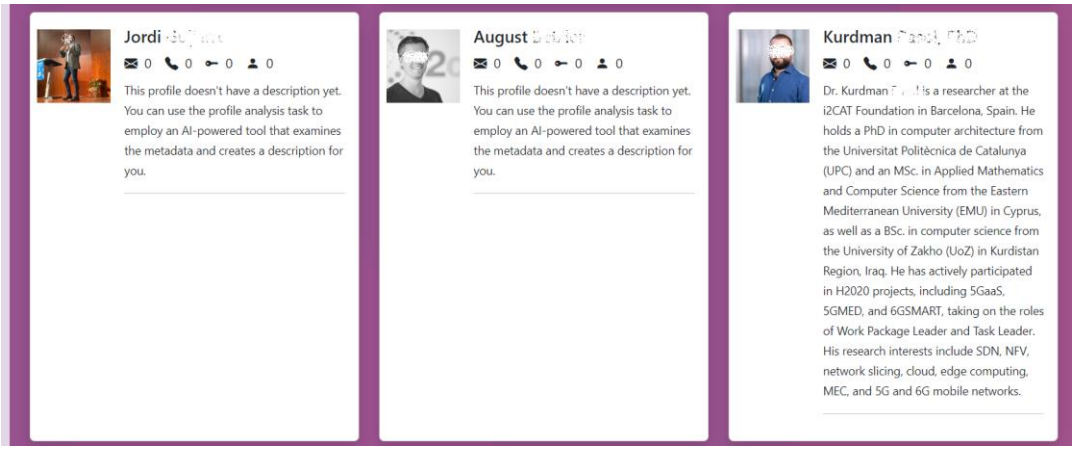

Ilustración 22 – Resultados a mitad de procesamiento de la tarea “AI-Powered Profile Analisys”

Y finalmente otro ejemplo de cómo quedaría la vista de personas con los perfiles ya analizados por el contenedor de modelos de lenguaje de gran tamaño.

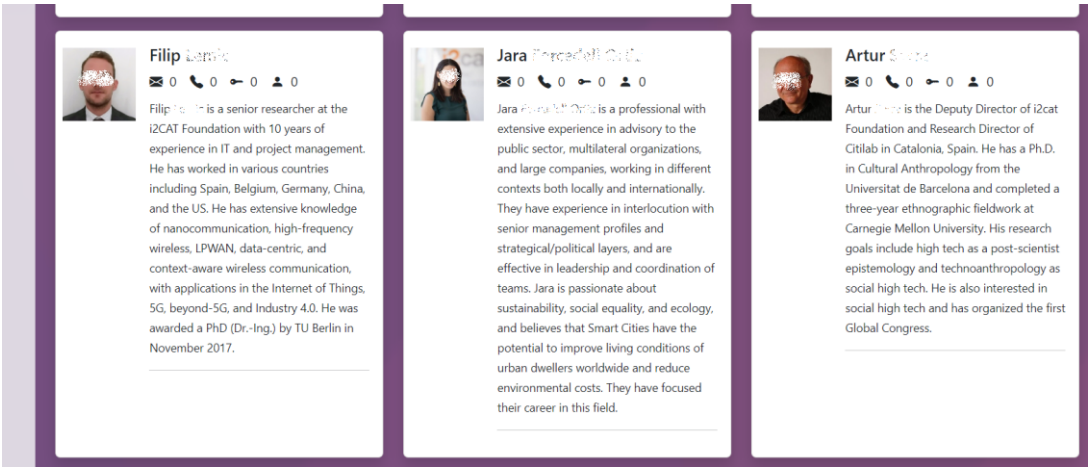

Ilustración 23 – Resultados de la tarea “AI-Powered Profile Analisys”

El análisis se ejecuta persona por persona y de manera asíncrona en segundo plano, por lo tanto el analista podrá seguir usando la plataforma mientras se van creando las descripciones de los perfiles.

2.6.3. Tiempos y observaciones

Uno de los principales contratiempos observados tiene que ver con el tiempo de procesamiento de cada consulta al LLM.

En la máquina donde se ha ejecutado el demostrador con las siguientes características, la resolución de las instrucciones ha tardado de media unos 3 o 4 minutos por perfil.

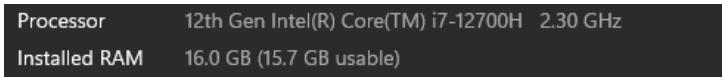

Ilustración 24 – Caracteristicas técnicas del host de pruebas

Estos tiempos podrían mejorarse ejecutando el contenedor en una máquina mas potente o activando el procesamiento por GPU de Ollama. Adicionalmente, se podrían reducir los tiempos con cierta optimización de las prompts y limpieza de los datos pasados en crudo a estas.

Sin embargo, al tratarse de una tarea en segundo plano estos tiempos podrían ser considerados aceptables en un entorno de demostración.

3. Resultados

3.1. Resultado del caso de uso práctico

Se ha analizado y verificado exitosamente un caso de uso en el que un LLM puede ser de utilidad para InfoHound. Los resultados han sido los esperados al haber generado exitosamente resúmenes de perfiles profesionales de los trabajadores de una organización analizada desde un conjunto de datos desorganizados por medio de la inteligencia artificial.

A nivel funcional, el proceso llevado a cabo para la demostración corresponde al definido en el ciclo de inteligencia, empezando por una planificación, recolección de la información, análisis haciendo uso del contenedor LLM y finalmente la presentación o difusión de los datos.

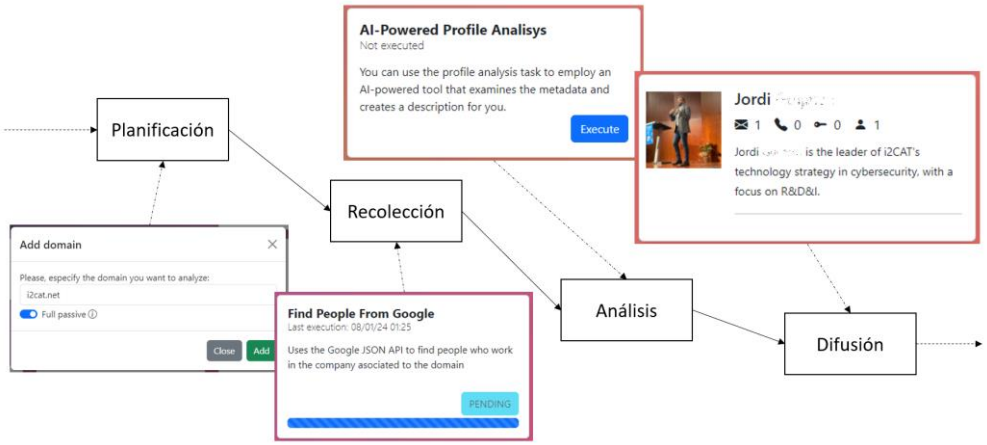

Ilustración 25 – Pasos de la nueva funcionalidad integrados con el ciclo OSINT

A nivel de datos y procesamiento, el avance remarcable donde la integración del LLM ha sido de gran ayuda es en la busqueda y resumen automáticos de esos datos para la generación del perfil.

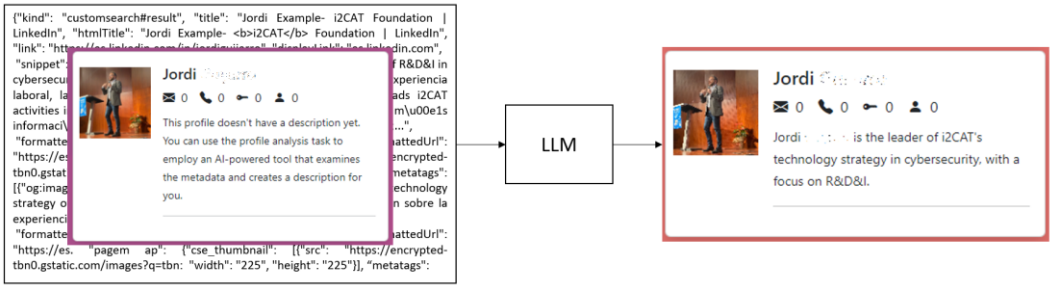

Ilustración 26 – Generación por sistema LLM caja negra

Así mismo, a vista de los resultados ejecutados sobre “i2cat.net” se puede dar por demostrada la búsqueda y síntesis de datos por medio de un LLM.

3.2. Producto

El producto generado es una actualización de la ya existente herramienta InfoHound añadiendo las siguientes principales funcionalidades:

1. Integración de un contenedor de modelos de lenguaje de gran tamaño (LLM) y los respectivos medios para interaccionar con este.
2. Desarrollo e integración de una función de búsqueda de perfiles de personas de la organización auditada en LinkedIn por medio de Google JSON API. Lanzamiento de esta funcionalidad por medio de la tarea “Find People from Google”
3. Desarrollo e integración de una función de análisis de la información en crudo de los modelos de personas para sintetizarla y generar una descripción haciendo uso de la inteligencia artificial. Lanzamiento de esta funcionalidad por medio de la tarea “AI-Powered Profile Analisys”

Para conseguir la completa integración de estas nuevas funcionalidades además se han tenido que modificar total o parciamente los archivos incluidos en el Anexo I – Archivos Python, Anexo II – Archivos JS Frontend y Anexo III – Archivos de Configuración.

4. Conclusiones y trabajos futuros

4.1. Conclusiones generales

De manera general, la principal idea que este trabajo pretende transmitir es la versatilidad, flexibilidad y el valor añadido que puede proporcionar la inteligencia artificial por medio de los lenguajes de gran tamaño (LLMs) a las plataformas de recolección de información de fuentes abiertas (OSINT). Para demostrar esto se ha llevado a cabo un análisis del estado del arte y se ha realizado un caso práctico con el que dar validez a este tipo de tecnologías dentro de InfoHound, como caso de ejemplo. Las conclusiones a las que se ha llegado son las siguientes:

1. Los LLMs son relativamente sencillos de integrar dentro de plataformas al poder modelarse como un sistema caja negra.
2. Los LLMs son una alternativa interesante para analizar grandes cantidades de datos de manera sencilla.
3. Los LLMs tienen multitud de aplicaciones, más allá del análisis de datos, de gran valor para las herramientas OSINT.
4. La velocidad de resolución de instrucciones de un LLM es muy dependiente del hardware de la máquina anfitriona. Ejecutar un LLM en local puede penalizar la velocidad.
5. Existen gran variedad de modelos LLM actualmente en auge por lo que será una tecnología muy cambiante en los próximos años y aparecerán nuevos actores. Lo que pueda parecer la mejor solución hoy, mañana podría ser resuelta de manera más eficiente gracias a la aparición de un nuevo actor.

Tras esto se puede concluir que se han obtenido los resultados esperados y que los LLMs jugaran un papel vital en hacer crecer las capacidades de las plataformas web tales como InfoHound al proveer de una forma eficiente y versátil de buscar, organizar, analizar, sintetizar, traducir o crear datos.

4.2. Planificación, materialización de riesgos y problemas encontrados

El trabajo no ha estado exento de contratiempos, aun así, las dificultades que han ido apareciendo se han ido sorteando con éxito.

El proyecto en sus orígenes, cuando se planteó, tenía un enunciado y unos requisitos distintos a los actuales. Pretendía de igual manera contribuir a la plataforma InfoHound, pero, recopilando y analizando datos de Twitter. Durante el avance del proyecto debido a un riesgo (políticas de librerías de terceros) de carácter externo materializado, cambio de la API de Twitter a servicio de pago, hubo que replantear como se iba a hacer la adquisición de datos. De esta manera juntando a las partes interesadas se elaboró un plan de contingencia y se redefinieron los objetivos. El impacto fue tener que reelaborar gran parte del análisis y por consiguiente reducir a más de la mitad el tiempo. Por ello no solo es importante la monitorización constante

de los riesgos de proyecto y contar con un plan de contingencia, sino que, además, habría sido interesante haber realizado un análisis de viabilidad al inicio del proyecto.

4.3. Puntos de mejora y trabajo a futuro

Durante el desarrollo del proyecto se han ido anotando algunos puntos a mejorar que podrían ser implementados en el futuro.

Para la función de adquisición:

- En la función de adquisición mejorar problemas con algunas empresas que tienen nombres comunes dando pie a falsos positivos.
- En la función de adquisición agregar una función para poder realizar búsquedas de empleados de aquellas empresas que tengan un nombre compuesto. Hoy en día es una limitación.

Para la función de análisis:

- Hacer de ingeniería de prompts para mejorar las respuestas dado que en ciertos casos se pasa de longitud.
- Crear otra función en la librería interna de Ollama que permita hacer peticiones con formatos de salida específicos, por ejemplo, JSON. Esto podría dar pie a pedir al LLM que te analice un documento y te devuelva en JSON una lista de los nombres de empresas o personas que haya podido encontrar.

Para el propio LLM:

- Configurar correctamente el contenedor de Ollama para activar la aceleración por GPU. Actualmente no tiene soporte en Windows.
- Sería de gran interés introducir la información descargada en la plataforma como contexto del LLM para así poder dar mejores salidas a las instrucciones. Esto es costoso, pero daría pie a poder analizar los documentos descargados o mediante un chatbot interactuar con la plataforma para que el analista indique lo que necesita.
- Sería de gran interés usar el LLM para clasificar a las personas por categorías, para por ejemplo poder recuperar todos los profesionales que trabajen en ciberseguridad.

Bibliografía

A continuación, se puede encontrar la lista de referencias consultadas para el desarrollo del proyecto:

[1] InfoHound tool – Improving OSINT Open Source CyberArsenal for Good
Xavier Marrugat Plaza – Jordi Guijarro Olivares
13 de junio de 2023
Universitat Oberta de Catalunya

[2] OSINT – La Información es Poder
Asier Martínez
28 de mayo de 2014
INCIBE (Instituto Nacional de Ciberseguridad)

[3] Docker Official Site
Documentation, Manuals and Examples
Recurso web

[4] Django Official Site
Documentation, Manuals and Examples
Recurso web

[5] Inteligencia artificial generativa – Introducción a los LLMs
Gobierto
Recurso web

[6] Bootstrap knowledge of LLMs ASAP
rain-1
GitHub Gist
Recuso web

[7] Ollama Official GitHub
Documentation, Manuals and Examples
Recurso web

[8] GPT4All Official Site
Documentation, Manuals and Examples
Recurso web

[9] PrivateGPT Officia Site
Documentation, Manuals and Examples
Recurso web

Anexo I – Archivos Python

Nombre: infohound/tool/data_sources/google_data.py
Descripción: Este archivo contiene la función que permite lanzar la búsqueda de personas en LinkedIn integrada con Google API JSON.
Tamaño: 6.17KB
SHA1: e350a709fbb51dcea8749f742372c1c6b47f237a
Archivo adjunto:

google_data.py

Nombre: infohound/tool/retrieve_modules /people.py
Descripción: Este archivo contiene el código que permite llamar a las diferentes funciones existentes para la adquisición de perfiles. Se usa como proxy entre las funciones de bajo nivel y las tareas.
Tamaño: 3.22 KB
SHA1: 968651dd897c71dad381aebcb4df1a1e32b6de27
Archivo adjunto:

people.py

Nombre: infohound/tool/analisys_modules /people_analisys.py
Descripción: Este archivo contiene las funciones de bajo nivel que permiten analizar los perfiles recopilados.
Tamaño: 759 bytes
SHA1: 79006d7e05c2f0e0bdf5f491027a23f56deff83a
Archivo adjunto:

people_analisys.py

Nombre: infohound/tool/ai_assistant/ollama.py
Descripción: Este archivo contiene la integración con el contenedor de Ollama por medio de la librería de Langchain.
Tamaño: 401 bytes
SHA1: 99ae0e924803847fd81ee50c06d9993b395809e0
Archivo adjunto:

ollama.py

Nombre: infohound/models.py
Descripción: Este archivo contiene los modelos de la aplicación Django. No se han añadido nuevos, pero sí que se han modificado los campos de algunos de los existentes.
Tamaño: 4.65 KB
SHA1: 5bc4ddcfe56ef051f8b564c4c29c6706fb461a87
Archivo adjunto:

models.py

Nombre: infohound/tasks.py
Descripción: Este archivo permite definir las tareas que posteriormente Celery ejecutará de manera asíncrona. Para ello hay que definir un nombre y una función objetivo a la que llamar.
Tamaño: 3.66 KB
SHA1: 51f9c36e6680778cb2e178b6c641d44a4059b24b
Archivo adjunto:

tasks.py

Nombre: infohound/utils.py
Descripción: Este archivo inicializa las tareas permitiendo tanto guardarlas en la base de datos como que posteriormente la vista genere un elemento interactivo por cada una de ellas.
Tamaño: 8.29 KB
SHA1: f5adbfdb7fd1ad27a72a4f219e9f68a15176cc79
Archivo adjunto:

utils.py

Nombre: infohound_project/settings.py
Descripción: Este archivo contiene variables de configuración del sistema.
Tamaño: 3.67 KB
SHA1: 9446f7fef686ded193906c3f616acd0372e44192
Archivo adjunto:

settings.py

Anexo II – Archivos JS Frontend

Nombre: infohound/static/infohound/js/index.js
Descripción: Este archivo se usa para el renderizado de las vistas desde el Frontend. La parte cliente recibirá la información necesaria para componer el frontend en formato JSON y este se construirá interpretándolo por medio de este archivo.
Tamaño: 28.9 KB
SHA1: 81da12792096309a1221fb0e681bda16d5174a8f
Archivo adjunto:


index.js

Anexo III – Archivos de Configuración

Nombre: /docker-compose.yml
Descripción: Este archivo contiene permite definir y ejecutar aplicaciones Docker multi-contenedor de manera fácil, especificando servicios, redes y volúmenes para la configuración y despliegue simplificado.
Tamaño: 1.50 KB
SHA1: 3b0e51f388c0034dfdd1b52cd30e77dc25c2284f
Archivo adjunto:


docker-compose.yml

Nombre: /requirements.txt
Descripción: Este archivo lista las dependencias y versiones de las bibliotecas Python necesarias para ejecutar una aplicación Django específica, facilitando la instalación automática de dichas dependencias
Tamaño: 401 bytes
SHA1: 84dabf764a49590b5f77ef57ec18bbd5cfc1bbdf
Archivo adjunto:


requirements.txt

Anexo IV – Diagrama de Gantt del proyecto

Nombre: Gantt Diagram
Descripción: Este archivo contiene una representación gráfica de la evolución del proyecto y sus actividades como complemento al cronograma de la Ilustración 1.
Tamaño: 22KB
SHA1: 76768150a0d23a31896749dd0eaa2ec2481edd9a
Archivo adjunto:

Gantt Diagram.xlsx