



Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE GRADO

Asistente virtual “Telegram” para la interpretación de documentos.

Grado en Ingeniería Mecánica

ALUMNO: Gonzalo Fernández Conde

DIRECTORES: Milagros Fernández Gavilanes
Norberto Fernández García

CURSO ACADÉMICO: 2023-2024

Universida_{de}Vigo



Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE GRADO

Asistente virtual “Telegram” para la interpretación de documentos.

Grado en Ingeniería Mecánica
Intensificación en Tecnología Naval
Cuerpo General

Universida_{de}Vigo

RESUMEN

El campo del aprendizaje profundo (deep learning) y los modelos de procesamiento del lenguaje natural han transformado radicalmente el panorama de la inteligencia artificial (IA). Los grandes modelos del lenguaje (LLM's) son un claro ejemplo de esta innovación, capaces de generar texto con una calidad cada vez más cercana a la humana, como demuestran herramientas como ChatGPT. El objetivo del presente trabajo es el desarrollo de un asistente virtual (basado en un chatbot) que aproveche este tipo de técnicas de IA, utilizando para ello modelos de lenguaje y herramientas de código abierto. Este asistente procesará documentos en un cierto dominio, permitiendo al usuario realizar consultas sobre ellos. Como interfaz de acceso a dicho asistente virtual, se utiliza un bot de Telegram, la conocida aplicación de mensajería online. La aplicación desarrollada ha sido validada en un escenario de prueba consistente en el análisis de contenidos publicados en redes sociales (como Youtube, Instagram, etc.) para determinar si dichos contenidos resultan de interés para la temática de Defensa, pensando en su posible uso en aplicaciones de inteligencia de fuentes abiertas (OSINT).

PALABRAS CLAVE

Inteligencia artificial, Asistente virtual, Telegram, LLM, OSINT.

AGRADECIMIENTOS

Ante todo, quiero expresar mi gratitud a mis tutores, D. Norberto Fernández García y Dña. Milagros Fernández Gavilanes, por su dedicación y compromiso en el desarrollo de este trabajo, así como por convertir este proyecto en algo tan interesante y ameno a pesar de la dificultad que puede suponer este campo de investigación. ¡Muchas gracias, profesores!

Por otro lado, me gustaría agradecer a mis compañeros del estudio 204 por un año de duro trabajo en equipo, el cual podría haber sido aún más difícil de no ser por las risas, su apoyo y, en general, todos esos momentos que hacen el día a día menos rutinario. También quiero agradecer a Aarón Montesa Foncillas por su infinita paciencia al resolver mis dudas durante estos 5 años de carrera.

Asimismo, quiero expresar mi más sincero agradecimiento a toda mi familia por su apoyo incondicional durante todos estos años, sin el cual no habría podido alcanzar las metas que desde tan pronto anhelaba. En especial, quiero agradecer a mi madre por ser el ejemplo de fuerza y resiliencia que me ha ayudado a levantarme tras cada caída un poco más fuerte.

Finalmente, quiero hacer una mención especial a mi abuelo, D. Carlos Conde Sánchez, por inculcarme desde pequeño los valores de la vida castrense, los cuales me han impulsado a seguir sus pasos en la carrera de las armas.

¡A todos, muchas gracias!

CONTENIDO

Contenido	1
Índice de Figuras	3
Índice de Tablas.....	5
1 Introducción y objetivos	6
1.1 Contexto y motivación.	6
1.1.1 Antecedentes históricos.	6
1.1.2 Orígenes del OSINT.	7
1.1.3 La importancia de las redes sociales.....	7
1.1.4 La importancia de la inteligencia artificial.	8
1.2 Objetivos y metodología.	8
1.3 Estructura de la memoria.	9
2 Estado del arte	10
2.1 Inteligencia y espionaje.....	10
2.1.1 Modelo de obtención de inteligencia	10
2.1.2 OSINT y sus aplicaciones.....	12
2.1.3 Herramientas de OSINT.	13
2.2 Inteligencia artificial.	14
2.2.1 Introducción e historia.	14
2.2.2 Aprendizaje automático.	15
2.2.3 Deep learning y redes neuronales artificiales.	17
2.2.4 Transformers y LLM.	20
2.2.5 Llama 2.	21
2.3 Social Media.	22
2.3.1 Introducción e historia.	22
2.3.2 X/Twitter.....	23
2.3.3 Instagram.	25
2.3.4 YouTube.	25
2.3.5 Telegram.	27
2.3.6 API de Telegram.....	27
2.4 Trabajos relacionados.	29
2.4.1 Sistema de detección inteligente de usuarios de Twitter interesados en la temática de defensa.....	29
2.4.2 Herramientas de OSINT para la obtención de información en Twitter sobre el conflicto Ucrania-Rusia.....	29

2.4.3 Plataforma OSINT para el almacenamiento y análisis inteligente de la red social Twitter.	29
2.4.4 Evaluación de los chatbot LLM para la concienciación de Amenazas Cibernéticas basada en OSINT.	29
2.4.5 ChatGPT como oficial de inteligencia.	30
3 Desarrollo del TFG	31
3.1 Entorno de programación.	32
3.2 Interfaz de usuario.	34
3.2.1 Creación del bot.	34
3.2.2 Conexión con la API de Telegram.	37
3.3 Captura de datos.	39
3.3.1 YouTube scraper.	39
3.3.2 Instagram scraper.	44
3.3.3 Analizador de archivos.	47
3.4 Análisis de datos.	48
3.4.1 Construcción del dataset.	48
3.4.2 Parámetros de entrenamiento.	49
3.4.3 Optimización del entrenamiento.	51
3.4.4 Ajuste fino.	54
3.4.5 Inferencia.	55
4 Prueba y validación	58
4.1 Prueba del flujo de código.	59
4.2 Prueba del capturador de datos.	61
4.3 Prueba del analizador.	63
4.3.1 Modelo sin ajustar.	63
4.3.2 Modelo ajustado.	66
4.3.3 Evaluación general.	68
4.3.4 Conclusión de las pruebas y resultados	70
5 Conclusiones y líneas futuras	74
5.1 Conclusiones.	74
5.1.1 Conclusiones generales.	74
5.2 Líneas futuras.	75
6 Bibliografía.	77
Anexo I: Implicaciones Sociales, Económicas, Ambientales	84
Anexo II: Reflexiones Éticas y Sociales	86
Anexo III: Código	87

ÍNDICE DE FIGURAS

Figura 1-1 Ejemplo de uso ilícito de X/Twitter [7].	7
Figura 2-1 Ciclo de la inteligencia (Autoría propia).	11
Figura 2-2 Fuentes de obtención de inteligencia (Autoría propia).	12
Figura 2-3 <i>Machine Learning</i> vs <i>Deep Learning</i> .	17
Figura 2-4 Ejemplo de entrenamiento de una red neuronal [38].	18
Figura 2-5 Ejemplo de RNN [39].	18
Figura 2-6 Ejemplo de CNN [39].	19
Figura 2-7 Ejemplo de <i>embedding</i> semántico [45].	20
Figura 2-8 Dataset de Llama 2 [57].	22
Figura 2-9 Distribución de plataformas de <i>social media</i> en función del número de usuarios [30].	23
Figura 2-10 Captura de usuario de X (Autoría propia).	24
Figura 2-11 Captura de usuario de Instagram (Autoría propia).	25
Figura 2-12 Captura de usuario de YouTube (Autoría propia).	26
Figura 2-13 Protocolo MTPProto [78].	28
Figura 3-1 Principales módulos de la herramienta (Autoría propia).	31
Figura 3-2 Arquitectura de la herramienta (Autoría propia).	32
Figura 3-3 Captura de Colab. Abrir nuevo cuaderno (Autoría propia).	33
Figura 3-4 Captura de Colab. Ejemplo de ejecución de bloque de código y texto (Autoría propia).	33
Figura 3-5 Captura de Colab. Configuración del entorno (Autoría propia).	34
Figura 3-6 Configuración de Telegram (Autoría propia).	35
Figura 3-7 Búsqueda de “BotFather” (Autoría propia).	35
Figura 3-8 Inicio de “BotFather” (Autoría propia).	36
Figura 3-9 Obtención del token de “BotFather” (Autoría propia).	36
Figura 3-10 Comando de instalación de <i>pytelegrambotapi</i> (Autoría propia).	37
Figura 3-11 Código de Python para conectarse al bot mediante un token (Autoría propia).	37
Figura 3-12 Código de Python para implementar un teclado de botones en Telegram (Autoría propia).	38
Figura 3-13 Captura de Telegram. Interfaz de botones (Autoría propia).	38
Figura 3-14 Código de Python para implementar un manejador de comandos (Autoría propia).	38
Figura 3-15 Código de Python. Método <i>bot.register_next_step_handler</i> (Autoría propia).	39
Figura 3-16 Código de Python para respuesta y envío de mensajes (Autoría propia).	39
Figura 3-17 Ejemplo de cuota de YouTube Data v3 para procesos en vídeos [92].	40
Figura 3-18 Fragmento de código de Python para solicitar un canal al usuario (Autoría propia).	40

Figura 3-19 Fragmento de código de Python para obtener el identificador de usuario (Autoría propia).	41
Figura 3-20 Fragmento de código de Python para obtener y almacenar subtítulos de YouTube (Autoría propia).	42
Figura 3-21 Ejemplo de archivo de transcripción de vídeo del canal de la OTAN (Autoría propia).	43
Figura 3-22 Flujo del proceso de raspado de YouTube (Autoría propia).	44
Figura 3-23 Código de Python para inicio de sesión en Instagram (Autoría propia).	45
Figura 3-24 Código de Python para obtener los pies de foto de un perfil (Autoría propia).	45
Figura 3-25 Código de Python para limpiar caracteres especiales de un texto (Autoría propia).	46
Figura 3-26 Flujo del proceso de raspado de Instagram (Autoría propia).	46
Figura 3-27 Código de Python para manejo de documentos de texto (Autoría propia).	47
Figura 3-28 Fragmento del <i>dataset</i> utilizado (Autoría propia).	48
Figura 3-29 Disposición del <i>dataset</i> en Hugging Face (Autoría propia).	49
Figura 3-30 Ejemplo de formato de <i>prompt</i> para Llama 2 (Autoría propia).	49
Figura 3-31 Parámetros de entrenamiento (Autoría propia).	51
Figura 3-32 Esquema de LoRA y QLoRA [98].	52
Figura 3-33 Parámetros de cuantización y QLoRA (Autoría propia).	53
Figura 3-34 Código de Python para formatear los datos del ajuste fino (Autoría propia).	54
Figura 3-35 Gráfica evolución del <i>training loss</i> durante el ajuste (Autoría propia).	55
Figura 3-36 Código de Python para ejecutar Llama 2 (Autoría propia).	56
Figura 3-37 Bibliotecas usadas en la inferencia del modelo (Autoría propia).	56
Figura 3-38 Ejemplo de análisis de Marina Militare en YouTube (Autoría propia).	57
Figura 4-1 Definición de defensa nacional según DPEJ (Autoría propia).	58
Figura 4-2 Funcionamiento comando “start” y comando “info” (Autoría propia).	59
Figura 4-3 Funcionamiento proceso YouTube (Autoría propia).	60
Figura 4-4 Funcionamiento proceso Instagram (Autoría propia).	60
Figura 4-5 Funcionamiento proceso de análisis de documentos (Autoría propia).	60
Figura 4-6 Ejemplo de matriz de confusión [109].	69
Figura 4-7 Gráfico comparativo de sensibilidad (Autoría propia).	70
Figura 4-8 Gráfico comparativo de especificidad (Autoría propia).	71
Figura 4-9 Gráfico comparativo de falsos positivos (Autoría propia).	71
Figura 4-10 Gráfico comparativo de falsos negativos (Autoría propia).	72
Figura 4-11 Gráfico comparativo de tiempos de procesado para perfiles no relacionados con defensa (Autoría propia).	72
Figura 4-12 Gráfico comparativo de tiempos de procesado para perfiles relacionados con defensa (Autoría propia).	73

ÍNDICE DE TABLAS

Tabla 3-1 Evolución del <i>training loss</i> durante el ajuste (Autoría propia).	54
Tabla 4-1 Prueba de funcionamiento de comandos (Autoría propia).	59
Tabla 4-2 Análisis de raspado de YouTube (Autoría propia).	61
Tabla 4-3 Análisis de raspado de Instagram (Autoría propia).	62
Tabla 4-4 Análisis de errores de interacción (Autoría propia).	62
Tabla 4-5 Evaluación de respuestas del modelo sin ajustar para la plataforma X (Autoría propia).	64
Tabla 4-6 Evaluación de respuestas del modelo sin ajustar para la plataforma Instagram (Autoría propia).	65
Tabla 4-7 Evaluación de respuestas del modelo sin ajustar para la plataforma YouTube (Autoría propia).	65
Tabla 4-8 Evaluación de respuestas del modelo ajustado para la plataforma X (Autoría propia).	66
Tabla 4-9 Evaluación de respuestas del modelo ajustado para la plataforma Instagram (Autoría propia).	67
Tabla 4-10 Evaluación de respuestas del modelo ajustado para la plataforma YouTube (Autoría propia).	68
Tabla 4-11 Matriz de confusión obtenida para el modelo sin ajustar (Autoría propia).	69
Tabla 4-12 Matriz de confusión obtenida para el modelo ajustado (Autoría propia).	70

1 INTRODUCCIÓN Y OBJETIVOS

1.1 Contexto y motivación.

1.1.1 Antecedentes históricos.

Desde el amanecer de la existencia humana tal y como la conocemos, se han registrado recurrentes conflictos bélicos. Siendo estos casi los protagonistas de la historia y los determinantes que han forjado la sociedad actual.

La obtención de inteligencia y las prácticas de espionaje han ejercido un papel fundamental en la configuración y desarrollo de dichos enfrentamientos. Siendo innegable que el resultado de la mayoría de las contiendas ha dependido intrínsecamente del conocimiento del adversario. Un ejemplo notable de esto fue la operación “Mincemeat” [1]. Llevada a cabo por los aliados en la Segunda Guerra Mundial, fue una maniobra de engaño que consistió en dejar caer un cuerpo no identificado en la costa española junto con documentos falsos con el fin de desviar la atención del ejército alemán y facilitar el desembarco aliado en Sicilia en 1943. El éxito de esta operación fue un factor decisivo en la campaña del Mediterráneo.

Los primeros vestigios de servicios de inteligencia se remontan a la antigua Mesopotamia bajo el reinado de Sargón I de Acad en el tercer milenio a.C. Aunque algunos autores mencionan ataques selectivos bajo previa obtención de inteligencia en Asparn-Scheltz, actual Austria, que se remontan al quinto milenio a.C. [2].

Los métodos de obtención de inteligencia se han ido desarrollando con el paso de los años y su influencia en el transcurso de la historia ha dejado famosas figuras del espionaje como Jorge Juan, Margaretha Geertruida Zelle (Mata-Hari) o Isser Harel fundador del Mossad israelí [3].

Desde la antigua Roma a la revolución francesa, se han registrado multitud de casos de conspiración, espionaje y técnicas de obtención de inteligencia que han tenido una importante relevancia en el desarrollo de guerras, política y economía. Sin embargo, es a comienzos del siglo XX cuando el papel del espionaje alcanza un nivel superior tras la creación de redes internacionales de inteligencia haciendo que esta abarcara un papel global en el transcurso de la Primera Guerra Mundial y en sus años previos [2].

En la actualidad la información sigue siendo un factor de vital importancia en muchos aspectos de la sociedad, no obstante, su obtención se ha visto facilitada por el uso de nuevas tecnologías y el desarrollo de Internet. Desde su creación en 2002 es el Centro Nacional de Inteligencia el que se encarga de la obtención y evaluación de la inteligencia en España, aunque también podemos encontrar servicios de inteligencia en diferentes departamentos del Ministerio de Defensa y del Ministerio del Interior [2].

1.1.2 Orígenes del OSINT.

La definición de OSINT está recogida en la Ley 109-163 [4] de Estados Unidos como “aquella inteligencia producida a partir de información disponible públicamente que se recopila, explota y difunde de manera oportuna a una audiencia apropiada con el propósito de abordar un requisito de inteligencia específico” y por la publicación OTAN AJP-2.0.2001 [5] como “aquella inteligencia derivada de información disponible públicamente, así como otra información no clasificada que tiene una distribución o acceso público limitado”. En términos generales, el término OSINT se corresponde con el acrónimo en inglés *Open Source Intelligence* y hace alusión a toda aquella información que se extrae de fuentes públicas o de acceso libre como periódicos, redes sociales, noticieros y en general todas aquellas fuentes a las que puede acceder un ciudadano corriente.

Según la CIA, el uso de OSINT en la historia se remonta a principios del siglo XX. Durante la Segunda Guerra Mundial, con la creación del Research and Analysis del OSS, que se encargaba de recopilar y analizar información proveniente de fuentes abiertas y públicas, como periódicos, revistas, libros y transmisiones de radio del eje [5]. Sin embargo, el término “Inteligencia de Fuentes Abiertas” no se acuñó hasta la Guerra Fría, cuando las agencias de inteligencia occidentales formalizaron y ampliaron sus esfuerzos para recopilar y analizar información de fuentes públicas para obtener una comprensión más completa del panorama político, militar y social de sus adversarios. Desde entonces, con el avance de la tecnología de la información y la comunicación, el OSINT ha evolucionado significativamente y se ha convertido en una parte fundamental de las operaciones de inteligencia a nivel gubernamental y en el ámbito civil [5].

1.1.3 La importancia de las redes sociales.

En la actualidad la mayor fuente de inteligencia y datos de libre acceso se encuentra en el contexto del *social media*, convirtiendo las plataformas sociales en las fuentes objetivo de las herramientas de obtención y análisis de OSINT. Además, las redes sociales pueden suponer canales de comunicación para determinados grupos y asociaciones de especial interés como grupos supremacistas, terroristas o redes criminales (véase figura 1-1). A su vez, al contar con alcance mundial, las redes sociales son una potente herramienta de cara a organizar movimientos ideológico-políticos, manifestaciones, protestas y en general cualquier actividad que englobe colectivos [6]. Por otro lado, es un hecho que se ha registrado un notable crecimiento en la proliferación de noticias falsas y teorías conspirativas, las cuales pueden llegar a tener un impacto significativo en el ámbito público, cuestionando la confianza en las instituciones y desafiando la integridad del debate democrático. Este fenómeno plantea serias implicaciones para la calidad de la información y la capacidad de discernimiento de la ciudadanía, cuestionando la esencia misma de una sociedad fundamentada en la transparencia, la veracidad y la participación informada. Es por esto que es esencial vigilar y proteger estas herramientas sociales con el fin de garantizar una información veraz, precisa y actualizada.



Figura 1-1 Ejemplo de uso ilícito de X/Twitter [7].

Aunque en el mercado existen potentes herramientas de OSINT, estas pueden presentar una serie de problemas a abordar que afectarían a su utilidad y efectividad. En primer lugar, las herramientas comerciales presentan limitaciones en la personalización de enfoque en las necesidades específicas del usuario o del contexto de la investigación. Por otro lado, en un entorno cambiante y turbulento como son las redes sociales, las herramientas de análisis pueden tener dificultades de adaptación para poder continuamente abordar nuevas fuentes de información y métodos de ocultamiento utilizados por los usuarios. Además, algunas de las herramientas existentes de OSINT pueden ser costosas, lo que limita su accesibilidad para individuos y organizaciones con recursos financieros limitados. Destacando también que al usar proveedores externos puedes exponerte a no realizar una obtención totalmente pasiva y dejar rastros tras el análisis. Por otro lado, el contar con una interfaz sencilla en formato chat, puede facilitar a los usuarios su entendimiento y manejo además de facilitar su acceso y portabilidad.

Es por ello que el desarrollo de una herramienta propia de OSINT puede resultar la mejor opción para obtener una herramienta flexible, adaptable, eficaz, segura y de bajo costo.

1.1.4 La importancia de la inteligencia artificial.

En los últimos años, la IA ha transformado radicalmente la forma de interacción entre los humanos y las máquinas. Los grandes modelos del lenguaje (*Large Language Model*, LLM) y el desarrollo de *chatbots* como ChatGPT [8], son hitos clave en este avance ya que, alimentados por enormes conjuntos de datos y el aprendizaje profundo, permiten una comunicación fluida hombre-máquina. Estas herramientas tienen infinidad de aplicaciones desde servicios de atención al cliente hasta diagnósticos médicos. Esto se debe a que dichos agentes virtuales son capaces de mantener conversaciones complejas y personalizadas, impulsando la eficiencia operativa y la accesibilidad a la información.

Por otro lado, la IA empieza a tomar protagonismo en el campo del análisis de datos ya que permite extraer información significativa y tomar decisiones más precisas y, mediante algoritmos avanzados de aprendizaje automático y técnicas de procesamiento, puede identificar patrones complejos, revelar correlaciones ocultas y predecir tendencias futuras a partir de grandes conjuntos de datos. Esto no solo agiliza el proceso de análisis, sino que también mejora la calidad de los resultados, lo que optimiza el procesamiento de datos y facilita la toma de decisiones estratégicas.

1.2 Objetivos y metodología.

El objetivo de este proyecto se centra en la creación de una herramienta digital que cumpla con las características mencionadas anteriormente. Esta herramienta deberá ser capaz de analizar contenido públicamente disponible en cuentas de redes sociales como Instagram, YouTube y X/Twitter para determinar si dicho contenido está relacionado parcial o totalmente con temas del ámbito de defensa con el fin de encontrar fuentes de información de potencial interés para aplicaciones OSINT.

Para garantizar la adaptabilidad y eficacia de la herramienta, se desarrollará utilizando *Large Language Models* (LLM) pre-entrenados, centrados en el procesamiento del lenguaje natural. Se llevará a cabo un ajuste fino del modelo LLM para optimizar su respuesta en relación con los temas de defensa. Este ajuste implicará el entrenamiento del modelo con un conjunto de datos específicamente seleccionado que incluya artículos relacionados con el ámbito de defensa, previamente extraídos de un corpus construido mediante la *Application Programming Interface* (API) de Wikipedia.

Con el fin de garantizar un bajo costo y una interfaz sencilla además de portable, la herramienta se implementará a través de un *chatbot* de Telegram. Se ha seleccionado la API de Telegram debido a sus características de seguridad, a su uso de libre acceso y a sencillez de la interfaz en formato de chat. Además, se empleará el LLM de la empresa Meta conocido como Llama 2 [9, p. 2], el cual también está

disponible para su uso libre y ofrece una amplia gama de capacidades de procesamiento de lenguaje natural.

En cuanto a la obtención de datos de las cuentas a analizar, se utilizará la API de la propia aplicación en la mayoría de los casos. Sin embargo, para X/Twitter, debido al elevado costo de su API de suscripción, se utilizará un archivo de ejemplo que contendrá una serie de tuits descargados previamente, como prueba de concepto.

1.3 Estructura de la memoria.

A continuación, se presentará la estructura con la que se desarrollará la redacción de la memoria, que consta de 5 capítulos:

- Capítulo 1 (Introducción y Objetivos): en este capítulo, se ofrece una breve introducción al contexto del proyecto, destacando su relevancia y la motivación que impulsó su desarrollo. Se explicará la estructura de la memoria para proporcionar una visión general de los temas que se abordarán.
- Capítulo 2 (Estado del arte): este capítulo comprende un análisis exhaustivo de los antecedentes del proyecto. Se examinará el proceso de obtención de inteligencia, incluyendo una exploración de las diversas fuentes de obtención disponibles. Además, se estudiarán los aspectos relacionados con el *Machine Learning* y los LLM. Se introducirá el ámbito de las redes sociales y se discutirán las aplicaciones relevantes para la explotación de datos.
- Capítulo 3 (Desarrollo del TFG): en este capítulo, se detalla el proceso de desarrollo de la herramienta. Se presentan los pasos seguidos para obtener el producto final, incluyendo una descripción exhaustiva de cada etapa del proceso. Se analiza el código utilizado para los procesos de análisis del bot, proporcionando una visión profunda de la implementación técnica.
- Capítulo 4 (Prueba y validación): en este capítulo, se procede a validar los resultados obtenidos en el capítulo anterior. Se utilizarán métodos estadísticos y pruebas variadas para analizar los resultados y ajustar los parámetros del modelo según sea necesario. Se llevarán a cabo pruebas en diferentes plataformas para evaluar la eficiencia de la herramienta en cada una de ellas.
- Capítulo 5 (Conclusiones y Líneas Futuras): finalmente, se realiza una interpretación de los resultados obtenidos en los capítulos anteriores. Se justifican los hallazgos y se extraen conclusiones concretas del proyecto. Además, se discuten posibles direcciones futuras para la investigación y el desarrollo de la herramienta.

Por último, se incluirá un listado de la bibliografía utilizada, así como una serie de anexos sobre distintos aspectos, sociales, ambientales, económicos, éticos y técnicos del trabajo realizado.

2 ESTADO DEL ARTE

2.1 Inteligencia y espionaje.

2.1.1 Modelo de obtención de inteligencia

Con el paso de los años, las tecnologías y los medios de obtención de información han evolucionado notoriamente. Por ello han surgido diversas comunidades y agencias enfocadas al proceso de gestión de la inteligencia. El modelo de procesamiento y evaluación de la información es a día de hoy un tema controvertido, habiendo diferentes interpretaciones y doctrinas sobre este. Sin perjuicio a esto, y extrapolando las distintas interpretaciones, se puede destacar una sistema base primitivo del que divergen los demás modelos [4, 5].

Este sistema es utilizado por grandes agencias de inteligencia como la CIA, el CNI o el USAICoE [10], es de carácter cíclico y se divide en seis fases diferenciadas (véase figura 2-1): [11]

- Planificación: se trata de una etapa crucial que implica determinar las áreas estratégicas de interés y las necesidades de información específicas. Se establecen prioridades que derivan en objetivos generales de información, considerando la disponibilidad y fiabilidad de las fuentes. Es importante tener en cuenta que la inteligencia se clasifica en tres tipos: básica (general y permanente), actual (periódica o en respuesta a demandas específicas) y estimativa (orientada a prever la evolución de situaciones específicas).
- Toma de datos: esta fase involucra la adquisición, selección, autenticación y reunión de datos e información mediante diversos medios tecnológicos, humanos o documentales. La tipología y características de estos medios se describen posteriormente en este documento.
- Procesamiento: esta fase consiste en el proceso de control y conservación de datos. Se enfoca en la conversión y estructuración de información para su posterior uso en mensajes. Para gestionar la abundancia de datos, se utilizan sistemas avanzados y herramientas tecnológicas como la minería de datos y agentes inteligentes. La conservación de datos requiere establecer lugares seguros para su almacenamiento, implementar medidas preventivas contra el deterioro y desarrollar estrategias para hacer frente a la obsolescencia técnica. La gestión efectiva de la información es esencial para prevenir posibles fallos en el servicio y promover la colaboración entre diversos departamentos y servicios de inteligencia.
- Análisis y producción: en esta fase, se realiza la extracción rápida y precisa de información de los almacenes de datos con el propósito de obtener conocimiento. Este proceso es el encargado de transformar la información en inteligencia coherente, concisa e interpretable. Dentro de esta fase, se llevan a cabo tres acciones clave: evaluar datos para determinar su utilidad, integrar información de diversas fuentes para crear una visión completa y realizar la interpretación de los datos.
- Comunicación: esta etapa consiste en la entrega segura del documento contenedor del conocimiento obtenido al solicitante de este. Es una práctica común la difusión horizontal de inteligencia de bajo nivel entre departamentos y analistas para aumentar la circulación de información y el cruce de datos.
- Evaluación: es una de las etapas más importantes del proceso ya que permite realizar una retroalimentación con objeto de determinar si se han alcanzado o no los objetivos, corregir fallos en la planificación y mejorar el sistema en futuros de ciclos de obtención de inteligencia. Es por este motivo que este sistema se considera un proceso cíclico, ya que debe mantenerse en constante evolución debido a la situación global cambiante, la necesidad de información actualizada y al posible cambio de prioridades o estrategias.



Figura 2-1 Ciclo de la inteligencia (Autoría propia).

En lo que a la toma de datos se refiere, se trata de un proceso complicado, ya que se debe tener en cuenta el carácter y veracidad de la información y las fuentes. Es por esto que dichas fuentes se pueden clasificar en distintos grupos para evaluar su fiabilidad, utilidad y vigencia. Estos grupos determinan principalmente el origen y la clase de información obtenida. Podemos clasificarlos en seis bloques (véase figura 2-2): [12]

- HUMINT o inteligencia de fuentes humanas: es recogida a través de fuentes humanas sin ayuda de medios auxiliares (cámaras, grabadoras, etc.). Depende directamente de la actuación del personal a través de contactos, informadores o colaboradores.
- SIGINT o inteligencia de señales: se obtiene del procesamiento de señales y transmisiones de cualquier clase. Estas pueden ser acústicas o electromagnéticas.
- MASINT o inteligencia de mediciones: se obtiene de la medición de señales de fenómenos físicos como el movimiento o la radiación.
- IMINT o inteligencia de imagen: se obtiene a través del análisis de imágenes y fotografías sin importar su medio de obtención (cámaras, satélites, etc.).
- TECHINT o inteligencia técnica: se obtiene mediante técnicas de procesamiento tecnológico de datos provenientes de las demás fuentes.
- OSINT o inteligencia de fuentes abiertas: se obtiene a través del análisis de recursos de dominio público.

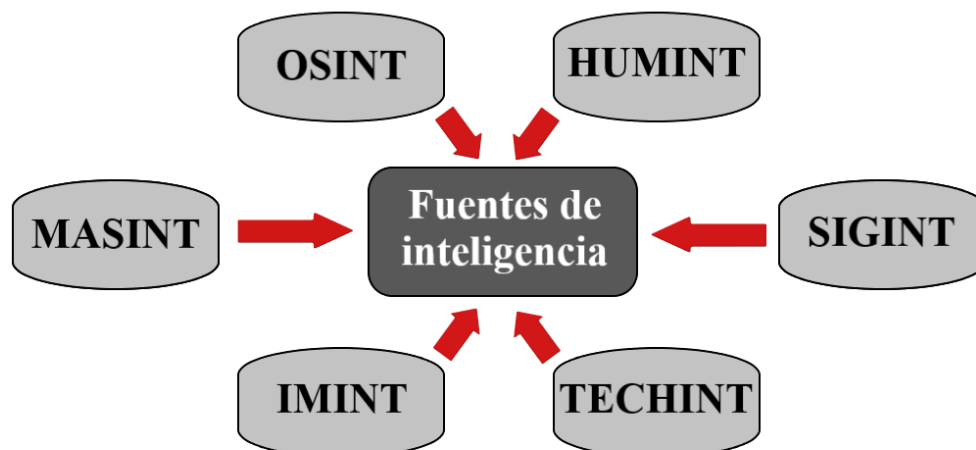


Figura 2-2 Fuentes de obtención de inteligencia (Autoría propia).

2.1.2 OSINT y sus aplicaciones.

Entre todas las fuentes de información existentes, la inteligencia de fuentes abiertas adquiere una importante relevancia al tener en cuenta la cantidad de datos de dominio público que son liberados a diario. El uso de Internet y las redes sociales han incrementado notablemente la cantidad de datos, así como su accesibilidad. Según la revista “Forbes” [13] cada día se publican 2,5 trillones de bytes en la red. Para llegar a estimar la cantidad de datos de fuentes abiertas generados, a este número habría que adicionar la cantidad de información física que es publicada en periódicos, documentos, boletines, etc. Además, con el desarrollo de la tecnología y el Internet de las cosas, esta estimación se ve incrementada diariamente, posicionando así a la inteligencia de fuentes abiertas como uno de los métodos con mayor alcance y generalmente el más actualizado.

Las tecnologías que permiten extraer información de fuentes abiertas son generalmente de fácil acceso y manejo sencillo, haciendo que el usuario medio haya usado estas de algún modo u otro al consultar datos en línea [14].

Las grandes agencias de seguridad e inteligencia de diversos países como Reino Unido, Estados Unidos o Corea del Sur han reconocido la importancia del uso de técnicas de análisis de OSINT ya que son muy útiles a la hora de rastrear información en línea, obtener perfiles de criminales o incluso hacer frente a amenazas cibernéticas [14].

En el ámbito corporativo, las empresas utilizan también fuentes OSINT para investigar nuevos mercados, analizar competidores o planificar actividades de marketing. Esto se debe a que no se requiere dedicar excesivo presupuesto para utilizar estas técnicas y ofrecen una ventaja competitiva que puede llegar a ser decisiva para introducirse en un mercado [14].

Además de todo lo mencionado, las fuentes abiertas pueden utilizarse de manera maliciosa. Las organizaciones criminales podrían llegar a utilizarlas para planear ataques. Pueden recopilar información sobre un objetivo, realizar reclutamiento en redes sociales, divulgar propaganda e incluso obtener material sensible que sea accidentalmente liberado [14].

Como podemos observar, las fuentes OSINT pueden tener un notable impacto en determinadas áreas de acción. No obstante, se debe tener en cuenta que dichas fuentes tienen ventajas y desventajas apreciables. Es innegable que mediante el uso de fuentes abiertas podemos obtener información rápida o incluso en tiempo real, ya que se pueden utilizar de manera legal y sin exponerse a riesgos innecesarios,

por lo general las fuentes son de fácil acceso y con un coste de implementación ínfimo. Por otro lado, la gran cantidad de información liberada hace que en ocasiones sea complicado extraer los datos de interés, además el alcance de los datos suele ser global por lo que otras agencias o servicios también tendrán acceso a esta información, añadiendo a esto también la posibilidad de recibir ciberataques.

2.1.3 Herramientas de OSINT.

Para hablar de las herramientas, técnicas y las distintas formas de obtención de OSINT, debemos tener en cuenta que existen tres tipos de obtención de inteligencia de fuentes abiertas [15]:

- Obtención Pasiva: se basa en la búsqueda de información sin conocimiento por parte del objetivo. Para ello, es esencial mantener el anonimato y la seguridad del analista. La información obtenida es limitada ya que no se interactúa directamente con el objetivo, solo se intercepta el tráfico y contenidos que este genere.
- Obtención Semi-pasiva: el analista genera huella en la red ya que realiza determinadas consultas a la fuente de información. Por lo general se obtiene más información que en la obtención pasiva, pero se asumen más riesgos de ser detectado por el objetivo.
- Obtención Activa: en este modo, el analista interactúa directamente con el objetivo. Este método no es discreto y puede dejar rastros muy claros en los registros del objetivo, pero proporciona más datos, como puertos abiertos, protocolos empleados, vulnerabilidades, entre otros.

Analizando la clasificación anterior podemos decir que el método más recomendable es la obtención pasiva debido al anonimato y seguridad que aporta, es por esto que la mayoría de las herramientas OSINT se agrupan en ese escenario. Sin embargo, la obtención pasiva puede no ser la más eficiente en todos los casos, ya que hay multitud de fines del análisis de OSINT y por ello infinidad de tipos de herramientas para su explotación. Podemos clasificar estas herramientas en los siguientes campos:

- Herramientas de búsqueda web: Google [16].
- Herramientas de redes sociales: XPro [17].
- Herramientas de análisis de datos: Maltego [18].
- Herramientas de monitorización de noticias: Google News [19].
- Herramientas de análisis de imágenes y videos: Google Lens [20].
- Herramientas de monitorización de tráfico de red: Wireshark [21].
- Herramientas de análisis de documentos: PDF Examiner [22].
- Herramientas de monitorización de eventos en tiempo real: Dataminr [23].
- Herramientas de investigación de dominios y direcciones IP: WHOIS Lookup [24].
- Herramientas de búsqueda en la Dark Web: Tor Browser [25].
- Herramientas de análisis de ficheros: ExifTool [26].
- Herramientas de análisis de redes y comunicaciones: NetWitness Investigator [27].
- Herramientas de análisis de código abierto y código fuente: GitMiner [28].
- Herramientas de análisis de documentos filtrados y filtraciones de datos: WikiLeaks [29].
- Herramientas de análisis de reputación y perfilado de usuarios: Social-Analyzer [30].
- Herramientas de análisis de inteligencia empresarial y competencia: Owler [31].
- Herramientas de análisis de amenazas y vulnerabilidades: NVD [32].
- Herramientas de análisis de comportamiento en línea y perfiles: Google Analytics [33].

2.2 Inteligencia artificial.

2.2.1 Introducción e historia.

Si bien la noción de inteligencia artificial (IA) se desarrolló durante la segunda mitad del siglo XX, sus antecedentes están vinculados tanto a la ficción literaria como a los avances tecnológicos de la época. En 1942, Isaac Asimov presentó las Tres Leyes de la Robótica en su obra “Runaround”, y Alan Turing contribuyó al campo con la máquina “The Bombe”, utilizada para descifrar códigos enemigos durante la Segunda Guerra Mundial. Estos acontecimientos plantearon preguntas sobre la posibilidad de que las máquinas pudieran exhibir inteligencia similar a la humana [34].

En 1950, Turing propuso el denominado “Test de Turing” para evaluar la capacidad de una máquina mostrando comportamiento inteligente. Seis años después, en 1956, Marvin Minsky y John McCarthy organizaron el Proyecto de Investigación de Verano sobre Inteligencia Artificial de Dartmouth, estableciendo así las bases de la investigación moderna en IA [34]. El objetivo del proyecto era desarrollar sistemas capaces de simular el comportamiento humano inteligente. Desde entonces, la IA ha evolucionado rápidamente, impulsando avances tecnológicos en áreas como el procesamiento de lenguaje natural, la visión por computadora y el aprendizaje automático. Este progreso ha llevado a una exploración continua de las capacidades y limitaciones de la IA, así como a debates sobre su impacto en la sociedad y la ética de su uso [34].

Tras la Conferencia de Dartmouth en 1956, se vivió un período de casi dos décadas de notables avances en el campo de la IA. Ejemplos notables incluyen el programa informático ELIZA, desarrollado por Joseph Weizenbaum en el MIT entre 1964 y 1966 el cual simulaba conversaciones humanas y el programa General Problem Solver, creado por Herbert Simon, Cliff Shaw y Allen Newell, que resolvía automáticamente ciertos problemas simples [34].

Estos éxitos inspiraron una financiación significativa para la investigación en IA. Sin embargo, en 1973, tanto el Congreso de EE. UU. como el gobierno británico comenzaron a cuestionar el elevado gasto en este campo. Un informe de James Lighthill cuestionó las expectativas optimistas, sugiriendo que las máquinas solo alcanzarían un nivel de habilidad de “aficionado experimentado” en juegos como el ajedrez. En respuesta, el gobierno británico y luego el estadounidense redujeron drásticamente su apoyo a la investigación en IA estancando este campo durante años, lo que marcó el comienzo de un período conocido como el “Invierno de la IA” [34].

Estos programas primitivos como ELIZA o el General Problem Solver eran sistemas expertos, es decir, colecciones de reglas que asumen que la inteligencia humana puede ser formalizada y reconstruida en un enfoque de arriba hacia abajo como una serie de declaraciones “*if-then*”. Sin embargo, los sistemas expertos funcionan mal en determinadas áreas ya que no pueden realizar tareas que requieran un aprendizaje o entrenamiento previo. Para tales tareas es necesario que un sistema sea capaz de interpretar correctamente datos externos, aprender de dichos datos y usar esos aprendizajes para lograr objetivos y tareas específicas mediante una adaptación flexible. Es por esto que la realidad quedó muy por debajo de las expectativas y el desarrollo de IA se estancó durante años [34].

Desde la década de 1940 se ha discutido sobre los mejores métodos estadísticos para lograr un algoritmo plenamente efectivo. En esa década, el psicólogo canadiense Donald Hebb desarrolló una teoría de aprendizaje conocida como “Aprendizaje Hebbiano” el cual replicaba el proceso de las neuronas en el cerebro humano. Esto condujo a la creación de la investigación sobre Redes Neuronales Artificiales. Sin embargo, el progreso en el desarrollo de redes neuronales artificiales se vio obstaculizado en la década de 1960 y 1970, en parte debido a las limitaciones tecnológicas y al reconocimiento de las limitaciones en los perceptrones y redes neuronales para resolver problemas que no eran linealmente separables, reflejado en los trabajos de Marvin Minsky y Seymour Papert [35]. Sin embargo, estos problemas fueron solo algunos de los muchos factores que contribuyeron al estancamiento conocido como el ‘invierno de la IA’, que también incluyó cambios en la financiación y expectativas poco realistas sobre el potencial de la IA en ese momento [34].

No fue hasta 2012 cuando las redes neuronales artificiales tuvieron su resurgimiento en forma de aprendizaje profundo cuando un programa desarrollado por Google denominado “AlphaGo” pudo vencer al campeón mundial en el juego de mesa “Go”. Hoy en día, las redes neuronales artificiales y el *Deep Learning* forman la base de la mayoría de las aplicaciones que conocemos bajo el nombre de IA. Son la base de los algoritmos de perfilamiento de imágenes utilizados por Instagram, los algoritmos de procesamiento del lenguaje que alimentan interfaces como “chatGPT” e incluso los altavoces inteligentes como Alexa de Amazon [34].

2.2.2 Aprendizaje automático.

El aprendizaje automático o *Machine Learning* es una rama de la IA que se centra en el desarrollo de algoritmos y modelos que permiten a las computadoras aprender y mejorar su rendimiento en tareas específicas a través de la experiencia. Este método, se basa en la idea de que las computadoras pueden aprender patrones intrínsecos en los datos y utilizar estos patrones para hacer predicciones o tomar decisiones sin una programación explícita. En vez de seguir reglas específicas como los sistemas expertos, los algoritmos de aprendizaje automático utilizan técnicas estadísticas para aprender de los datos y mejorar su rendimiento con la experiencia. Estos modelos, pueden clasificarse en tres grupos en función de su modo de aprendizaje [36]:

- Aprendizaje supervisado: el aprendizaje supervisado se basa en el uso de conjuntos de datos etiquetados para entrenar los algoritmos donde cada unidad de datos se asocia con una etiqueta lógica concreta. Para el entrenamiento se enseña al algoritmo a relacionar las entradas con las salidas deseadas, permitiendo así, hacer predicciones precisas sobre futuros datos no etiquetados [34, 35].
- Aprendizaje no supervisado: el aprendizaje no supervisado se basa en la búsqueda de patrones y estructuras en un grupo de datos no etiquetados para realizar su clasificación posteriormente [34, 35].
- Aprendizaje por refuerzo: el aprendizaje por refuerzo se basa en la interacción de un modelo con un entorno dinámico determinado. El algoritmo se entrenará a través de la retroalimentación que recibe en forma de recompensas o penalizaciones. El objetivo del modelo es desarrollar una política óptima que maximice la recompensa acumulada a lo largo del tiempo [34, 35].

El aprendizaje automático ha experimentado un desarrollado exponencial revolucionando radicalmente la manera en que las computadoras procesan la información y toman decisiones. Además, este tiene aplicaciones en una amplia gama de campos como el análisis de datos o la conducción autónoma, permitiendo a las máquinas aprender de los datos y mejorar su rendimiento con el tiempo sin intervención humana directa. Para implementar los diversos tipos o modelos de *Machine Learning*, ya sea supervisado, no supervisado o de aprendizaje reforzado, es fundamental emplear una serie de algoritmos específicos. Los principales algoritmos de aprendizaje automático supervisado son los siguientes [37]:

- Árboles de decisión: estos algoritmos se comportan como herramientas estructurales y son útiles para tomar decisiones basadas en criterios predefinidos. Utilizan un enfoque ramificado que representa las diferentes posibilidades, generando nodos que representan variables específicas y ramas que muestran los resultados del algoritmo.
- Algoritmos bayesianos: los algoritmos bayesianos se basan principalmente en la aplicación del teorema de Bayes. Son utilizados para clasificar valores en función de un conjunto de características predefinidas utilizando la probabilidad para predecir clases o categorías.

- Algoritmos de regresión logística: estos algoritmos se emplean cuando se es necesario estimar y relacionar variables de un conjunto. La regresión consiste en intentar predecir los valores de una variable continua (variable dependiente) a partir de una o varias variables independientes.
- Máquinas de vectores de soporte: estos algoritmos resultan muy útiles tanto para problemas de clasificación como para regresión. Su funcionamiento se basa en encontrar el hiperplano óptimo que mejor separa las clases en un espacio multidimensional. Es decir, funcionan encontrando una línea o superficie que separa claramente diferentes grupos en un espacio de múltiples dimensiones.
- Métodos *ensemble*: son algoritmos de aprendizaje que se basan en la construcción de un conjunto de clasificadores, los cuales combinan sus predicciones para clasificar nuevos puntos de datos mediante un voto ponderado.

Los principales algoritmos de aprendizaje automático no supervisado son los siguientes [37]:

- Algoritmos de agrupamiento: los algoritmos de agrupación son esenciales en el campo del aprendizaje no supervisado, estos organizan y categorizan datos no etiquetados mediante una búsqueda iterativa de grupos dentro del conjunto de datos.
- Algoritmos *clustering*: consisten en organizar un conjunto de elementos de manera que aquellos dentro de un mismo grupo, o clúster, sean más afines entre sí que con los pertenecientes a otros grupos.
- Análisis de Componentes Principales (PCA): es una técnica estadística que emplea una transformación ortogonal para convertir un conjunto de observaciones de variables, posiblemente correlacionadas, en un conjunto de valores de variables linealmente no correlacionadas, conocidas como componentes principales.
- Descomposición en Valores Singulares (SVD) es una técnica del álgebra lineal que descompone una matriz compleja en tres componentes: dos matrices unitarias y una matriz diagonal, lo que facilita el análisis de los datos y la reducción de la dimensionalidad.
- Análisis de Componentes Independientes (ICA) es un método estadístico que identifica los factores subyacentes en conjuntos de variables, mediciones o señales, al suponer que los datos observados son mezclas lineales de variables latentes desconocidas, las cuales son no gaussianas y mutuamente independientes.

Métodos avanzados de aprendizaje automático:

- Redes neuronales: las redes neuronales emulan el funcionamiento del cerebro humano. Principalmente son sistemas de datos interconectados entre sí organizadas en capa de una red.
- Algoritmos de aprendizaje profundo: los algoritmos de aprendizaje profundo o *Deep Learning* funcionan mediante el uso de varias capas de redes neuronales en serie. Extraen representaciones complejas de los datos en cada nivel para realizar tareas de análisis avanzado. (Véase figura 2-3)

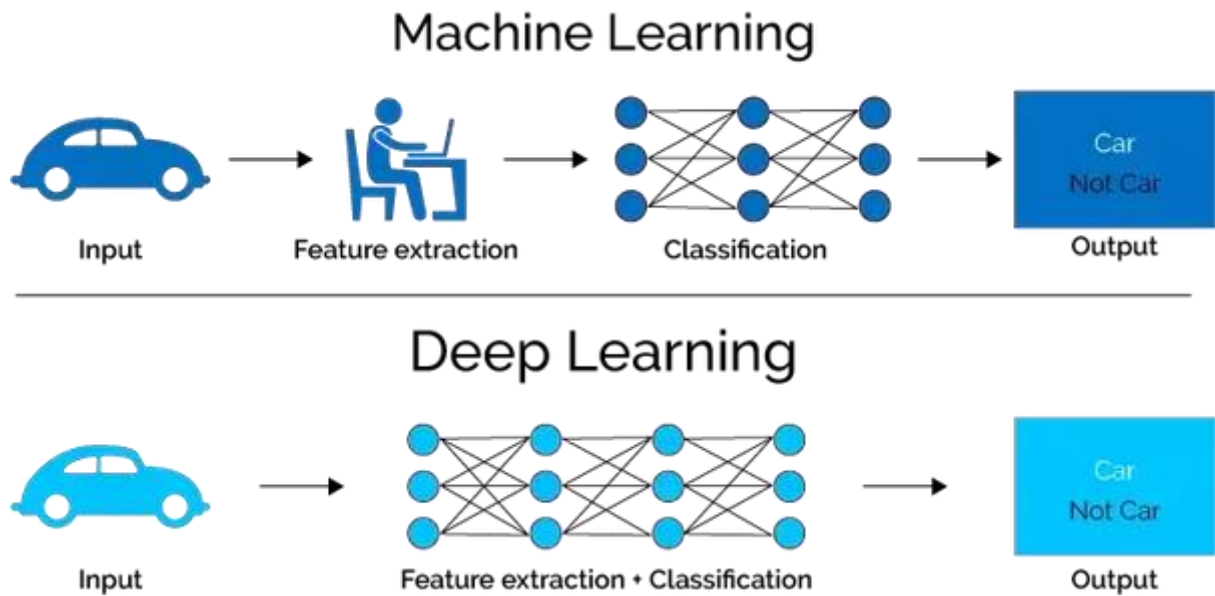


Figura 2-3 *Machine Learning vs Deep Learning.*

Estos son únicamente algunos de los principales tipos de algoritmo que podemos encontrar en el conjunto de herramientas del *Machine Learning*, pero se debe tener en cuenta de que existe una amplia variedad de ellos. Estos ofrecen soluciones para una amplia variedad de problemas y aplicaciones. Su implementación adecuada y comprensión son esenciales para aprovechar al máximo el potencial del aprendizaje automático en diversos los múltiples campos y sectores donde estas tecnologías son aplicables.

2.2.3 Deep learning y redes neuronales artificiales.

Una vez analizados los conceptos fundamentales de la IA, su evolución histórica y los principios básicos del aprendizaje automático, nos centraremos en una de las áreas más trascendentales y dinámicas de este campo, el aprendizaje profundo.

El aprendizaje profundo es una disciplina del aprendizaje automático que se enfoca en la construcción y entrenamiento de redes neuronales artificiales compuestas por múltiples capas, permitiendo la representación jerárquica de los datos. A diferencia de los enfoques tradicionales de aprendizaje automático, el aprendizaje profundo se caracteriza por la capacidad de extraer automáticamente características relevantes y de alto nivel de los datos de entrada.

En el núcleo del aprendizaje profundo residen las redes neuronales profundas, que consisten en múltiples capas de unidades de procesamiento llamadas neuronas, organizadas en una arquitectura interconectada. Estas redes utilizan un proceso iterativo de propagación hacia adelante y hacia atrás para aprender automáticamente a partir de los datos de entrada, ajustando los pesos de las conexiones entre las neuronas para minimizar una función de pérdida predefinida. (Véase figura 2-4)

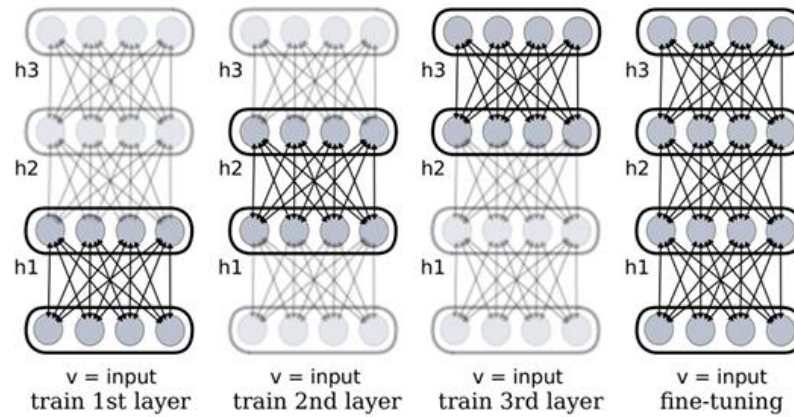


Figura 2-4 Ejemplo de entrenamiento de una red neuronal [38].

Las redes neuronales son herramientas altamente adaptables cuya estructura y funcionamiento pueden variar considerablemente en función del propósito específico para el que se las emplee. Esta variabilidad se logra mediante la aplicación de diversos algoritmos, métodos probabilísticos, técnicas de optimización de gradientes e inicialización de pesos. Por ejemplo, en el ámbito del procesamiento de lenguaje natural, se utilizan las redes neuronales recurrentes (RNN), diseñadas específicamente para trabajar con secuencias de datos, lo que las hace ideales para tareas como la traducción automática o el análisis de texto (véase figura 2-5). Por otro lado, para aplicaciones de visión computacional, como el reconocimiento de objetos en imágenes, se emplean las redes neuronales convolucionales (CNN), que destacan por su capacidad para detectar patrones locales en datos de alta dimensión (véase figura 2-6).

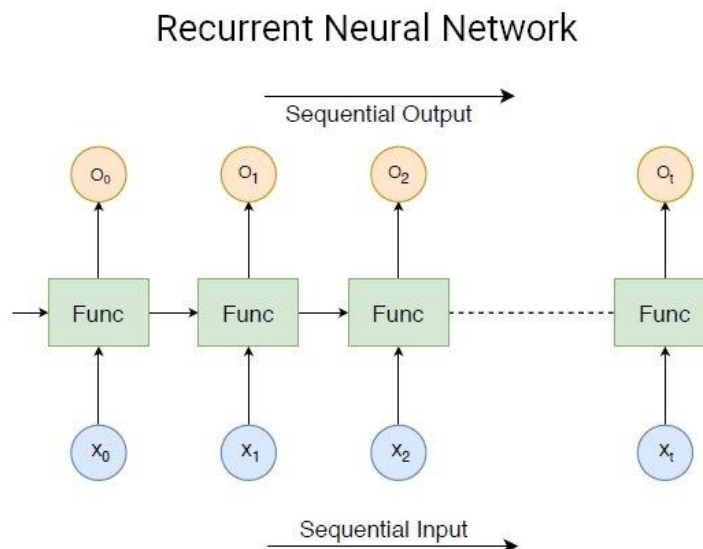


Figura 2-5 Ejemplo de RNN [39].

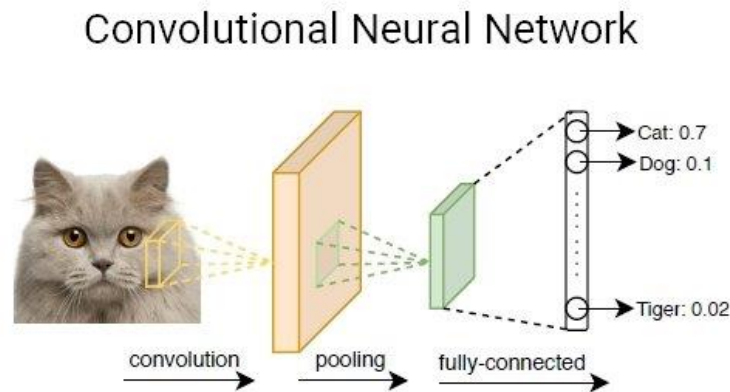


Figura 2-6 Ejemplo de CNN [39].

Además de la elección de la arquitectura de red adecuada, se utilizan distintas técnicas de optimización para mejorar el rendimiento del modelo. Por ejemplo, el algoritmo Adam [40] se utiliza para adaptar la tasa de aprendizaje durante el proceso de entrenamiento mediante una variante del descenso de gradiente estocástico. Es ampliamente empleado debido a su capacidad, lo que acelera la convergencia y evita estancamientos en mínimos locales. Otro aspecto crucial es la inicialización de pesos, que impacta significativamente en el rendimiento y la estabilidad del modelo. Para ello hay herramientas como la inicialización de Xavier [41], una técnica comúnmente utilizada que ajusta los pesos iniciales de manera que la propagación bidireccional de la información se mantenga estable a lo largo de las capas de la red. Esta técnica es especialmente útil para evitar problemas como el desvanecimiento o la explosión del gradiente, que pueden obstaculizar el entrenamiento eficaz de la red.

Como podemos observar el aprendizaje profundo está muy desarrollado en cuanto a algoritmos y técnicas de implementación, no obstante, no se puede determinar que su uso sea completamente eficaz y este exento de problemas en un cien por cien. El *Deep Learning* tiene al menos dos problemas de importancia: el *overfitting* y su comportamiento como caja negra [42].

El *overfitting* o sobreajuste en español, se encuentra en situaciones en las que un modelo entrenado se ajusta tan precisamente a su entorno de trabajo que se vuelve impreciso e incluso incapaz de funcionar con conjuntos diferentes de datos. Este problema puede surgir debido a la alta capacidad de representación de las redes neuronales profundas, que les permite aprender relaciones altamente complejas entre las entradas y las salidas. Cuando se entrena con un conjunto de datos suficientemente grande y flexible, el modelo puede encontrar patrones sutiles y específicos en los datos de entrenamiento que no son generalizables y pueden considerarse ruido. El sobreajuste se manifiesta cuando el error de entrenamiento continúa disminuyendo mientras que el error de validación (o prueba) comienza a aumentar después de cierto punto durante el entrenamiento. Esto indica que el modelo está comenzando a memorizar los datos de entrenamiento en lugar de aprender patrones subyacentes que se pueden generalizar a nuevos datos [42].

Para abordar el *overfitting*, se utilizan diversas técnicas de regularización, como la penalización de pesos grandes, la introducción de términos de regularización en la función de pérdida, o el uso de técnicas como el *dropout*, que inhibe selectivamente ciertas conexiones neuronales durante el entrenamiento para evitar la coadaptación de las neuronas. Además, se emplean estrategias de validación cruzada y técnicas de monitoreo del rendimiento durante el entrenamiento, como la parada temprana, que detiene el proceso de entrenamiento cuando el rendimiento en un conjunto de validación comienza a empeorar. Estas medidas ayudan a garantizar que el modelo generalice adecuadamente a datos no vistos y mantenga un rendimiento óptimo en aplicaciones del mundo real [42].

Por otro lado, las redes neuronales profundas, especialmente aquellas con muchas capas y parámetros, son inherentemente opacas. Es por esto que se dice que se comportan como cajas negras. Este comportamiento se debe a que, a medida que los datos se propagan a través de múltiples capas de neuronas interconectadas, se vuelven cada vez más difíciles de interpretar y entender para los seres humanos. A menudo, no está claro cómo una red neuronal profunda llega a una determinada predicción o decisión, lo que dificulta la explicación de su funcionamiento a los usuarios y expertos en la materia [42].

Para abordar estos problemas, los investigadores están trabajando en el desarrollo de técnicas y herramientas para mejorar la interpretabilidad de los modelos de aprendizaje profundo. Esto incluye métodos para visualizar y analizar las características aprendidas por el modelo, así como enfoques para cuantificar y mitigar el sesgo y la discriminación en los datos y en las decisiones del modelo [42].

A pesar de estos problemas generales, el aprendizaje profundo ha demostrado ser altamente efectivo en una amplia gama de aplicaciones, incluyendo el reconocimiento de imágenes, el procesamiento del lenguaje natural o la conducción autónoma y se esperan grandes avances en su desarrollo durante los próximos años.

2.2.4 Transformers y LLM.

Entre los distintos tipos de redes neuronales profundas que existen, destacan notablemente los modelos de tipo *Transformers* ya que permiten desarrollar herramientas de procesamiento del lenguaje natural (Natural Language Processing, NLP). Estas herramientas, permiten comprender, interpretar y generar lenguaje humano de manera automatizada utilizando técnicas de aprendizaje profundo para analizar y generar texto en forma de palabras, frases o incluso párrafos completos, aunque existe la posibilidad de interpretar datos de voz e incluso imágenes [43].

Los *Transformers* destacan por su capacidad para modelar dependencias a largo plazo en secuencias de datos y realizar tareas complejas de NLP de manera eficiente y escalable. Se caracterizan principalmente por utilizar el mecanismo de atención multi-cabeza, que permite al modelo considerar interacciones entre todas las posiciones en una secuencia de entrada simultáneamente. Los modelos parten las entradas de texto en *tokens*, asignando un *token* a cada palabra o símbolo (como el punto final de frase). A su vez cada *token* es transformado a un vector numérico denominado *embedding* que permite desarrollar relaciones semánticas entre *tokens*. En la figura 2-7 podemos observar un ejemplo de *embedding* semántico, donde los vectores se componen en función de la edad y el género del *token* [44].

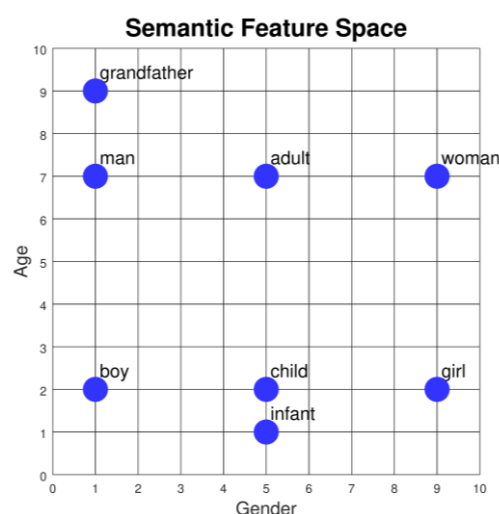


Figura 2-7 Ejemplo de *embedding* semántico [45].

Los *Transformers* representan una innovadora arquitectura en el campo del procesamiento del lenguaje natural, empleando mecanismos de atención ponderada para modelar las interacciones entre los elementos de una secuencia de entrada. A través de múltiples capas de atención y redes neuronales de alimentación hacia adelante (*feedforward*) los *Transformers* capturan de manera eficaz las relaciones contextuales, es decir, transportan los datos de entrada por capas de neuronas y calculan una salida sin necesidad de retroalimentación, permitiendo al modelo aprender representaciones profundas y jerárquicas del texto [44].

La aparición de la arquitectura de los *Transformers* y su popularización ha hecho posible el desarrollo de los denominados grandes modelos del lenguaje, potentes modelos pre-entrenados diseñados para trabajar con grandes volúmenes de datos y generar texto coherente y relevante. Durante su fase de pre-entrenamiento, estos modelos absorben grandes cantidades de texto sin etiquetar, lo que les permite internalizar las estructuras y patrones del lenguaje humano. Los LLMs pueden desempeñar diversos roles como, por ejemplo:

- Generación de Texto (ej. Ai Dungeon [46]).
- Resumen de Texto (ej. Otter.ai [47]).
- Traducción de Idiomas (ej. Google Translate [48]).
- Clasificación de Texto (ej. MonkeyLearn [49]).
- Chatbots (ej. GPT-4 [50]).

A pesar de las capacidades de aprendizaje de los LLMs genéricos, los desarrolladores y las empresas pueden personalizar estos sistemas para adaptarlos a sus necesidades específicas mediante técnicas como el ajuste fino (*fine tuning*) [51].

2.2.5 Llama 2.

Desde sus inicios con modelos relativamente simples como Word2Vec [52] y GloVe [53], hasta la llegada de arquitecturas más sofisticadas como GPT-3, hemos sido testigos de una notable evolución en los grandes modelos del lenguaje. Estos modelos han pasado de simplemente asignar vectores a palabras a capturar relaciones complejas y contextuales en el lenguaje humano. La introducción de modelos pre-entrenados masivos, como BERT [54], Gemini [55] o Llama [56], ha revolucionado radicalmente el campo del procesamiento del lenguaje natural, produciendo grandes avances en áreas como la generación de texto, la traducción automática y la comprensión del lenguaje. La mejora continua en la capacidad de estos modelos para entender y generar lenguaje natural ha despertado un creciente interés y aplicación en diversas industrias, desde la atención médica hasta la educación y el servicio al cliente y esto ha transformado el desarrollo de los LLMs en una intensa competición entre las principales empresas de tecnología que buscan impulsar aún más los límites de la inteligencia artificial y su aplicación práctica en el mundo real [57].

Un ejemplo notable de avance en este campo es el caso de Meta, que presentó en febrero de 2023 su propio LLM denominado Llama, y lanzó su actualización a Llama 2 en julio del mismo año. En su primera versión, se entrenaron cuatro tamaños de modelo: 7, 13, 33 y 65 mil millones de parámetros. Según anunciaban, el rendimiento del modelo de 13 mil millones de parámetros en tareas típicas de NLP superaba en la mayoría de los casos a modelos mucho más grandes como GPT-3 (con 175 mil millones de parámetros) y que el modelo más grande era competitivo con modelos de vanguardia como PaLM y Chinchilla [57]. Otro punto destacado de Llama es que, mientras que la mayoría de los LLMs más potentes del mercado generalmente solo han sido accesibles a través de APIs limitadas, Meta lanzó el modelo de Llama a la comunidad de investigación bajo una licencia de uso libre [58].

A día de hoy, Llama-2 incluye tanto modelos fundamentales como modelos ajustados para diálogos y generación de código, llamados Llama-2 Chat y Llama 2-Code. Además, podemos encontrar tres tamaños distintos del modelo: 7, 13 y 70 mil millones de parámetros. La arquitectura permanece en gran medida sin cambios respecto a la del modelo anterior, pero se utilizó un 40% más de datos para entrenar

los modelos fundamentales que en el *dataset* de Llama 1 (véase figura 2-8). Meta menciona también un modelo con 34 mil millones de parámetros que podría ser lanzado en el futuro una vez que se cumplan los objetivos de seguridad.

Dataset	Sampling prop.	Epochs	Disk size
CommonCrawl	67.0%	1.10	3.3 TB
C4	15.0%	1.06	783 GB
Github	4.5%	0.64	328 GB
Wikipedia	4.5%	2.45	83 GB
Books	4.5%	2.23	85 GB
ArXiv	2.5%	1.06	92 GB
StackExchange	2.0%	1.03	78 GB

Figura 2-8 Dataset de Llama [57].

2.3 Social Media.

2.3.1 Introducción e historia.

En la era digital contemporánea, el fenómeno del *social media* ha transformado radicalmente la manera en que las personas se comunican, interactúan y consumen información. Las plataformas de *social media* han pasado de ser simples herramientas de conexión a complejos ecosistemas que influyen en la opinión pública, la cultura y hasta en la política, llegando a tener impacto incluso en el transcurso de conflictos bélicos como la guerra entre Rusia y Ucrania [59]. Es tal la facilidad de compartir información que se empieza a mencionar el término de ciudadano periodista, refiriéndose a que cualquier persona con un dispositivo móvil conectado puede reportar hechos en distintos escenarios [60].

La historia de las redes sociales se remonta al comienzo de Internet, cuando la conectividad global comenzó a transformar radicalmente la forma en que las personas se comunicaban y compartían información. Aunque los orígenes exactos son difíciles de precisar, los primeros vestigios de lo que ahora conocemos como *social media* pueden encontrarse en las formas primitivas de comunicación en línea, como los tableros de anuncios y los grupos de discusión en la década de 1980. Sin embargo, no fue hasta el advenimiento de la web 2.0 a principios del siglo XXI cuando el *social media* realmente despegó. La web 2.0 permitió la interactividad y la participación de los usuarios, sentando las bases para el surgimiento de sitios web centrados en la comunicación social. Estas plataformas ofrecían a los usuarios la posibilidad de crear perfiles, conectarse con otros usuarios y compartir contenido de diversas formas [61].

Una de las primeras plataformas de *social media* que capturó la imaginación del público fue Six Degrees [62], lanzada en 1997. Permitía a los usuarios crear perfiles y conectarse con otros usuarios, sentando así las bases para las redes sociales modernas. Sin embargo, fue en la década de 2000 cuando el *social media* experimentó un crecimiento explosivo con el lanzamiento de sitios como Friendster, MySpace y LinkedIn, cada uno dirigido a diferentes aspectos de la interacción social en línea. Sin embargo, fue Facebook, fundada en 2004 por Mark Zuckerberg, la que realmente revolucionó el panorama del *social media*. Facebook rápidamente se convirtió en la red social más grande del mundo, permitiendo a los usuarios compartir actualizaciones de estado, fotos, vídeos y mensajes con amigos y familiares. Su éxito inspiró una ola de nuevas plataformas de *social media*, incluidas X/Twitter, YouTube, Instagram y Snapchat (véase figura 2-9), cada una con su propia propuesta única y enfoque en la comunicación en línea [61].

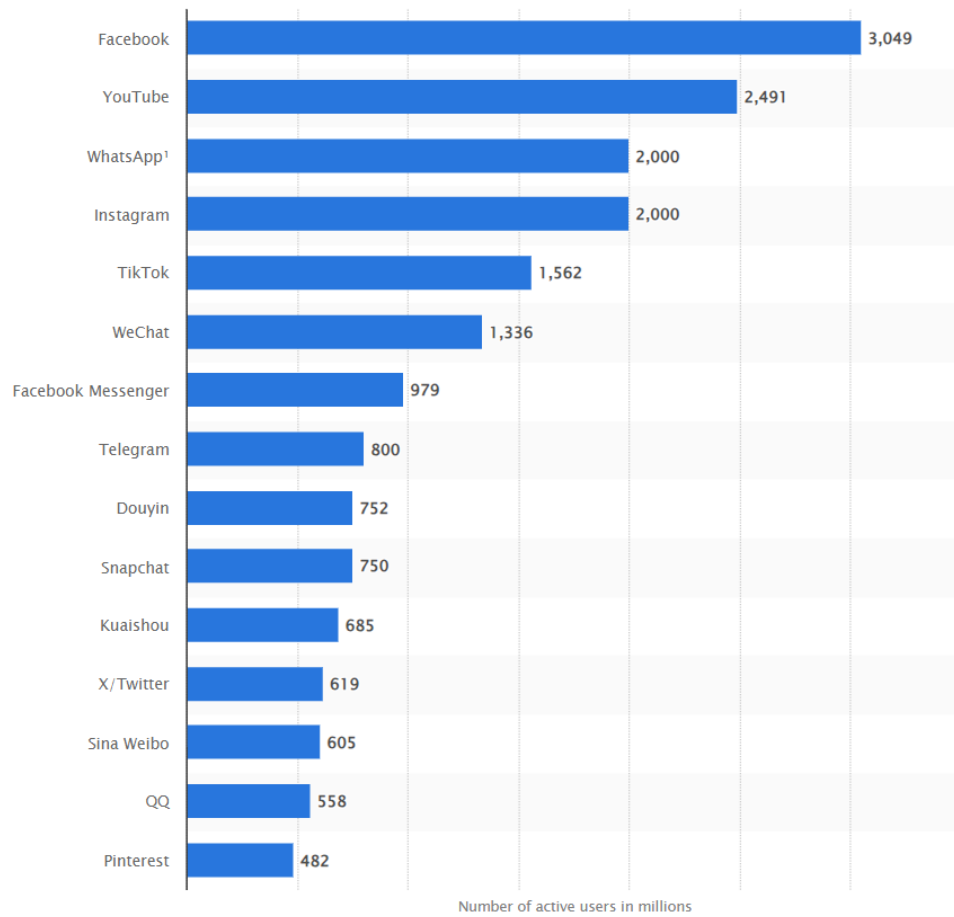


Figura 2-9 Distribución de plataformas de *social media* en función del número de usuarios [30].

A lo largo de los años, el *social media* ha evolucionado constantemente, adaptándose a los cambios tecnológicos y a las preferencias de los usuarios. Ha democratizado la comunicación, permitiendo que las voces antes marginadas tengan una plataforma para ser escuchadas, pero también ha generado preocupaciones sobre la privacidad, la veracidad de la información y el impacto en la salud mental [61].

En las próximas secciones, se presentarán algunas de las plataformas y redes sociales que se han empleado durante el desarrollo del presente trabajo, principalmente como fuentes de información.

2.3.2 X/Twitter.

La plataforma Twitter, actualmente denominada X [63], fue implementada por Jack Dorsey, Biz Stone y Evan Williams en julio de 2006. Su formato distintivo de *microblogging* proporcionaba un sistema para compartir pensamientos y actualizaciones de estado de forma rápida y concisa en un máximo de 140 caracteres. Inicialmente, Twitter se utilizaba para comunicaciones personales y sociales, pero rápidamente ganó popularidad en el ámbito profesional y de los medios de comunicación. Durante sus primeros años, Twitter experimentó un crecimiento significativo de usuarios y se convirtió en un medio fundamental para la cobertura en tiempo real de eventos importantes, desde desastres naturales hasta protestas políticas. Su papel en la difusión de información durante eventos como la Primavera Árabe en 2011 [64] y las elecciones presidenciales de Estados Unidos en 2016 destacó su importancia como una herramienta de comunicación global [65].

A lo largo de los años, Twitter ha introducido una serie de características nuevas, como la capacidad de incluir imágenes, vídeos y enlaces en los tuits (véase figura 2-10), así como la ampliación del límite

de caracteres a 280 en 2017 y finalmente la ampliación a 4000 caracteres para los suscriptores a su servicio de pago Blue en junio de 2021 [66]. Estos cambios han ampliado la funcionalidad de Twitter y han permitido una mayor expresión y creatividad por parte de los usuarios.

En la actualidad, X es una plataforma diversa que abarca una amplia gama de usos y usuarios. Además de las interacciones personales y sociales, X se utiliza ampliamente en el ámbito profesional para la promoción de marcas, la generación de *leads* (proceso de *marketing* digital para la atracción de clientes) y el servicio al cliente. Los medios de comunicación utilizan X como una herramienta para la distribución de noticias en tiempo real y de participación de la audiencia. Los líderes políticos y activistas aprovechan Twitter para comunicarse directamente con el público y movilizar apoyo para diversas causas [65].

Aunque X es una de las redes sociales más aclamadas, también ha sido objeto de críticas y controversias, especialmente en lo que respecta a la moderación del contenido, la gestión de la desinformación y la cantidad de cuentas automatizadas (denominadas *bots*) existentes. La plataforma ha implementado una serie de medidas para abordar estos problemas, incluyendo la introducción de estrictas políticas de comportamiento de los usuarios y la verificación de cuentas [67].



Figura 2-10 Captura de usuario de X (Autoría propia).

2.3.3 Instagram.

Instagram [68], lanzado en 2010 por Kevin Systrom y Mike Krieger, revolucionó la forma en que las personas comparten imágenes y vídeos en línea. A diferencia de otras plataformas de *social media*, Instagram se centra en la estética visual y la narración de historias a través de imágenes. Con funciones como filtros y *stories*, la influencia de Instagram en la cultura contemporánea es innegable, tanto ha sido así que la plataforma ha dado lugar a una nueva generación de celebridades e *influencers* (véase figura 2-11) que han construido redes lucrativas a través de la promoción de productos y la colaboración con marcas. Sin embargo, también ha sido objeto de críticas por fomentar una cultura de comparación y superficialidad, con el énfasis en la imagen corporal y el estilo de vida aspiracional [68].

Instagram ha experimentado un crecimiento exponencial desde su adquisición por parte de Facebook en 2012, con la introducción de nuevas características como IGTV (*Instagram television*), que permite a los usuarios compartir vídeos de larga duración, e Instagram Shopping, que facilita la compra de productos directamente desde la plataforma. Estas innovaciones han consolidado aún más la posición de Instagram como una de las principales plataformas de *social media* del mundo [68].

A pesar de ser ampliamente elogiado por su rápido ascenso y su enorme popularidad, Instagram ha sido objeto de críticas debido a problemas comunes en las plataformas de redes sociales: contenido inapropiado, desinformación y falta de moderación. Estudios también han demostrado que puede tener un efecto perjudicial en la salud mental de los adolescentes, especialmente afectando a la autoestima de las jóvenes según Britannica [68].

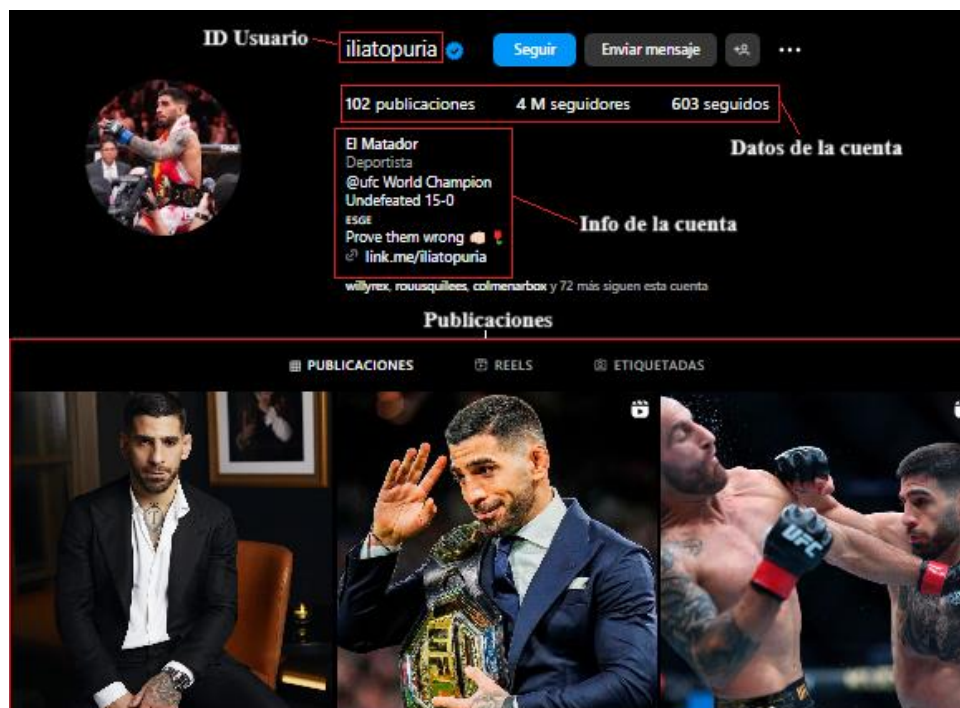


Figura 2-11 Captura de usuario de Instagram (Autoría propia).

2.3.4 YouTube.

Desde su fundación en 2005 por Chad Hurley, Steve Chen y Jawed Karim, YouTube [69] ha crecido exponencialmente, pasando de ser una plataforma para compartir vídeos caseros a convertirse en un gigante de los medios de comunicación. Su modelo de negocio basado en la publicidad ha sido una fuerza impulsora en esta expansión, proporcionando una vía para que los creadores obtengan ingresos y desarrollen comunidades de seguidores leales. Sin embargo, a medida que YouTube ha crecido, también

ha enfrentado críticas significativas. La plataforma ha sido objeto de controversia por su papel en la proliferación de desinformación y teorías de la conspiración. Además, existe la preocupación de que los algoritmos de recomendación de YouTube puedan fomentar cámaras de eco y polarización al dirigir a los usuarios hacia contenido cada vez más extremo y sesgado [70]. En otro ámbito, algunos creadores de contenido han expresado preocupaciones sobre la equidad y la transparencia en las políticas de monetización de YouTube provocando migraciones de creadores de contenido y usuarios a otras plataformas [71].

A pesar de estos desafíos, YouTube ha demostrado una capacidad notable para adaptarse y evolucionar. Servicios como YouTube Premium y YouTube Music suponen un intento de diversificación para satisfacer las demandas cambiantes de los usuarios. Además, la plataforma ha comenzado recientemente la producción de contenido original para competir directamente con otros servicios de *streaming* como Netflix [72] y Hulu [73].

Por otro lado, YouTube ha demostrado ser una herramienta valiosa para la educación y el aprendizaje en línea. La plataforma ha facilitado el acceso a una amplia variedad de contenido educativo, desde tutoriales sobre habilidades prácticas (véase figura 2-12) hasta conferencias de expertos en campos académicos. Esto ha democratizado el conocimiento y ha brindado oportunidades de aprendizaje a personas de todo el mundo, independientemente de su ubicación o situación socioeconómica.

En el campo de la economía digital, YouTube ha sido un motor realmente importante, proporcionando oportunidades de empleo y emprendimiento para una amplia gama de personas. Los creadores de contenido pueden generar ingresos a través de programas de asociación de YouTube, patrocinios y ventas de productos relacionados con sus vídeos. Esto ha creado una economía vibrante en torno a la creación de contenido en línea y ha permitido que individuos talentosos encuentren éxito y reconocimiento en la plataforma.

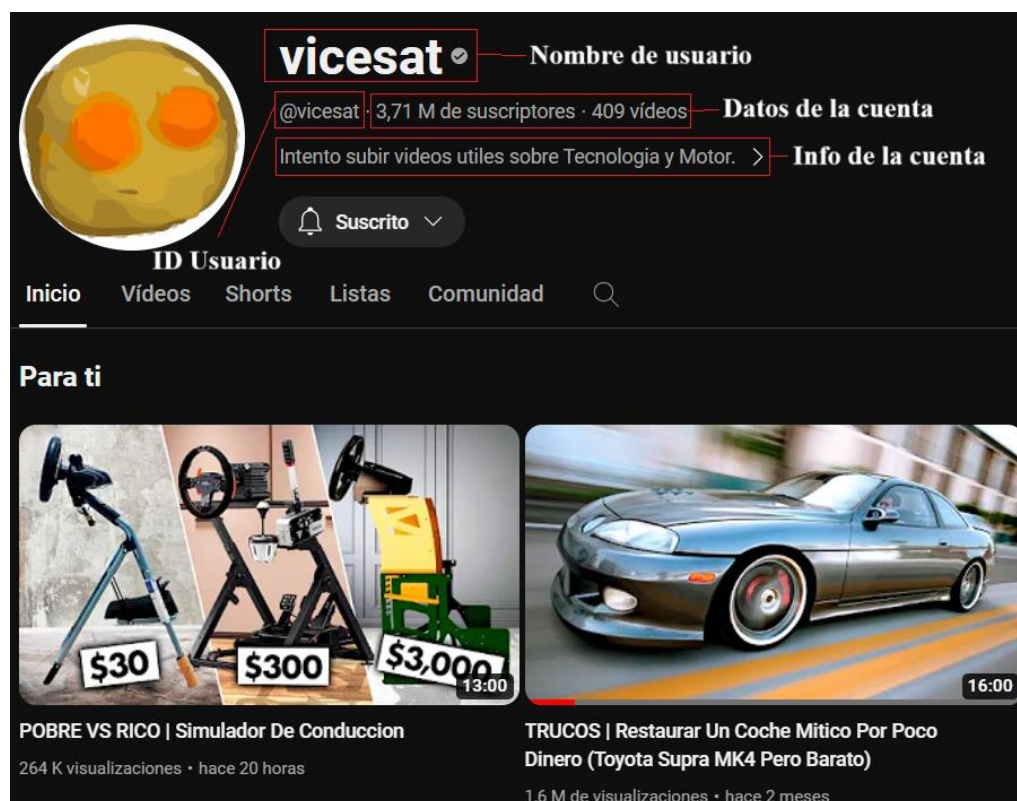


Figura 2-12 Captura de usuario de YouTube (Autoría propia).

2.3.5 Telegram.

En 2006, Pavel Durov fundó VKontakte, o VK, una red social a menudo comparada con Facebook, pero de origen ruso. Su rápido crecimiento, superando los 100 millones de usuarios a principios de la década de 2010, atrajo la atención del gobierno ruso. En 2011, el gobierno solicitó a la empresa que censurara contenido relacionado con protestas por las elecciones parlamentarias, a lo que Durov respondió negativamente. Esto desencadenó una campaña de difamación pública, presiones para la venta de VK e incluso una intimidante visita policial a su domicilio. Ante esta conflictiva situación, Durov vendió sus acciones restantes y abandonó Rusia. Tras estos hechos, en 2013, Pavel y Nikolai Durov fundaron la plataforma Telegram [74] como un medio de comunicación libre de interferencias gubernamentales, atrayendo a usuarios afines a su postura anti censura [75].

Telegram es una aplicación de mensajería online. La plataforma ofrece diversas funcionalidades como intercambio de mensajes de texto, llamadas de voz, compartición de archivos, la posibilidad de unirse a grupos de hasta 200,000 miembros y suscripción a canales de transmisión pública. El servicio es accesible a través de smartphones y computadoras personales, manteniendo la sincronización de conversaciones entre dispositivos [75].

En el panorama de la mensajería, Telegram se destaca como competidor de WhatsApp, el cual ha enfrentado controversias por compartir datos de usuarios con su empresa matriz, Meta [76]. Sin embargo, Telegram se diferencia al ofrecer chats “secretos” con cifrado de extremo a extremo, asegurando que solo los participantes del chat puedan acceder a los mensajes. Aunque las videollamadas de Telegram también cuentan con este nivel de seguridad, es importante destacar que los chats no marcados como “secretos” no están protegidos de la misma manera [75].

Es destacable que las características de privacidad de Telegram han atraído desde el principio a diversos grupos extremistas o de fines poco éticos. Un ejemplo de ello es la detección de canales y redes del ISIS operando en la plataforma en el año 2015. Aunque Pavel Durov anunció que se bloquearían todos los canales relacionados con el Estado Islámico otros canales extremistas continuaron operando sin restricciones. En 2023, la revista The Atlantic llamó a Telegram “la aplicación más importante del mundo (por ahora)” debido a su importante papel tanto en la insurrección liderada por Prigozhin contra Rusia como en el transcurso de la guerra ruso-ucraniana [77].

2.3.6 API de Telegram.

La API de Telegram es una interfaz de programación de aplicaciones que permite a los desarrolladores interactuar con dicha plataforma. Esta API está diseñada para proporcionar acceso a las funciones y capacidades de Telegram, permitiendo a los desarrolladores la creación de aplicaciones personalizadas, bots, servicios de envío y recepción de mensajería o la gestión de chats y grupos.

La API de Telegram se basa en el protocolo MTProto (véase figura 2-13), que es un protocolo de comunicación propio de Telegram diseñado para proporcionar un alto nivel de seguridad y eficiencia en la transmisión de datos. Este protocolo utiliza cifrado simétrico de 256-bit AES, cifrado RSA 2048 e intercambio de claves seguras mediante el protocolo Diffie-Hellman simétrico [74] para garantizar la seguridad y la privacidad de las comunicaciones entre los clientes de Telegram y los servidores de la plataforma.

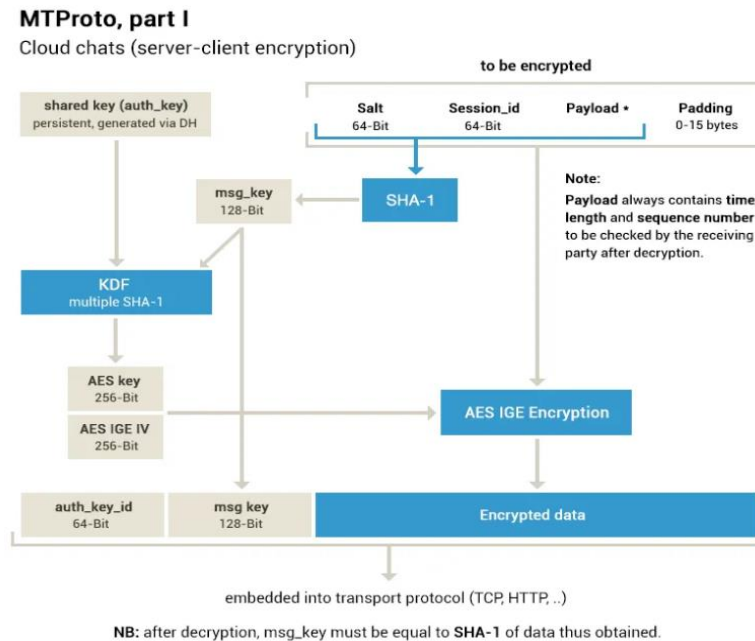


Figura 2-13 Protocolo MTPROTO [78].

Entre las principales características de la API de Telegram encontramos [79]:

- Mensajería: la API permite enviar y recibir mensajes de texto, multimedia y otros tipos de contenido a través de Telegram. Los desarrolladores pueden utilizar métodos específicos para enviar mensajes individuales, mensajes en grupos, mensajes multimedia, etc.
- Gestión de chats y grupos: se proporcionan métodos para crear, editar y eliminar chats y grupos, así como para agregar y eliminar usuarios, establecer permisos y configurar otras propiedades de los chats y grupos.
- Bots de Telegram: una de las características más poderosas de la API de Telegram son los bots, que son cuentas especiales controladas por programas de software en lugar de personas. Los desarrolladores pueden crear bots utilizando la API de Telegram y configurarlos para responder a comandos, interactuar con usuarios y realizar diversas acciones. Esta es la principal funcionalidad a emplear en el desarrollo de este TFG.
- Notificaciones *push*: Telegram permite configurar notificaciones *push* para recibir alertas en tiempo real sobre nuevos mensajes, eventos y otras actividades dentro de la plataforma.
- Integración con servicios externos: además, la API de Telegram puede integrarse con otros servicios y plataformas, lo que permite a los desarrolladores conectar Telegram con sus propias aplicaciones, sistemas de *backend* y servicios en la nube para ampliar su funcionalidad y utilidad.

2.4 Trabajos relacionados.

2.4.1 Sistema de detección inteligente de usuarios de Twitter interesados en la temática de defensa.

Trabajo Fin de Grado (TFG) realizado por María Fernández Torrejón (2023) en el Centro Universitario de la Defensa de la Escuela Naval Militar [80]. En este documento se estudia la realización de una herramienta de software para la detección automatizada de perfiles de usuarios en Twitter que compartan contenido relacionado con el ámbito de la defensa. Para ello, se emplean técnicas avanzadas de procesamiento de lenguaje natural y aprendizaje automático para entrenar un clasificador capaz de discernir entre contenidos relevantes y no relevantes para este contexto. La construcción y validación del clasificador se lleva a cabo utilizando un corpus de datos previamente seleccionado y se comparan los resultados obtenidos con modelos de inteligencia artificial de gran escala, como el chatbot de ChatGPT, para evaluar su rendimiento relativo.

2.4.2 Herramientas de OSINT para la obtención de información en Twitter sobre el conflicto Ucrania-Rusia.

Trabajo Fin de Grado (TFG) realizado por Álvaro Hernández Ruiz (2022) en el Centro Universitario de la Defensa de la Escuela Naval Militar [81]. El transcurso de este trabajo de fin de grado se desarrolla en tres fases principales. En primer lugar, se hace un estudio sobre las diferentes herramientas y técnicas de explotación de inteligencia de fuentes abiertas. Posteriormente se selecciona una herramienta en función de su utilidad y características y se le realizan una serie de pruebas para mostrar su funcionamiento. En última fase, se hace un análisis de esta herramienta en un contexto de posible interés en el ámbito de la Defensa. En este caso en el conflicto ruso-ucraniano.

2.4.3 Plataforma OSINT para el almacenamiento y análisis inteligente de la red social Twitter.

Trabajo Fin de Grado (TFG) realizado por Javier Jiménez-Cervantes Pérez (2022) en el Centro Universitario de la Defensa de la Escuela Naval Militar [82]. Este documento detalla la construcción de una plataforma de inteligencia de código abierto destinada a almacenar, procesar y analizar datos extraídos de la plataforma Twitter para obtener Información de Fuentes Abiertas (OSINT). Esta herramienta abarca desde la recolección inicial de datos hasta la visualización de la información procesada, con funcionalidades avanzadas como la segmentación y la detección de patrones. Se ilustra su aplicación en el análisis de opiniones públicas y la identificación de figuras influyentes. Desarrollada en Python y basada en herramientas de procesamiento de datos, esta plataforma de código abierto permite la adaptación a las necesidades específicas de los usuarios. En resumen, ofrece una solución de inteligencia de acceso abierto para obtener información actualizada sobre temas de interés público, con posibles repercusiones significativas en la investigación y la sociedad al facilitar el análisis eficiente de datos sociales a gran escala.

2.4.4 Evaluación de los chatbot LLM para la concienciación de Amenazas Cibernéticas basada en OSINT.

Este artículo académico aborda la relevancia del intercambio de conocimientos sobre amenazas emergentes en el ámbito de la ciberseguridad, destacando su papel fundamental en la construcción de la inteligencia de amenazas cibernéticas (*cyber threat awareness*, CTI). En este documento se examina el impacto de LLM en el campo de la ciberseguridad, y se presenta un estudio que evalúa el rendimiento de diversos *chatbots*, incluyendo ChatGPT, GPT4all, Dolly, Stanford Alpaca, Alpaca-LoRA, Falcon y

Vicuna, en tareas de clasificación binaria y reconocimiento de entidades nombradas (Named Entity Recognition, NER) como personas, organizaciones y fechas utilizando inteligencia de fuentes abiertas. La evaluación se basa en datos previamente recopilados de X/Twitter, y se emplea para medir la competitividad de estos *chatbots* frente a modelos especializados entrenados específicamente para dichas tareas [83].

2.4.5 ChatGPT como oficial de inteligencia.

Este artículo académico examina cómo los LLM, especialmente ChatGPT, pueden ser utilizados para mejorar las actividades de los oficiales de inteligencia. Los LLMs son herramientas valiosas debido a sus impresionantes habilidades en el procesamiento y generación de lenguaje natural, lo que los convierte en recursos esenciales para tareas como la recuperación, análisis y toma de decisiones basadas en información. Se ofrece una visión general inicial sobre cómo los LLMs pueden ser aplicados en operaciones de inteligencia. A través de un estudio de caso centrado en ChatGPT, se ejemplifica cómo los oficiales de inteligencia pueden aprovechar estas herramientas para mejorar diferentes aspectos de su trabajo, desde la recolección y fusión de datos hasta el análisis, la elaboración de informes y el intercambio de información. Los resultados resaltan tanto el amplio potencial como las limitaciones que los LLMs tienen en su apoyo a los oficiales de inteligencia en tareas complejas y exigentes [84].

3 DESARROLLO DEL TFG

Con el fin de implementar el asistente han de solucionarse una serie de cuestiones y problemas individuales (véase figura 3-1) y para ello se deben desarrollar una serie de códigos que permitan ejecutar cada función de la herramienta. Con el fin de esquematizar el proceso de manera ordenada y coherente se dividirá en tres componentes que posteriormente combinados constituirán la herramienta objetivo (véase figura 3-2):

- Interfaz de usuario: utilizando la API de Telegram, se realizará un bot con diversos comandos predeterminados que permitan interactuar con la herramienta a través del chat de la plataforma. En primer lugar, se deberá realizar una selección de la red social que se quiere analizar (YouTube, Instagram o documentos de texto) e introducir unos parámetros de entrada para que la herramienta extraiga los datos y realice el proceso de análisis. Esta interfaz debe ser clara y sencilla para facilitar su comprensión al usuario.
- Proceso de captura de datos: inicialmente, se solicitará al usuario introducir el nombre de del perfil a analizar a través de una entrada en el chat. Posteriormente, la herramienta automatizará la obtención de las primeras N publicaciones de dicho perfil en la plataforma previamente seleccionada, las cuales serán almacenadas en un fichero para su posterior análisis. En la plataforma de YouTube, la herramienta obtendrá texto mediante la extracción de subtítulos del vídeo. En el caso de Instagram, la herramienta obtendrá los pies de foto de las publicaciones del perfil. Además, se añadirá la posibilidad de introducir cualquier documento en formato “.txt” para ampliar las fuentes de análisis.
- Proceso de análisis de datos: se introducirá el fichero resultado del proceso anterior en el LLM ajustado mediante un corpus de artículos de defensa y este determinará si se trata o no de una cuenta relacionada con el ámbito de la defensa, respondiendo al usuario de manera clara y concisa.

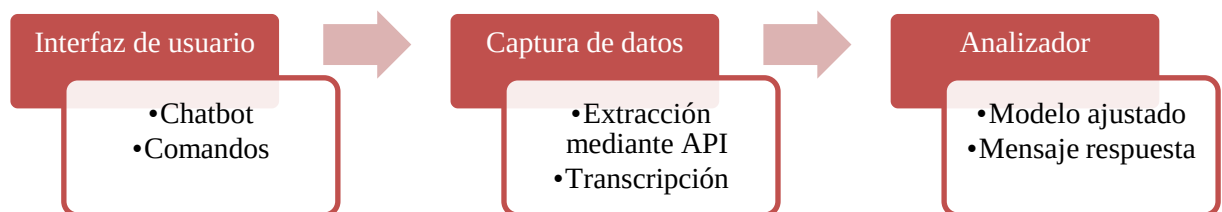


Figura 3-1 Principales módulos de la herramienta (Autoría propia).

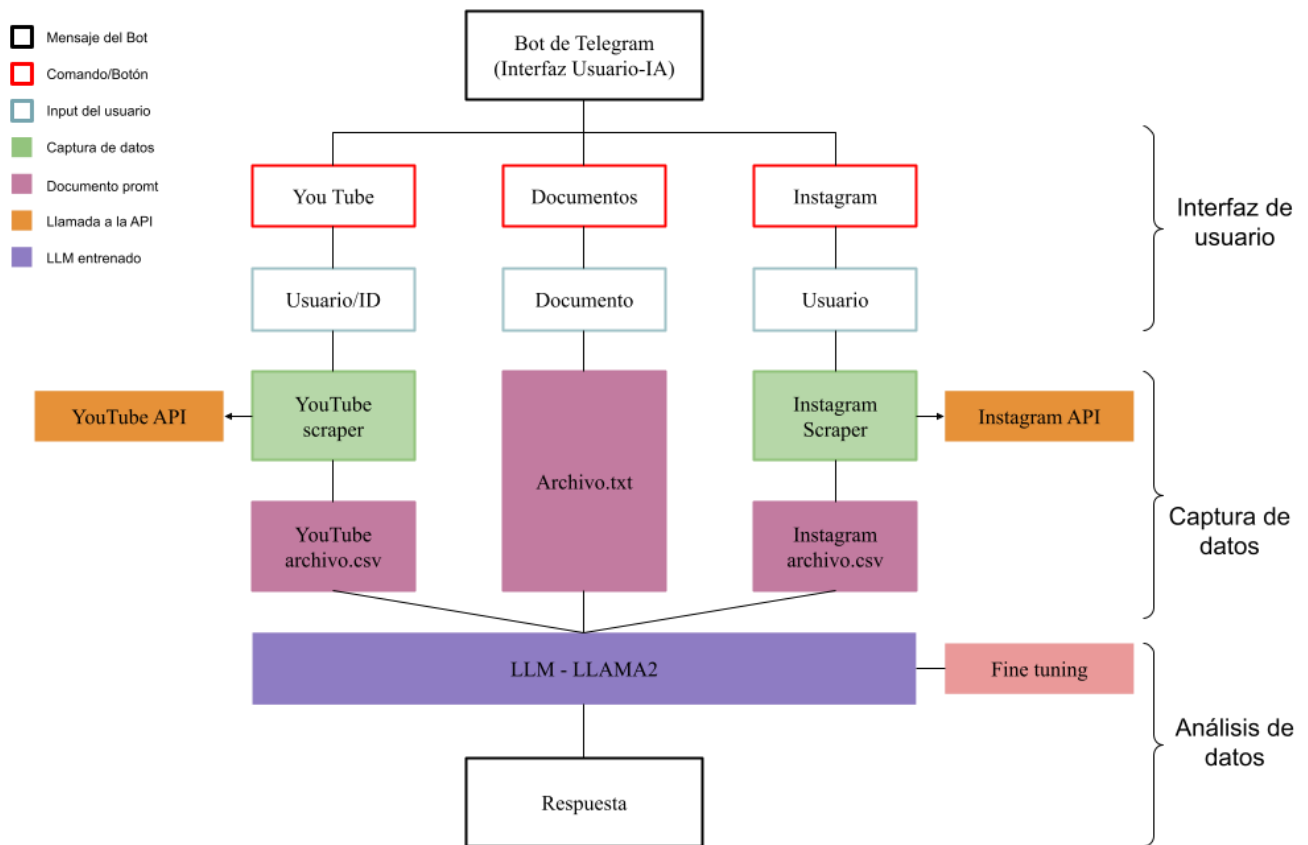


Figura 3-2 Arquitectura de la herramienta (Autoría propia).

3.1 Entorno de programación.

Dada la considerable demanda de recursos para el procesamiento de los *Large Language Models* y los procedimientos de *fine tuning*, la herramienta deberá desarrollarse en un entorno adecuado el cual permita realizar un trabajo rápido y eficaz. Para ello, se ha seleccionado Google Colab [85], el entorno de programación online de Google LLC, como entorno de trabajo debido al acceso gratuito a GPUs y TPUs de alto rendimiento que proporciona.

Para empezar a programar en Colab, lo único necesario es tener un dispositivo con acceso a Internet y una cuenta de Google, por lo que realmente las características del hardware del dispositivo no adquieren tanta importancia a la hora de ejecutar los comandos.

El primer paso para elaborar un bloque de código sería abrir un nuevo cuaderno (véase figura 3-3). Los cuadernos de Colab nos permiten escribir líneas de código y texto, así como ejecutar el código y obtener resultados visuales del mismo (véase figura 3-4). Una vez este creado el cuaderno podremos empezar a escribir el código e ir ejecutándolo por bloques. Los cuadernos de Google Colab se guardan en formato “.ipynb”, que es el formato estándar de los cuadernos Jupyter, facilitando su compartición y edición en diversos entornos de desarrollo, como Jupyter Lab [86].

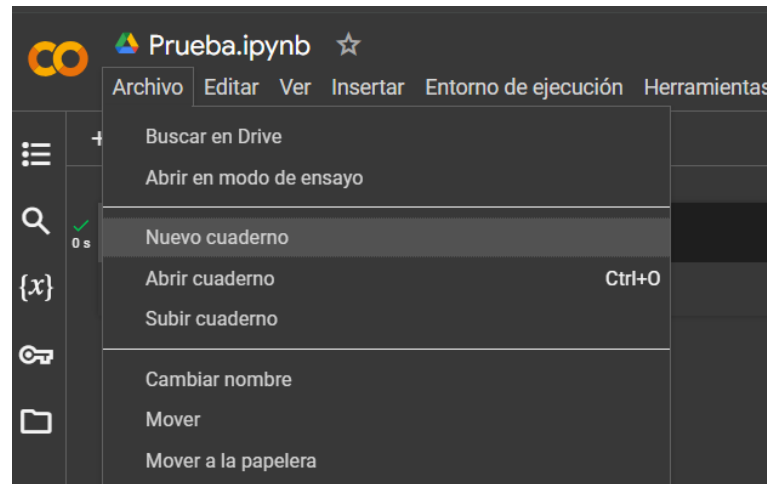


Figura 3-3 Captura de Colab. Abrir nuevo cuaderno (Autoría propia).

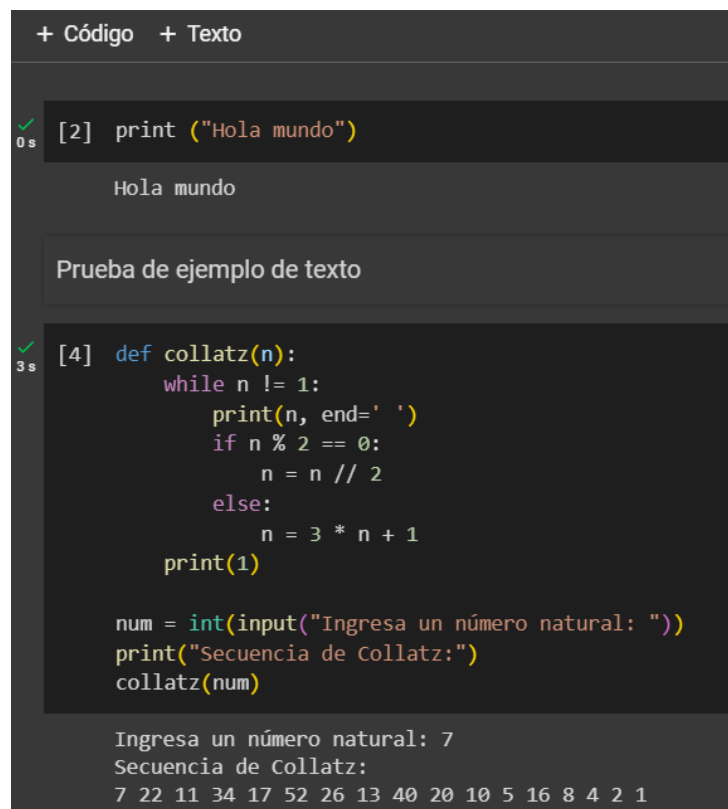


Figura 3-4 Captura de Colab. Ejemplo de ejecución de bloque de código y texto (Autoría propia).

Aunque la aplicación permite el uso de otros lenguajes, el lenguaje predeterminado de Google Colab es Python, versión 3 [87]. En los últimos 10 años, este lenguaje de programación de alto nivel ha ganado popularidad en una variedad de campos, incluyendo el desarrollo web, la ciencia de datos, la investigación académica y la educación.

Una de las ventajas más destacadas de Python es su sintaxis simple y legible, que hace que sea fácil de aprender y entender incluso para aquellos que tienen poca experiencia en programación. Además, Python cuenta con una vasta biblioteca estándar y una gran cantidad de bibliotecas de terceros especializadas en una variedad de áreas. Esto proporciona acceso a herramientas para realizar análisis de datos, modelado y simulación, sin tener que escribir todo desde cero. Por otro lado, Python es de

código abierto y gratuito, lo que significa que se pueden acceder a una amplia gama de recursos educativos, tutoriales y plataformas de desarrollo gratuitas, como Jupyter Notebook, Google Colab y Anaconda [88], para crear y compartir contenido educativo, realizar investigaciones y colaborar en proyectos [89].

Sin embargo, aunque Python es conocido por su simplicidad y facilidad de uso, no es el lenguaje más rápido en términos de rendimiento. Esto puede ser un inconveniente en aplicaciones que requieren un procesamiento intensivo de datos o cálculos computacionales muy rápidos, como simulaciones físicas o análisis de grandes conjuntos de datos en tiempo real. Adicionalmente, puede no ser la mejor opción en casos donde se requiera programación de sistemas o desarrollo de software de baja latencia, ya que otros lenguajes como C++ tienen mayor control sobre los recursos del sistema y su rendimiento [89].

Cabe destacar también, que Colab permite configurar el entorno de ejecución permitiendo seleccionar CPU, TPU y distintas GPU en función de la suscripción del usuario (véase figura 3-5).

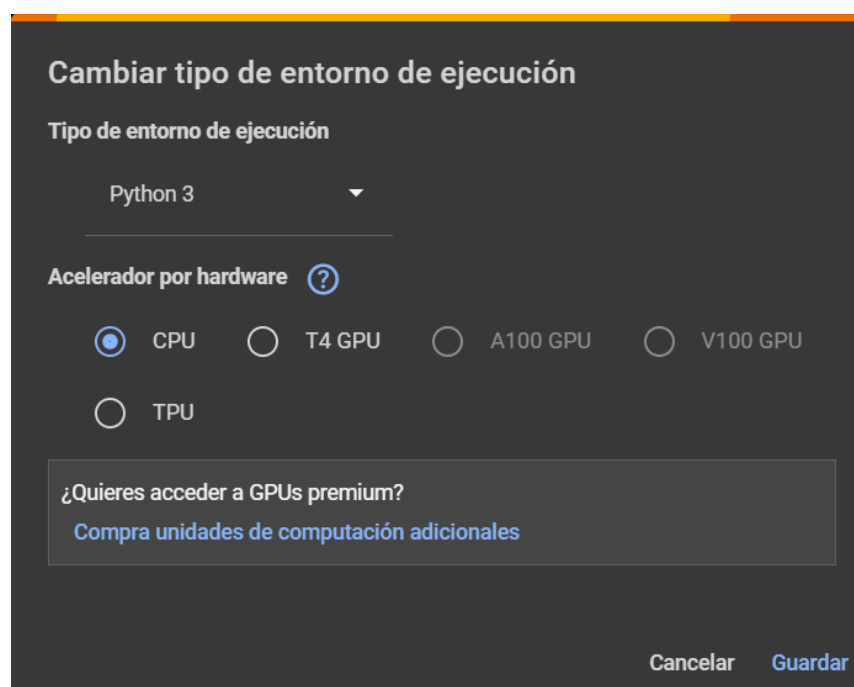


Figura 3-5 Captura de Colab. Configuración del entorno (Autoría propia).

3.2 Interfaz de usuario.

Siguiendo la estructura de módulos reflejada en la figura 3-1, en los siguientes apartados estudiaremos la creación de una interfaz que permita la interacción entre usuario y máquina.

3.2.1 Creación del bot.

Para la creación del bot de Telegram lo primero será instalar la plataforma. Telegram se encuentra disponible para Android, iOS, Windows, Windows Phone y Linux. Por lo que podremos descargarlo sencillamente desde las tiendas de aplicaciones del dispositivo móvil o simplemente utilizarlo desde el navegador del ordenador.

El siguiente paso será registrarse en la aplicación, para ello se requiere conexión a Internet y un número de teléfono funcional (véase figura 3-6). Una vez seguidos los pasos de configuración de la aplicación ya podríamos empezar a disfrutar de su uso.

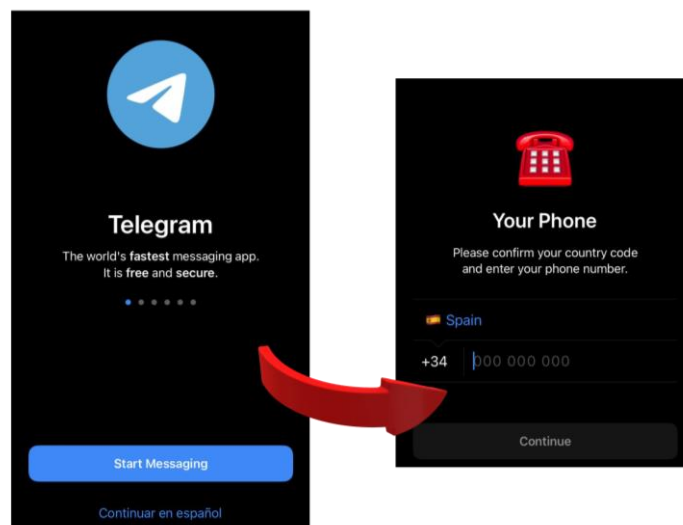


Figura 3-6 Configuración de Telegram (Autoría propia).

A partir de este momento, contamos con varias opciones para crear un bot de Telegram. Podemos optar por programarlo directamente o utilizar la API desde la aplicación mediante BotFather, una herramienta de interfaz para la configuración de bots disponible en la plataforma. Para ello, simplemente debemos introducir el nombre “BotFather” en el buscador de chats y seleccionar el resultado correspondiente (véase figura 3-7).



Figura 3-7 Búsqueda de “BotFather” (Autoría propia).

Una vez que hayamos accedido a “BotFather”, activaremos la herramienta escribiendo el comando “/start” y automáticamente responderá con la lista de comandos funcionales de la herramienta (véase figura 3-8). Para crear un nuevo bot deberá seleccionarse el comando “/newbot.”. A partir de este punto, “BotFather” solicitará automáticamente la información necesaria para la creación del bot. En primer lugar, nos pedirá un nombre y un @username para el bot. Seguido de esto generará un token privado necesario para programar el bot, es decir, generará una contraseña única que permitirá enlazar nuestro bot con el código fuente que se le quiera asignar (véase figura 3-9). Y con esto ya dispondremos de todo lo necesario para empezar a programar.

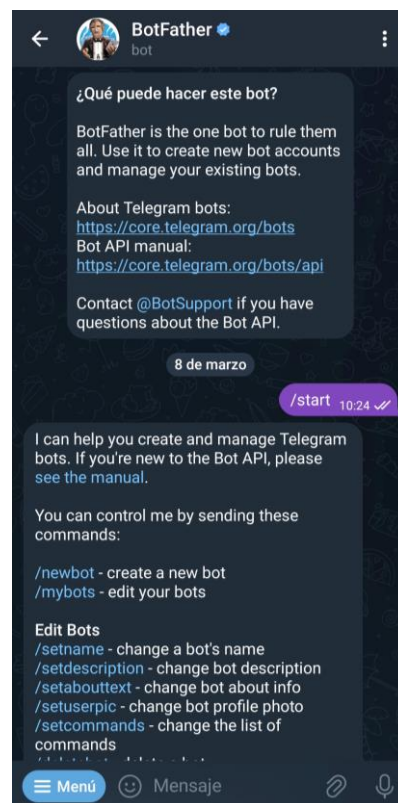


Figura 3-8 Inicio de “BotFather” (Autoría propia).

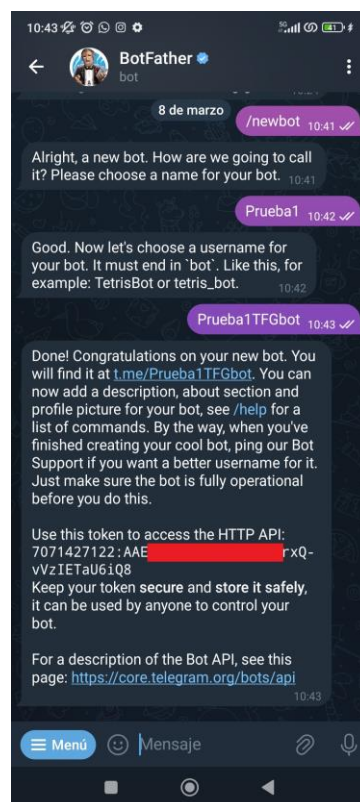


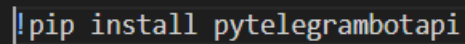
Figura 3-9 Obtención del token de “BotFather” (Autoría propia).

3.2.2 Conexión con la API de Telegram.

Una vez que se ha obtenido el *token*, la API de Telegram facilita la configuración del bot para que ejecute los procesos programados. Además, posibilita la configuración de una serie de comandos que comiencen con “/” para llevar a cabo acciones específicas en el bot. Para desarrollar la herramienta del proyecto implementaremos en nuestro bot los siguientes comandos:

- “/Instagram”: este comando extraerá y almacenará en un archivo de texto los pies de foto de las últimas publicaciones del usuario que se introduzca, para analizarlos posteriormente.
- “/YouTube”: este comando extraerá y almacenará en un archivo de texto los subtítulos de los últimos vídeos publicados por el usuario que se introduzca, para analizarlos posteriormente.
- “/Doc”: este comando recibirá documentos de texto enviados por el usuario para analizarlos posteriormente.
- “/start”: este comando envía el mensaje de bienvenida al chat y permite navegar entre los demás comandos.
- “/info”: este comando proporciona información general del uso y origen del bot.
- “/help”: este comando despliega la lista de comandos disponibles.

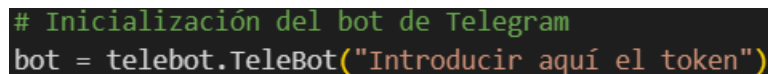
Para implementar los comandos y desarrollar los procesos de interacción entre el usuario y el bot, será necesario utilizar la librería *pytelegrambotapi* [90]. Esta librería proporciona una interfaz sencilla y completa para interactuar con la API de Telegram, facilitando la creación y gestión de bots. Permite recibir actualizaciones de mensajes entrantes, enviar mensajes, gestionar teclados personalizados, manejar archivos multimedia, entre otras funcionalidades, todo ello de una manera intuitiva y eficiente. Por lo que el primer paso es instalar la librería (véase figura 3-10).



```
!pip install pytelegrambotapi
```

Figura 3-10 Comando de instalación de *pytelegrambotapi* (Autoría propia).

Ahora se deberá inicializar el bot de Telegram utilizando el *token* que obtuvimos durante su creación (véase figura 3-11). Cada bot tendrá un *token* único que permita su configuración por lo que es muy importante que dicho *token* sea conocido solo por el desarrollador ya que cualquier otro usuario que disponga de él podría hacer cambios en la interfaz.



```
# Inicialización del bot de Telegram
bot = telebot.TeleBot("Introducir aquí el token")
```

Figura 3-11 Código de Python para conectarse al bot mediante un token (Autoría propia).

Desde este punto ya se tienen los requisitos para proceder con la configuración mediante código de todos los procesos que pretendemos implementar en nuestra herramienta, así como también de establecer los comandos principales. Con el propósito de optimizar la experiencia del usuario, podemos incorporar una interfaz compuesta por botones que representen los distintos comandos de proceso, lo cual contribuirá a su usabilidad y accesibilidad.

```

Definir el teclado personalizado
eclado_personalizado = types.ReplyKeyboardMarkup(resize_keyboard=True)
eclado_personalizado.row('/Instagram 📷', '/YouTube 📺')
eclado_personalizado.row('/Doc 📄', '/info ⓘ')

```

Figura 3-12 Código de Python para implementar un teclado de botones en Telegram (Autoría propia).

Como se ilustra en la figura 3-12, es posible configurar una serie de botones para iniciar los comandos. En este caso específico, se ha dispuesto una matriz de cuatro botones organizados dos a dos (véase figura 3-13).

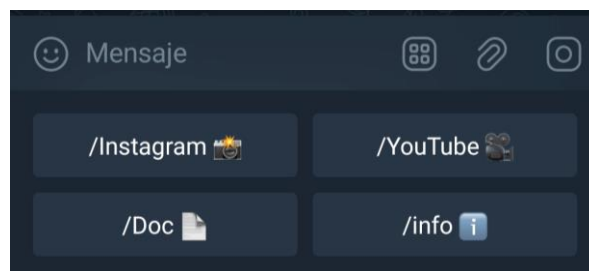


Figura 3-13 Captura de Telegram. Interfaz de botones (Autoría propia).

Para configurar las acciones que se ejecutan al introducir un comando utilizaremos manejadores de texto. Los manejadores de texto son una característica fundamental de la API de Telegram que nos permite definir acciones específicas que se activan cuando se introducen ciertos comandos en el chat del bot. Estos manejadores son esenciales para dar funcionalidad a nuestro bot y permitir que responda de manera inteligente a las interacciones de los usuarios. Cuando un usuario envía un mensaje que contiene un comando, como “/start” o “/help”, la API de Telegram procesa este mensaje y lo envía al bot. Luego, el bot utiliza los manejadores de texto para identificar qué comando se ha recibido y ejecutar la función correspondiente.

Para configurar los manejadores de texto en nuestro bot, primero debemos definir una función para cada comando que deseamos manejar y utilizar el método “@bot.message_handler” (véase figura 3-14). Por ejemplo, si queremos manejar el comando “/start”, creamos una función llamada `handle_start(message)` que se encargará de procesar ese comando específico. Dentro de esta función, podemos escribir el código necesario para responder al usuario de acuerdo con el comando recibido.

```

#Ejemplo de manejador de texto
@bot.message_handler(commands=['Ejemplo'])
def ejemplo(message):
    print("Esto es una función ejemplo")

```

Figura 3-14 Código de Python para implementar un manejador de comandos (Autoría propia).

Una vez que se han definido los manejadores para cada comando del bot de Telegram, podrían empezar a ser utilizados desde su interfaz. Esto significa que cuando un usuario envíe un mensaje que

contenga uno de los comandos configurados, el bot detectará automáticamente el comando y ejecutará la función asociada a ese comando específico. La interfaz de Telegram proporciona una forma conveniente para que los usuarios interactúen con nuestro bot. Pueden simplemente escribir el comando deseado en el chat y el bot responderá en consecuencia.

Sin embargo, en ciertos casos, es posible que queramos ejecutar una serie de funciones secuenciales como parte de un proceso más complejo. Para manejar esta situación, utilizamos el método `bot.register_next_step_handler`. Este método nos permite definir una función que será ejecutada después de que se haya completado una acción específica, por ejemplo, recibir un mensaje o procesar un comando (véase figura 3-15). Esto se debe a que los manejadores de texto pueden presentar problemas de anidamiento cuando se intenta utilizar múltiples funciones en un mismo proceso [91]. Esto puede ocasionar comportamientos inesperados o errores en la ejecución del código. Por ello, es recomendable utilizar el método `bot.register_next_step_handler` para coordinar la ejecución de múltiples funciones de manera más efectiva y evitar posibles anomalías. Además, se ha introducido una variable global `is_running` para que si un proceso está en curso no pueda iniciarse otro.

```
bot.register_next_step_handler(message, funcionejemplo)
```

Figura 3-15 Código de Python. Método `bot.register_next_step_handler` (Autoría propia).

Finalmente, será necesario que el bot pueda solicitar información al usuario y también informarle sobre el estado de los procesos en curso. Para lograr esto, emplearemos métodos para enviar mensajes al usuario y para remitir respuestas a este (véase figura 3-16).

```
bot.reply_to(message, "Ejemplo de mensaje de respuesta.")
bot.send_message(chat_id, "Ejemplo de envío de mensaje")
```

Figura 3-16 Código de Python para respuesta y envío de mensajes (Autoría propia).

3.3 Captura de datos.

Continuando con el segundo módulo de la figura 3-1, en los siguientes apartados se describirá como se ha implementado todo lo necesario para permitir la captura de los datos de carácter público de las plataformas de redes sociales.

3.3.1 YouTube scraper.

Para obtener la transcripción de los vídeos de YouTube e introducirlos en el analizador, emplearemos un método de *scrape* basado en la API de YouTube [92] junto con la librería de Python *youtubetranscriptapi* [93]. Este proceso se centra en la adquisición del archivo de subtítulos del vídeo, incluyendo los subtítulos generados automáticamente, mediante un proceso iterativo de transcripción con la librería. Para que funcione, el usuario introducirá el nombre de un canal, el cual será buscado mediante la API de YouTube Data v3. Si el identificador del canal no se puede obtener mediante el nombre de usuario, se deberá introducir manualmente. En ese momento la herramienta obtendrá, también con la API, el identificador de los N primeros vídeos de ese canal. Una vez obtenida la lista de identificadores de los vídeos, se escribirán en un archivo CSV (*coma-separated values*).

Para poder utilizar la API de YouTube, deberemos registrarnos en su interfaz de programación para obtener un *token* que nos permita la interacción con la plataforma. Una vez estemos registrados podremos utilizar la API sin problema. Sin embargo, se debe tener en cuenta que esta interfaz mantiene una cuota de *tokens* por cada proceso que se le solicite con el fin de que los usuarios no ejecuten procesos innecesarios que puedan ocupar el servicio y provocar anomalías en el funcionamiento de la API [92]. La versión 3 de la API de YouTube asigna de manera predeterminada una cantidad de 10000 *tokens* por usuario al día, teniendo cada llamada a la API un coste variable de *tokens* en función de las diferentes solicitudes que se realicen. Por ejemplo, subir un vídeo en un canal tendría un coste de 1600 *tokens* (véase figura 3-17) por lo que solo podríamos subir 6 vídeos al día desde la API.

videos	list	1
	insert	1,600
	update	50
	de conversiones	50
	getRating	1
	denunciarabuso	50
	borrar	50

Figura 3-17 Ejemplo de cuota de YouTube Data v3 para procesos en vídeos [92].

Para acceder a la función de análisis de esta plataforma dispondremos del comando `/YouTube`. En ese momento el bot enviará un mensaje al chat requiriendo el nombre del canal del cual se quiere obtener la información publicada. Seguido de esto se ejecutará la función `handle_youtube_username` (véase figura 3-19) la cual se encarga de procesar la entrada solicitada al usuario. Además, cuenta con un condicional con la variable `is_running` para evitar que se pueda ejecutar otro proceso si ya se ha empezado un análisis de YouTube.

Una vez se ejecuta la función de obtención mediante nombre de usuario, el bot quedará a la espera que el usuario introduzca el nombre del canal (véase figura 3-18), en caso de que introdujera por error un comando el programa notificaría al usuario y retrocedería al menú principal.

```

29 #Manejador del comando /YouTube
30 @bot.message_handler(commands=['YouTube'])
31 def youtube(message):
32     global is_running
33     global chat_id
34     chat_id = message.chat.id
35     if not is_running:
36         is_running = True
37         bot.reply_to(message, "Ingresa el nombre de usuario del canal de YouTube:")
38         bot.register_next_step_handler(message, handle_youtube_username)
39     else:
40         bot.reply_to(message, "Ya hay un proceso en curso. Por favor, espere a que termine.")

```

Figura 3-18 Fragmento de código de Python para solicitar un canal al usuario (Autoría propia).

Como se puede observar en la figura 3-19, la función `handle_youtube_username` (línea 43) tiene en cuenta tres posibles cursos de acción: que se introduzca un canal existente, que la función `get_channel_id` (línea 55) no funcione correctamente (por canal inexistente o error del usuario) o que se introduzca un comando. Ante un fallo en esta función se ejecutará la función `handle_manual_input` (línea 62) que solicita que se introduzca el identificador del canal manualmente. A todas estas acciones el bot irá informando por el chat del estado de los procesos. En el momento que se obtenga el identificador del canal y se almacene en su correspondiente variable se ejecutará la función `youtubescraper` (línea 59) cuya función es obtener un listado con los identificadores de los últimos vídeos publicados por la cuenta, transcribirlos y guardarlos en formato CSV. Esto lo realiza con la ayuda de tres subfunciones auxiliares: `get_channel_videos`, `get_video_transcript` y `save_to_csv` (véase figura 3-20).

```

42 #Función para obtener el ID a partir del @username
43 def handle_youtube_username(message):
44     global yusername
45     global is_running
46     yusername = message.text
47     if message.text.startswith('/'):
48         # Si el texto contiene un comando, solicitar nuevamente la contraseña
49         bot.reply_to(message, "Ya hay un comando en curso. Por favor, reinicie el proceso")
50         send_welcome(message)
51         is_running = False
52         return
53     else:
54         bot.reply_to(message, "Buscando ID del canal...")
55         channel_id = get_channel_id(yusername)
56         if channel_id:
57             bot.reply_to(message, f"ID del canal encontrado: {channel_id}")
58             bot.send_message(chat_id, "Cargando... Por favor espere.")
59             youtubescraper(channel_id, yusername)
60         else:
61             bot.reply_to(message, "No se pudo encontrar el ID del canal. Introduzca el ID manualmente:")
62             bot.register_next_step_handler(message, handle_manual_input)
63
64 def handle_manual_input(message):
65     global is_running
66     chat_id = message.chat.id
67     channel_id = message.text
68     if message.text.startswith('/'):
69         # Si el texto contiene un comando, solicitar nuevamente la contraseña
70         bot.reply_to(message, "Ya hay un comando en curso. Por favor, reinicie el proceso")
71         send_welcome(message)
72         is_running = False
73         return
74     else:
75         bot.send_message(chat_id, "Cargando... Por favor espere.")
76         youtubescraper(channel_id, yusername)
77
78 # Función para obtener el ID del canal a partir del @username
79 def get_channel_id(yusername):
80     try:
81         youtube = build(YOUTUBE_API_SERVICE_NAME, YOUTUBE_API_VERSION, developerKey=DEVELOPER_KEY)
82         request = youtube.channels().list(part="id", forUsername=yusername)
83         response = request.execute()
84         if 'items' in response and response['items']:
85             return response['items'][0]['id']
86         else:
87             return None
88     except Exception as e:
89         print(f"Error al obtener el ID del canal: {e}")
90         return None

```

Figura 3-19 Fragmento de código de Python para obtener el identificador de usuario (Autoría propia).

```

92 # Función para realizar el raspado de YouTube
93 def youtubescraper(channel_id, yusername):
94     global is_running
95     if channel_id:
96         videos = get_channel_videos(channel_id)
97         if len(videos)!=0:
98             filename = f'{yusername}_videos.csv'
99             save_to_csv(videos, filename)
100             print(f"Los datos de los {len(videos)} últimos videos de {yusername} se han guardado en {filename}")
101             bot.send_message(chat_id,f"Los datos de los {len(videos)} últimos videos de {yusername} se han guardado en {filename}")
102             csv_filename = filename
103             llama(csv_filename)
104             is_running = False
105             return
106         else:
107             bot.send_message(chat_id,"Error en ID del canal.")
108             is_running = False
109             return
110     else:
111         bot.send_message(chat_id,"Error en el proceso por favor vuelva a empezar.")
112         is_running = False
113         return
114
115 def get_channel_videos(channel_id, max_results=10):
116     try:
117         youtube = build(YOUTUBE_API_SERVICE_NAME, YOUTUBE_API_VERSION, developerKey=DEVELOPER_KEY)
118         videos = []
119         next_page_token = None
120         while len(videos) < max_results:
121             request = youtube.search().list(part="snippet", channelId=channel_id, maxResults=min(50, max_results - len(videos)),
122             pageToken=next_page_token, order="date", type="video")
123             response = request.execute()
124             videos.extend(response.get('items', []))
125             next_page_token = response.get('nextPageToken')
126             if not next_page_token:
127                 break
128             return videos[:max_results]
129     except Exception as e:
130         print(f"Error al obtener videos del canal: {e}")
131         return []
132
133
134 def get_video_transcript(video_id):
135     try:
136         transcript_list = YouTubeTranscriptApi.list_transcripts(video_id)
137         for lang in transcript_list:
138             if lang.language_code:
139                 transcript = lang.fetch()
140                 text = '\n'.join([line['text'] for line in transcript])
141                 return text
142         print(f"No se encontró transcripción para el video {video_id}")
143         return None
144     except Exception as e:
145         print(f"Error al obtener transcripción para el video {video_id}: {e}")
146         return None
147
148 def save_to_csv(videos, filename):
149     try:
150         with open(filename, 'w', newline='', encoding='utf-8') as csvfile:
151             fieldnames = ['Title', 'Video ID', 'Published At', 'Transcript']
152             writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
153             writer.writeheader()
154             for video in videos:
155                 transcript = get_video_transcript(video['id']['videoId'])
156                 writer.writerow({'Title': video['snippet']['title'], 'Video ID': video['id']['videoId'],
157                 'Published At': video['snippet']['publishedAt'], 'Transcript': transcript})
158             print(f"Datos de los últimos {len(videos)} videos guardados en {filename}")
159     except Exception as e:
160         print(f"Error al guardar datos en el archivo CSV: {e}")

```

Figura 3-20 Fragmento de código de Python para obtener y almacenar subtítulos de YouTube (Autoría propia).

Como se puede observar en la figura 3-20 el proceso de *scraping* comienza con la llamada a la función `youtubescraper` (línea 93) introduciéndole las variables de nombre de usuario e identificador del canal obtenidas previamente. Mediante el identificador del canal se ejecuta la función `get_channel_videos` (línea 118) la cual obtendrá y guardará en la variable `videos` mediante un proceso iterativo los N primeros identificadores de vídeo del canal, siendo N un número configurable que determinaremos cambiando el parámetro `max_results` en la línea 115, en función de las características del LLM del analizador. Una vez obtenidos los identificadores de vídeo se llama a la función `save_to_csv` (línea 53) que también mediante un proceso iterativo y la función secundaria `get_video_transcript` (línea 38) obtendrá por pasos los subtítulos del vídeo almacenándolos en el archivo CSV resultado.

Una vez completado este proceso, se habrá generado temporalmente un archivo en formato CSV (véase figura 3-21) conteniendo la transcripción correspondiente al número de vídeos seleccionados. A continuación, se ejecutará la función `llama`, la cual se explicará más adelante, con los datos obtenidos del proceso como variable `prompt`, permitiendo su análisis por parte del LLM para generar una respuesta al usuario.

```
Title,Video ID,Published At,Transcript
NATO Secretary General visits NATO partners in the South Caucasus,
2Hy1vZAxRuQ,2024-03-20T14:26:53Z,"Peace and stability is not only important here
but for security more broadly. Armenia and Azerbaijan now have an opportunity
to achieve enduring peace after years of conflict. I can just encourage you
to seize this opportunity to reach a lasting peace agreement with Armenia.
NATO's partnership with Georgia is already strong and today we discussed how
to develop it even further. NATO supports Armenian sovereignty
and territorial integrity, and your peaceful aspirations."Sweden se joins NATO,
ERA0go2s0jA,2024-03-11T13:54:49Z,"Sweden has joined the NATO family.
We have been close friends for decades and now we'll keep on working to maintain
our security as Allies. Sweden's historic decision to become a NATO member,
makes our Alliance even stronger and our people safer.Välkommen Sverige.
"We're NATO aircrew, #ofcourse we&#39;re filming this in the air #shorts",
T35zA2GNdqc,2024-03-05T09:37:35Z,"we're NATO air crew of course we're
always on time we a NATO air crew of course we use a lot of acronyms we're
NATO air crew of course our hair is always messy we are NATO air crew of
course we love following procedures we're NATO air crew of course our watch
is set to Zulu time we're NATO air crew of course we keep extra barf bags for
when our students getsick"
Everything you need to know about NATO's AWACS plane in 40 seconds 🇺🇸🇨🇦
#shorts,sxT7bsz7nvQ,2023-10-20T09:07:22Z,"we're currently in the Aira Poland and
this is everything you need to know about the awax in 40 seconds we're in a
Boeing 707 we have three people two pilots one engineer in front taking a
safe thr disguise we have a big radar fitted on the back of the aircraft we
can detect and tracks service and air tracks several hundred miles away we
have four engines we can stay in the air for 8.5 hours if we have to and the
mission requires we can extend that time with air to a refueling crew goes up to
33 you have six bunks in the backs two oven only one laboratory a couple seats
if could need to take a rest and that's everything you need to know in less than
40 seconds about the NATO ax"Tanks don't float. Good thing this pontoon bridge does!
| #shorts,gdqD5I3XIn4,2023-08-08T07:43:52Z,"my name is Captain Don Hamrick the
Bradley's behind me they wave 33 tons today we are actually going across the
Danube River
[Music]
to Lithuania lt, host of the 2023 NATO Summit #shorts",F_jj-q9hLfw,2023-07-05T09:10:09Z,
"love us welcome to Lithuania where the speed of internet The Bravery of the
people and the beauty of the nature Knows No Limits located in the geographical center of
Europe on the Baltic Sea Lithuania literally translates to land of rain and we are known
for our Green Forest and our Lakes this year our Capital Village home over half
a million people place hosts to the NATO Summit our history has often been turbulent
with long period of occupation by other nations despite all tough times we kept our
language and our love of Freedom alive Lithuania joined to NATO alliance in
```

Figura 3-21 Ejemplo de archivo de transcripción de vídeo del canal de la OTAN (Autoría propia).

Puede observarse a continuación el flujo completo del proceso (véase figura 3-22):

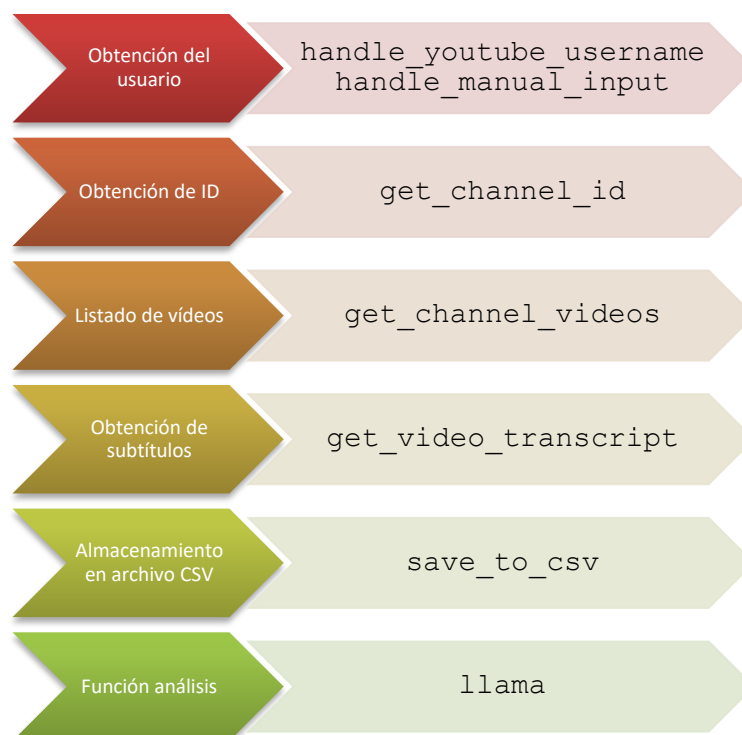


Figura 3-22 Flujo del proceso de raspado de YouTube (Autoría propia).

3.3.2 Instagram scraper.

Al igual que en el ejemplo anterior, el objetivo de este proceso es extraer la información pública de un usuario para introducirla en el analizador. Para ello utilizaremos la librería de Python *instaloader*, la cual permite, a través de una cuenta de Instagram, obtener publicaciones e información pública de otros usuarios. La condición de uso de esta librería es que debe utilizarse desde un usuario autenticado en Instagram, por ello el bot solicitará inicialmente un usuario y una contraseña para acceder a la plataforma.

La secuencia de ejecución es similar a la captura de datos en YouTube. Sin embargo, en vez de obtener la transcripción de los vídeos, obtendremos los pies de fotografía también llamados *captions* en inglés. El acceso al proceso será con el comando “/Instagram” y comenzará con la petición del nombre del perfil sobre el que se quiere obtener la información publicada. Este usuario se obtendrá mediante la función `handle_instagram_username`. Una vez se haya almacenado el nombre de usuario en su correspondiente variable, se requerirá el inicio de sesión a la plataforma por lo que se solicitará un usuario y contraseña mediante dos funciones auxiliares, `handle_instagram_user` y `handle_instagram_password` para la posterior ejecución de la función `iniciar_sesion` que permite el *log in* en la plataforma (véase figura 3-23 línea 248). Esta función maneja diversos escenarios, como la detección de perfiles inexistentes o credenciales inválidas, proporcionando mensajes de error descriptivos en caso de fallo.

Cabe destacar que Instagram cuenta con un sistema de autenticación en dos pasos para el *log in* si el usuario así lo tiene configurado, por lo que dentro de la función `iniciar_sesion` se deberá tener en cuenta esta posibilidad para el acceso.

```

225 def handle_instagram_user(message):
226     global user, is_running
227     if message.text.startswith('/'):
228         # Si el texto contiene un comando, solicitar nuevamente la contraseña
229         bot.reply_to(message, "Ya hay un comando en curso. Por favor, reinicie el proceso")
230         is_running = False
231         send_welcome(message)
232         return
233     else:
234         user = message.text
235         bot.reply_to(message, "Por favor, introduzca su contraseña:")
236         bot.register_next_step_handler(message, handle_instagram_password)
237
238 def handle_instagram_password(message):
239     global password, is_running
240     if message.text.startswith('/'):
241         # Si el texto contiene un comando, solicitar nuevamente la contraseña
242         bot.reply_to(message, "Ya hay un comando en curso. Por favor, reinicie el proceso")
243         send_welcome(message)
244         is_running = False
245         return
246     else:
247         password = message.text
248         if iniciar_sesion(user, password):
249             bot.reply_to(message, "Sesión iniciada correctamente")
250             bot.reply_to(message, f"Obteniendo las 20 primeras publicaciones de {username}")
251             scrape_instagram(username, user, password)
252             is_running = False # Establecer la variable global en False cuando el proceso haya terminado
253         else:
254             bot.reply_to(message, "Credenciales inválidas. Por favor, inténtelo de nuevo.")
255             is_running = False # Asegurarse de que la variable global se actualice incluso si hay un error en el proceso

```

Figura 3-23 Código de Python para inicio de sesión en Instagram (Autoría propia).

```

296 # Función para raspar publicaciones de Instagram
297 def scrape_instagram(username, user, password):
298     L = iniciar_sesion(user, password)
299     global is_running
300     if L is None:
301         return
302
303     try:
304         profile = instaloader.Profile.from_username(L.context, username)
305         bot.send_message(chat_id, f"Perfil encontrado: {profile.username}")
306         bot.send_message(chat_id, f"Seguidores: {profile.followers}")
307         bot.send_message(chat_id, f"Siguiendo: {profile.followees}")
308         bot.send_message(chat_id, f"Número de publicaciones: {profile.mediacount}")
309
310         # Crear un archivo CSV para guardar los pies de foto
311         csv_file = f"{username}_captions.csv"
312         with open(csv_file, mode='w', newline='', encoding='utf-8') as file:
313             writer = csv.writer(file)
314             writer.writerow(["Caption"]) # Escribir encabezado del archivo CSV
315
316             count = 0
317             for post in profile.get_posts():
318                 caption = clean_caption(post.caption)
319                 writer.writerow([caption]) # Escribir pie de foto limpio en el archivo CSV
320                 count += 1
321                 if count >= 20:
322                     break
323
324         print(f"Los pies de foto se han guardado en el archivo {csv_file}")

```

Figura 3-24 Código de Python para obtener los pies de foto de un perfil (Autoría propia).

Una vez el usuario ha introducido los parámetros iniciales, el *script* procede a ejecutar la función `scrape_instagram` (véase figura 3-24 línea 297) introduciéndole las variables que contienen el usuario, la contraseña y el perfil del cual se quiere obtener la información. Desde aquí se hace la llamada a la función `iniciar_sesion` que realiza la autenticación del usuario en la plataforma. Si la autenticación no es correcta se informa al usuario y se repite el proceso, si lo es comienza la ejecución de *instaloader*.

En primer lugar, se accede al perfil mostrando en el chat del nombre del perfil, el número de usuarios que siguen a ese perfil, el número de usuarios a los que sigue y finalmente el número de publicaciones de las que se tiene acceso. Posteriormente se crea el archivo en el que se almacenarán los pies de foto del perfil y comienza un proceso iterativo en el que se escriben dichos pies de foto, limpiándolos de caracteres extraños mediante la función `clean_caption` (véase figura 3-25), en el archivo CSV que se introducirá al analizador.

```

271 # Función para limpiar el pie de foto de Instagram
272 def clean_caption(caption):
273     # Eliminar caracteres especiales y emojis
274     clean_caption = re.sub(r'^[\x00-\x7F]+', '', caption)
275     # Eliminar caracteres de nueva línea
276     clean_caption = clean_caption.replace('\n', ' ')
277     return clean_caption

```

Figura 3-25 Código de Python para limpiar caracteres especiales de un texto (Autoría propia).

Puede observarse a continuación el flujo completo del proceso (véase figura 3-26):



Figura 3-26 Flujo del proceso de raspado de Instagram (Autoría propia).

3.3.3 Analizador de archivos.

Para configurar una opción de análisis con un carácter más universal se dispondrá de este mecanismo de análisis de archivos. Este comando permitirá recibir documentos en formato “.txt” para analizarlos con el LLM.

El proceso comenzará cuando el usuario introduzca el comando “/Doc”. A continuación, el bot solicitará al usuario que introduzca un documento en el formato especificado y se ejecutará la función `handle_document` (véase figura 3-27 línea 345). Esta función comprobará que el archivo está en el formato adecuado y abrirá el documento de texto introduciendo su contenido como cadena de texto en la variable `prompt` de manera fragmentada por si el archivo fuera de un tamaño excesivo y pudiera provocar anomalías en el proceso. Seguidamente se ejecutará la función de análisis sobre la variable `prompt` ya almacenada.

```

333 @bot.message_handler(commands=['/Doc'])
334 def doc(message):
335     global is_running
336     global chat_id
337     chat_id = message.chat.id
338     if not is_running:
339         is_running = True
340         bot.reply_to(message, "Ingresa un documento en formato txt:")
341         bot.register_next_step_handler(message, handle_document)
342     else:
343         bot.reply_to(message, "Ya hay un proceso en curso. Por favor, espere a que termine.")
344
345 def handle_document(message):
346     global is_running
347     global prompt
348     if message.document:
349         document_id = message.document.file_id
350         file_info = bot.get_file(document_id)
351         file_path = file_info.file_path
352         downloaded_file = bot.download_file(file_path)
353         try:
354             # Dividir el archivo en partes más pequeñas
355             file_chunks = [downloaded_file[i:i+4096] for i in range(0, len(downloaded_file), 4096)]
356             # Decodificar y procesar cada parte
357             for chunk in file_chunks:
358                 file_content = chunk.decode("utf-8")
359                 # Introducir el contenido en la variable prompt
360                 if prompt:
361                     prompt += file_content
362                 else:
363                     prompt = file_content
364             # Responder al usuario con el contenido completo
365             bot.reply_to(message, "Documento recibido, analizando contenido:")
366             print(prompt)
367         except Exception as e:
368             print("Error al decodificar el archivo:", e)
369             bot.reply_to(message, "Lo siento, ocurrió un error al procesar el documento.")
370         finally:
371             is_running = False
372             send_welcome(message)
373     else:
374         bot.reply_to(message, "Debe enviar un documento en formato txt. Por favor reinicie el proceso")
375         is_running = False
376         send_welcome(message)

```

Figura 3-27 Código de Python para manejo de documentos de texto (Autoría propia).

3.4 Análisis de datos.

A continuación, se desarrollará el tercer módulo del proyecto el cual consiste en el proceso de análisis y clasificación de los datos. Para ello se hará uso de un corpus de artículos extraídos de Wikipedia etiquetados como relacionados o no con el ámbito de la defensa. A partir de esta compilación de datos construiremos un *dataset* enfocado al ajuste (*fine tuning*) de un LLM previamente entrenado, proporcionado el formato que tendrá el *prompt*, es decir, el formato de la entrada de datos al LLM. En este caso se utilizará el LLM de Meta denominado Llama 2 el cual se introdujo en el punto 2.2.5 del presente documento.

3.4.1 Construcción del dataset.

La construcción de un *dataset* de alta calidad es esencial para el éxito del *fine-tuning* en el aprendizaje automático. Para ello es imprescindible que esté intrínsecamente enfocado a la tarea objetivo. Además, debe ser variado y mostrar patrones claros exentos de ambigüedad para que el ajuste sea preciso. En este caso se ha utilizado un listado de artículos relacionados con el ámbito que queremos identificar, así como un grupo de artículos no relacionados con dicho contexto con el fin de que el modelo sea capaz de reconocer y diferenciar una mayor variedad de contextos. Este listado de artículos fue creado durante el desarrollo del trabajo fin de grado de María Fernández Torrejón en el año 2023 [80].

Etiquetaremos cada uno de los artículos para que el modelo relacione la respuesta esperada en función de los diferentes contenidos del texto. La etiqueta para los artículos relacionados será “Yes, this text is highly related with defense topics” y “No, this text is not related with defense topics” para el conjunto de artículos no relacionados (véase figura 3-28). Utilizaremos etiquetas en inglés para facilitar la comprensión del LLM el cual está principalmente entrenado en este idioma. Además, se debe formatear el campo de texto para eliminar caracteres extraños que puedan provocar errores en el entrenamiento o en el formato del archivo de datos CSV.

answer	text
string · classes	string · lengths
	
2 values	169 105k
Yes, this text is highly related with defense or military topics	The Amazon-class frigates of 1773, made up of 32-gun fifth rates with a main battery of 12-pounder guns. The class comprised eighteen ships; Amazon, Ambuscade and Thetis were launched in 1773; the...
Yes, this text is highly related with defense or military topics	HNLMs Van Speijk (F802) () was a frigate of the . The ship was in service with the Royal Netherlands Navy from 1967 to 1986. The frigate was named after Dutch naval hero Jan van Speyk. The ship's radio...
Yes, this text is highly related with defense or military topics	HMS Poseidon (P99) was a designed and built by Vickers Shipbuilding and Engineering in Barrow-in-Furness for the Royal Navy, launched on 22 August 1929. She spent most of her short career assigned to...
Yes, this text is highly related with defense or military topics	Galera Victoria was a Spanish galleon that sank in 1729 in the Atlantic Ocean off Gijón, Spain, while she was on her maiden voyage. Construction Galera Victoria was launched on 14 February 1729 in...
Yes, this text is highly related with defense or military topics	{{ {{Infobox ship image image=Sir Samuel Hood's engagement with the French Squadron off Rochefort, Sept. 25, 1806.jpg caption=The engagement with the French Squadron off Rochefort, HMS Monarch Capt...
Yes, this text is highly related with defense or military topics	The Cold War Victory Medal is both an official medal of the U.S. National Guard and an unofficial military medal of the United States. It is awarded by the U.S. states of Louisiana and Texas, and in...
Yes, this text is highly related with defense or military topics	The Armeno-Georgian War was a short border dispute fought in December 1918 between the newly independent Democratic Republic of Georgia and the First Republic of Armenia, largely over the control...
Yes, this text is highly related with defense or military topics	USS Sample (FF-1048) was a frigate in the US Navy and the seventh in its class named for William Dodge Sample (9 March 1898 – 2 October 1945) who was a Rear Admiral in the United States Navy and an Escort...
Yes, this text is highly related with defense or military topics	The Cold War originated in the breakdown of relations between the United States and the Soviet Union and their respective allies, the Western Bloc and the Eastern Bloc, in the years 1945–1949. The origin...

Figura 3-28 Fragmento del *dataset* utilizado (Autoría propia).

El *dataset* utilizado se ha construido con un total de 144 entradas de texto etiquetadas, de las cuales el 64% son entradas de detección positiva y 36% entradas sin ninguna relación con el contexto a identificar. Estos *dataset* son completamente personalizables y podremos enfocarlo según la tarea que queramos ejecutar. Concretamente en este caso, podríamos modificar el *dataset* en función del campo

que necesitemos investigar cómo podrían ser noticias falsas, comportamientos violentos o incluso movimientos terroristas.

El tamaño del *dataset* que se ha utilizado ha dependido principalmente de las limitaciones de memoria RAM de GPU de Google Colab ya que en función del tamaño del conjunto de datos se utilizará más o menos memoria durante el entrenamiento. En este caso el *dataset* cuenta con un peso de 1.09 MB lo cual nos permitirá realizar el entrenamiento sin colapsar la GPU. El archivo se encuentra disponible en la página de Hugging Face [94], una empresa tecnológica enfocada al desarrollo de herramientas de procesamiento del lenguaje natural que permite compartir, entre otras cosas, modelos y *datasets* con la comunidad de desarrolladores (véase figura 3-29).

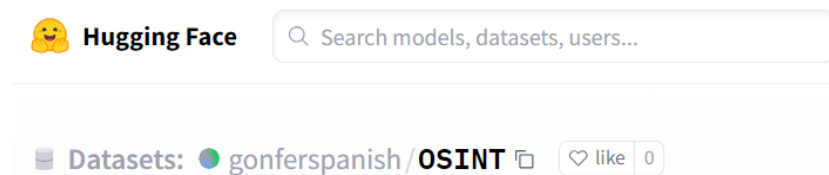


Figura 3-29 Disposición del *dataset* en Hugging Face (Autoría propia).

Para enfocar la tarea de clasificación, introduciremos también una instrucción clara al modelo con el nombre de *system_prompt*. La cual tendrá el siguiente formato: “*Classify whether the following text is explicitly related to defense or military topics or not.*”

Es importante destacar que las entradas que se hagan al modelo deben de tener el formato adecuado para que este sea capaz de diferenciar el *prompt*, el *system_prompt* y la respuesta esperada. En este caso Llama 2 utiliza etiquetas para diferenciar cada parte de la entrada (véase figura 3-30).

```
LLaMA='''<s>[INST]
<<SYS>>{{Introducir aquí la instrucción para el modelo}}<<SYS>>
{{Introducir aquí la entrada del usuario}}
[/INST] {{Respuesta generada}} </s>'''
```

Figura 3-30 Ejemplo de formato de *prompt* para Llama 2 (Autoría propia).

3.4.2 Parámetros de entrenamiento.

Para el entrenamiento del modelo utilizaremos una librería desarrollada por Meta AI Research [95] en 2021 denominada TRL (*Transformer Reinforcement Learning*) [96]. Esta librería permite realizar entrenamientos supervisados de modelos LLM mediante *dataset* pudiendo personalizar los parámetros del entrenamiento en función del resultado deseado.

Los parámetros de entrenamiento determinan aspectos fundamentales del proceso de entrenamiento y la optimización del modelo. A continuación, se resumen los principales parámetros que se deben configurar:

- Número de épocas de entrenamiento: indica cuántas veces se pasará por el conjunto de datos durante el entrenamiento. En nuestro caso realizaremos tres vueltas al modelo que es el máximo que nos permitirán los 16 GB de RAM de GPU de Colab (véase figura 3-31 línea 43).

- Tamaño del lote de GPU para entrenamiento y evaluación: define el número de ejemplos de entrenamiento y evaluación que se procesarán simultáneamente en cada dispositivo de hardware (GPU). Seleccionaremos un lote de valor dos para no exceder la capacidad de Colab (véase figura 3-31 líneas 45 y 47).
- Pasos de acumulación de gradiente: es el número de pasos que se realizan antes de actualizar los pesos de un modelo durante el entrenamiento. Debemos disminuirlo si se aumenta el tamaño del lote para compensar el uso de RAM (véase figura 3-31 línea 49).
- Checkpoint de gradiente: se activa para calcular y almacenar solo algunos valores intermedios de los gradientes en lugar de todos con el fin de ahorrar memoria (véase figura 3-31 línea 51).
- Recorte máximo del gradiente: controla el tamaño máximo de los gradientes durante el entrenamiento, ayudando a evitar problemas como el estallido del gradiente (véase figura 3-31 línea 53).
- Tasa de aprendizaje: configura el tamaño de los pasos que se dan para ajustar los parámetros del modelo durante el entrenamiento. Se configura para hacer el ajuste con la mayor celeridad, pero sin excederse para no provocar anomalías (véase figura 3-31 línea 55)
- Decaimiento de peso: aplica una penalización a los parámetros para prevenir que los valores de los parámetros del modelo se vuelvan muy grandes durante el entrenamiento, ayudando a evitar el sobreajuste y el colapso de RAM. (véase figura 3-31 línea 57).
- Tipo de optimizador: especifica el algoritmo de optimización utilizado durante el entrenamiento. El modelo *paged_adamw_32bit* es una variante eficiente del optimizador AdamW y se utiliza para calcular las actualizaciones de los parámetros. Se ha seleccionado este optimizador para acelerar el ajuste (véase figura 3-31 línea 59).
- Programador de tasas de aprendizaje: define cómo cambia la tasa de aprendizaje durante el entrenamiento. Se utilizará un programa modelo coseno que nos permitirá minimizar la tasa de pérdida de entrenamiento, parámetro que cuantifica la discrepancia entre estas predicciones y las etiquetas verdaderas, en el menor número de épocas posibles (véase figura 3-31 línea 61).
- Número máximo de pasos: limita el número total de pasos de entrenamiento. En nuestro caso no dispondremos límite de pasos (véase figura 3-31 línea 63).
- Ratio de calentamiento: controla la proporción de pasos de calentamiento durante el ajuste fino (véase figura 3-31 línea 65).
- Agrupamiento de lotes por longitud: determina si se agrupan los ejemplos de entrenamiento por longitud de secuencia. Esto permite agilizar el proceso, pero puede afectar a la precisión del resultado (véase figura 3-31 línea 67).
- Pasos de guardado y registro: especifica la frecuencia con la que se guardan los puntos de control del modelo y se registran los resultados durante el entrenamiento (véase figura 3-31 líneas 69 y 71).

- Longitud máxima de secuencia: define la longitud máxima permitida para las secuencias de entrada. No dispondremos de longitud máxima permitida para que no se pierda información del *dataset* (véase figura 3-31 línea 75).
- Empaquetamiento de ejemplos: sirve para empaquetar entradas del *dataset* en lotes para optimizar el uso de la memoria. En este caso no utilizaremos este método debido a que utilizaremos una función de formato que veremos más adelante y no es compatible con el uso desde parámetro (véase figura 3-31 línea 77).

```

39 #####
40 # TrainingArguments parameters
41 #####
42
43 num_train_epochs = 3 #Número de vueltas al dataset
44
45 per_device_train_batch_size = 2 #Tamaño de GPU para entrenamiento
46
47 per_device_eval_batch_size = 2 #Tamaño de GPU para evaluación
48
49 gradient_accumulation_steps = 1 #Gradiente acumulado en pasos
50
51 gradient_checkpointing = True #Punto de control de gradiente
52
53 max_grad_norm = 0.3 #Gradiente máximo
54
55 learning_rate = 2e-4 #Tasa de aprendizaje
56
57 weight_decay = 0.001 #Penalización de pesos
58
59 optim = "paged_adamw_32bit" #Optimizador
60
61 lr_scheduler_type = "cosine" #Programador de tasa de aprendizaje
62
63 max_steps = -1 #Número de pasos
64
65 warmup_ratio = 0.03 #Tasa de calentamiento
66
67 group_by_length = False #Agrupamiento de secuencias de misma longitud
68
69 save_steps = 0 #Almacenamiento de puntos de control
70
71 logging_steps = 20 #Pasos para cada registro
72 #####
73 # SFT parameters
74 #####
75 max_seq_length = None #longitud máxima de secuencia
76
77 packing = False #Empaquetamiento en lotes

```

Figura 3-31 Parámetros de entrenamiento (Autoría propia).

3.4.3 Optimización del entrenamiento.

Para garantizar la eficacia del entrenamiento, reducir los tiempos de ejecución y disminuir el uso de RAM por parte del modelo y el entrenador utilizaremos herramientas de cuantización y LoRA (*Low-Rank Adaptation of Large Language Models*) [97].

LoRA es una técnica de ajuste fino utilizada para entrenar LLM en tareas o dominios específicos la cual opera mediante la inserción de un número reducido de pesos nuevos en el modelo. El documento de LoRA garantiza que este método puede reducir el número de parámetros entrenables hasta en 10,000 veces y la necesidad de memoria de la GPU hasta 3 veces, mientras que aún mantiene o mejora la calidad del modelo de ajuste fino en diversas tareas [97].

Derivado de la herramienta anterior nace QLoRA (*Quantization and Low-Rank Adaptation of Large Language Models*) [98], la versión extendida de LoRA. Esta técnica funciona cuantizando la precisión de los parámetros de peso en el LLM preentrenado a una precisión de 4 bits cuando normalmente son almacenados en formato de 32 bits (véase figura 3-32). Esto reduce considerablemente la huella de memoria del LLM, lo que permite realizar el ajuste fino en una sola GPU utilizando hardware menos potente [98].

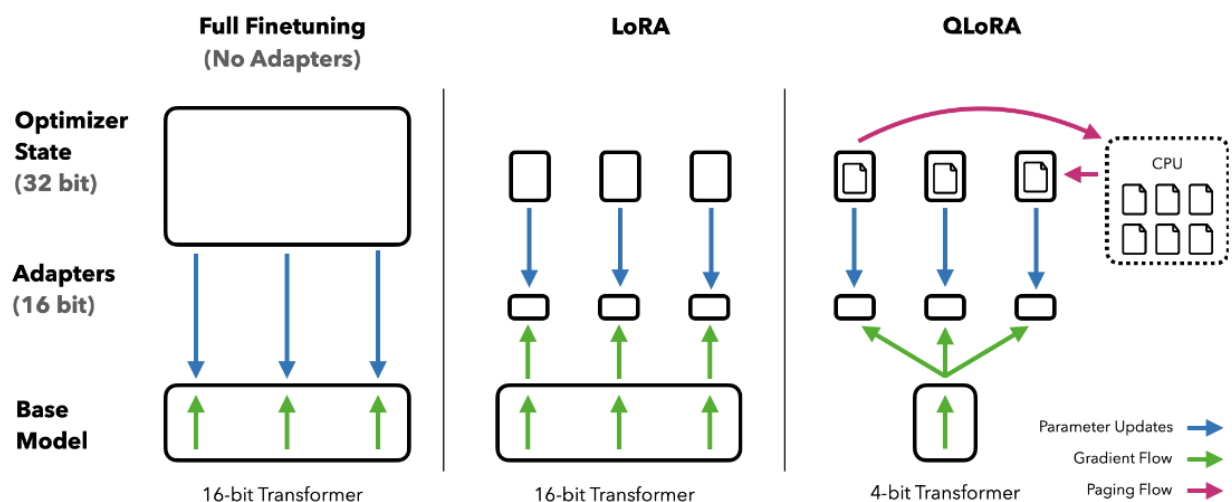


Figura 3-32 Esquema de LoRA y QLoRA [98].

Para configurar y ejecutar estas técnicas de optimización utilizaremos las bibliotecas *peft* [99] y *bitsandbytes* [100]. A continuación, veremos los parámetros que se deben configurar con carácter previo al entrenamiento:

- *Lora_r*: la técnica LoRA implica la inserción de matrices de descomposición de rango entrenables en cada capa de la arquitectura del transformador. Con el parámetro *lora_r* se controla la dimensión de estas matrices, influyendo en la flexibilidad y la capacidad de representación del modelo (véase figura 3-33 línea 15).
- *Lora_alpha*: se utiliza para ajustar la magnitud de los valores de salida de las capas de LoRA, permitiendo una modulación adecuada de la información de salida en función de las necesidades específicas del modelo o de la tarea (véase figura 3-33 línea 18).
- *Lora_dropout*: la probabilidad de abandono o *dropout* es una técnica de regularización que se emplea durante el entrenamiento para mitigar el sobreajuste, desactivando aleatoriamente un porcentaje de unidades neuronales en cada paso de entrenamiento (véase figura 3-33 línea 21).

- *Use_4bit*: este parámetro activa la carga del modelo base con precisión de 4 bits, lo que puede contribuir a la eficiencia del almacenamiento y al desempeño computacional (véase figura 3-33 línea 28).
- *Bnb_4bit_compute_dtype*: define el tipo de datos de cálculo para los modelos base de 4 bits. Este parámetro determina el formato de datos empleado durante los cálculos realizados en los modelos base de precisión de 4 bits, lo que puede impactar en la precisión y eficiencia del proceso de inferencia (véase figura 3-33 línea 31).
- *Bnb_4bit_quant_type*: indica el tipo de cuantificación aplicado a los modelos base de 4 bits. Esta especificación influye en la manera en que se realizan los procesos de cuantificación para adaptar los parámetros del modelo a la representación de precisión reducida, afectando directamente a la calidad de la inferencia y al rendimiento del modelo, en este caso utilizaremos cuantización nf4 (*normalize float 4*) que es la recomendada por defecto (véase figura 3-33 línea 34).
- *Use_nested_quant*: este parámetro activa la cuantificación anidada para los modelos base de 4 bits, también conocida como doble cuantificación. Al habilitar esta opción, se emplea una estrategia de cuantificación adicional para los modelos de precisión reducida, lo que puede mejorar la compresión de los parámetros del modelo y la eficiencia del almacenamiento (véase figura 3-33 línea 37). En este caso no se utilizará debido a la escasez de recursos de computación.

```
10 #####
11 # QLoRA parameters
12 #####
13
14 #Dimensión LoRA
15 lora_r = 64
16
17 #Parámetros alpha para escalada
18 lora_alpha = 16
19
20 #Probabilidad de dropout
21 lora_dropout = 0.1
22
23 #####
24 # bitsandbytes parameters
25 #####
26
27 # Activar procesamiento en precisión de 4-bit
28 use_4bit = True
29
30 #Formato para 4-bit
31 bnb_4bit_compute_dtype = "float16"
32
33 #Tipo de cuantización
34 bnb_4bit_quant_type = "nf4"
35
36 #Usar doble cuantización
37 use_nested_quant = False
```

Figura 3-33 Parámetros de cuantización y QLoRA (Autoría propia).

3.4.4 Ajuste fino.

Una vez tenemos seleccionados los parámetros de entrenamiento y configurada la optimización QLoRA, podemos iniciar el proceso de entrenamiento. Partiremos del modelo Llama 2 de 7 mil millones de parámetros ya que es suficiente para la tarea que queremos entrenar y la memoria de GPU en Colab es limitada. Para utilizar este modelo únicamente necesitaremos obtener un *token* de autenticación de la plataforma Hugging Face antes mencionada. También utilizaremos dicha plataforma para descargar nuestro *dataset* y cargar el modelo entrenado después del ajuste fino.

Finalmente se debe establecer una función para formatear los datos de manera acorde al *prompt* de Llama 2 para que podamos utilizarlos en el proceso de ajuste del modelo (véase figura 3-34).

```
2 dataset = load_dataset(dataset_name, split="train")
3
4 def formatting_func(example):
5     text = f'''<s>[INST] <<SYS>>Classify whether the following text
6     is explicitly related to defense or military topics or not.<<SYS>>
7     {{Text}}: {example['text']}
8     [/INST]{{Answer}}: {example['answer']}</s>'''
9     return text
```

Figura 3-34 Código de Python para formatear los datos del ajuste fino (Autoría propia).

Ante estos parámetros y características se han obtenido los siguientes resultados de *training loss* (véase tabla 3-1):

Paso	Training Loss
20	2,221
40	2,067
60	2,009
80	2,007
100	1,973
120	2,006
140	1,855

Tabla 3-1 Evolución del *training loss* durante el ajuste (Autoría propia).

Estos resultados representan la discrepancia entre las predicciones del modelo y las etiquetas reales durante el ajuste a medida que avanza el proceso. En estos datos, se espera que la pérdida disminuya con el tiempo a medida que el modelo se ajusta mejor a los datos de entrenamiento. Como se observa en la figura 3-35 la pérdida disminuye con los pasos del entrenamiento lo que indica que el modelo se ha ido ajustando correctamente a lo largo del proceso.

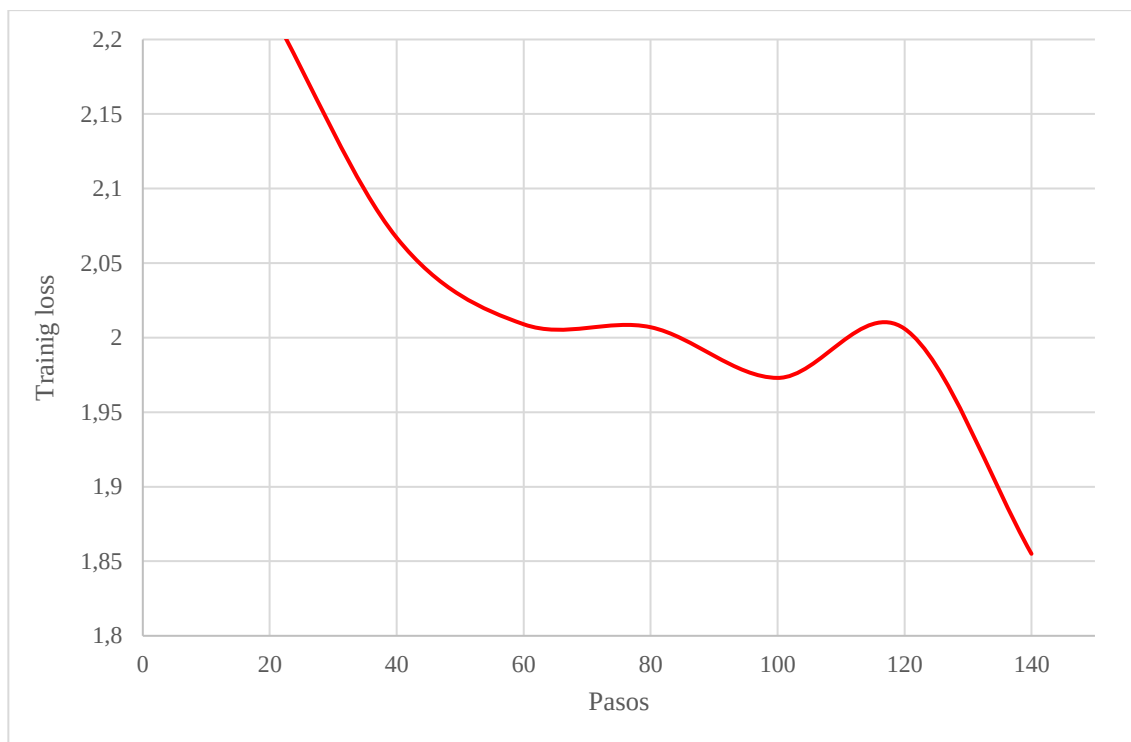


Figura 3-35 Gráfica evolución del *training loss* durante el ajuste (Autoría propia).

Puede decirse que el resultado del ajuste es satisfactorio ya que se obtiene una pérdida de 1,85 partiendo de una pérdida de 2,20 en el paso 20 del entrenamiento. Sin embargo, si se contara con mayor capacidad de GPU podrían configurarse unos parámetros iniciales que permitieran un ajuste más preciso y con un mayor número de pasos.

3.4.5 Inferencia.

En el contexto del aprendizaje automático se denomina inferencia al proceso de utilizar un modelo ajustado para realizar predicciones, clasificaciones o en general desarrollar las acciones para las que se ha ajustado dicho modelo.

En este caso, tras el proceso de ajuste los datos se han cargado en el repositorio de la plataforma Hugging Face, por lo que desde ese instante cualquier usuario tendrá acceso a él y podrá utilizarlo sin restricciones. El modelo se ha guardado bajo el nombre de “gonferspanish/PAQUITA” (acrónimo de autoría personal referente a Proceso de AdQUISición de Inteligencia para su Tratamiento y Análisis) [101].

Para utilizar el analizador dispondremos de una función que extraiga el contenido CSV de los archivos obtenidos mediante el módulo de captura de datos y lo almacene en una variable de cadena de texto denominada `user_prompt` para posteriormente introducirla en el modelo ajustado mediante la función `llama` (véase figura 3-36 línea 424). Esta función construirá el *prompt* añadiendo la instrucción para el sistema y las etiquetas de formato de Llama 2 (véase figura 3-36 línea 432). Tras diversas pruebas con distintas instrucciones, se ha construido la siguiente entrada para obtener la respuesta más cercana al objetivo de la herramienta: *analyze the text content of the provided posts and classify whether the account predominantly discusses military or defense-related topics. Provide the percentage of content related to these topics.*

Posteriormente se transmitirá el *prompt* al modelo habiendo transformado su contenido a *tokens* previamente para que el analizador sea capaz de interpretarlo (véase figura 3-36 línea 434). Para evitar

anomalías en el texto resultado generado, se ha limitado la entrada de *tokens* a 4000 (véase figura 3-36 línea 409), ya que el límite de Llama 2 es de 4096 *tokens* [94].

El texto resultante será procesado de manera similar que el de entrada, ya que el modelo generará la respuesta en *tokens* y se deberán convertir en texto para su lectura e interpretación (véase figura 3-36 línea 437). Seguido de ello, eliminaremos el *prompt* de la respuesta, puesto que sería redundante (véase figura 3-36 línea 438), para finalmente enviar el resultado del análisis por mensaje al chat de Telegram.

```

406 #-----
407 #----- LLAMA 2 -----
408 #-----
409 max_tokens = 4000
410 def limit_input_tokens(input_text):
411     input_tokens = tokenizer.tokenize(input_text) #Tokenizar el texto de entrada
412     if len(input_tokens) > max_tokens:
413         input_tokens = input_tokens[:max_tokens]
414         input_text = tokenizer.convert_tokens_to_string(input_tokens)
415     return input_text
416
417 def abrir_csv(csv_filename):
418     archivo_csv = csv_filename
419     datos = pd.read_csv(archivo_csv)
420     entrada = '\n'.join(['\t'.join(map(str, fila)) for fila in datos.values])
421     entrada = limit_input_tokens(entrada) #Limitar los tokens de entrada
422     return entrada
423
424 def llama(u_prompt):
425     system_prompt = '''Analyze the text content of the provided posts and classify whether the
426     account predominantly discusses military or defense-related topics. Provide the percentage
427     of content related to these topics. Respond with 'Yes' or 'No', followed by the percentage if applicable.'''
428     bot.send_message(chat_id, "Analizando contenido. Por favor espere...")
429     user_prompt = abrir_csv(u_prompt)
430     B_INST, E_INST = "[INST]", "[/INST]"
431     B_SYS, E_SYS = "<S>", "<S>"
432     prompt = f"{B_SYS} {B_INST}{system_prompt.strip()}{E_SYS}{user_prompt.strip()}{E_INST}\n\n"
433     prompt = limit_input_tokens(prompt)
434     inputs = tokenizer([prompt], return_tensors="pt").to(runtimeFlag)
435     streamer = TextStreamer(tokenizer)
436     completion = model.generate(**inputs, streamer=streamer, max_new_tokens=500)
437     tokenized_completion = tokenizer.decode(completion[0], skip_special_tokens=True)
438     inst_end_index = tokenized_completion.find("[/INST]") #Encontrar comienzo de la respuesta
439     if inst_end_index != -1:
440         tokenized_completion = tokenized_completion[inst_end_index + len("[/INST]"): ] #Extraer el texto respuesta
441     bot.send_message(chat_id, f"{tokenized_completion}")

```

Figura 3-36 Código de Python para ejecutar Llama 2 (Autoría propia).

Para ejecutar las funciones que permitan la inferencia se deberán utilizar una serie de bibliotecas de Python (véase figura 3-37) que participarán en el procesamiento del lenguaje natural y en la optimización del modelo, el cual será cuantificado en 4 bits como se hizo en el ajuste. Además, necesitaremos conectar el proceso a Hugging Face para que tenga acceso a los archivos generados en el ajuste.

```

1 !pip install -q -U bitsandbytes
2 !pip install -q -U transformers
3 !pip install -q -U accelerate
4 !pip install -q -U einops
5 !pip install -q -U safetensors
6 !pip install -q -U torch
7 !pip install -q -U xformers

```

Figura 3-37 Bibliotecas usadas en la inferencia del modelo (Autoría propia).

A continuación, se detallará la función de cada biblioteca:

- Bitsandbytes: esta biblioteca se utiliza para la cuantificación de modelos, permitiendo ejecutarlos con menor precisión, pero reduciendo su consumo de memoria y acelerando la velocidad de inferencia [100].
- Transformers y Xformers: estas bibliotecas permiten trabajar con modelos de lenguaje preentrenados y realizar tareas de procesamiento del lenguaje natural como la conversión de texto a *token*, ajuste fino e inferencia de modelos [95] y [96].
- Accelerate: es una biblioteca que proporciona herramientas para acelerar el entrenamiento y la inferencia de modelos [104].
- Einops: esta biblioteca facilita la manipulación y transformación de tensores (estructuras de datos) en Python [105].
- Safetensors: esta biblioteca ayuda a garantizar la seguridad y la integridad de los cálculos numéricos en tensores de Python [106].
- Torch: es una biblioteca que contiene funciones para crear y entrenar modelos de aprendizaje automático [107].

A continuación (véase figura 3-38), veremos un ejemplo de funcionamiento de la herramienta analizando el perfil de la Marina Militare en YouTube:

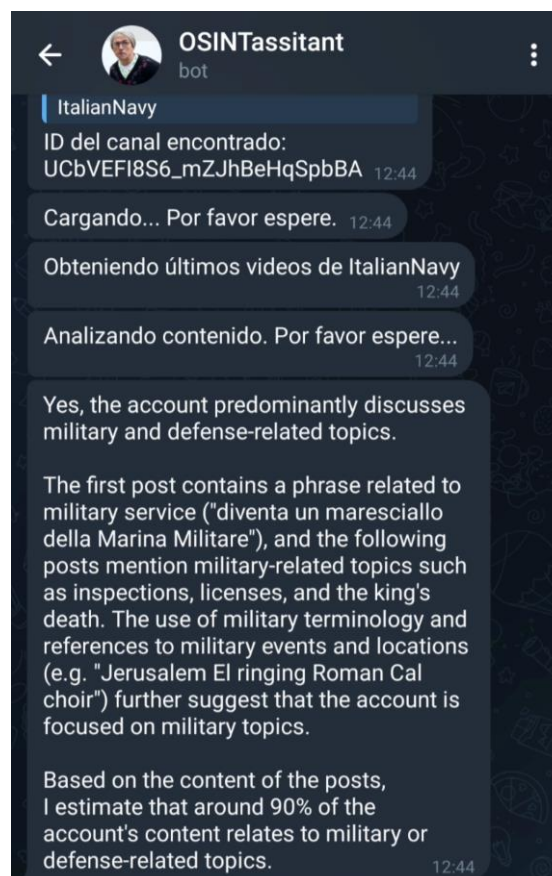


Figura 3-38 Ejemplo de análisis de Marina Militare en YouTube (Autoría propia).

4 PRUEBA Y VALIDACIÓN

En el presente capítulo se van a realizar una serie de pruebas en el flujo del código, así como en la efectividad y desempeño del analizador.

En primer lugar, se hará un análisis de la interfaz de Telegram para evaluar que es totalmente funcional y no presenta errores de programación o situaciones anómalas no contempladas en el código. En la segunda fase, se probará a extraer las últimas publicaciones de 10 perfiles de Instagram y 10 perfiles de YouTube para observar el comportamiento del módulo de captura de datos. Por último, se analizarán 40 perfiles de Instagram y sus correspondientes análogos de YouTube, de los cuales 20 perfiles estarán relacionados con defensa y 20 no. Por otro lado, se introducirán 40 documentos en formato txt de los cuales cada uno contendrá una recopilación de 100 tuits publicados por distintos perfiles de X, al igual que con las demás plataformas 20 de ellos contendrá contenido relacionado con defensa y 20 no. Esta prueba se realizará con el modelo original y con el modelo ajustado con el fin de comparar los resultados y determinar si dicho ajuste ha sido efectivo.

Para determinar si una cuenta contiene información de interés para la defensa partiremos de la definición del diccionario panegírico del español jurídico (DPEJ) que concibe la defensa nacional como la “disposición, integración y acción coordinada de todas las energías y fuerzas morales y materiales de España ante cualquier forma de agresión” así como la “salvaguarda de la soberanía y de los intereses nacionales” (véase figura 4-1) [108].

defensa nacional

1. *Gral.*; *Esp.* Disposición, integración y acción coordinada de todas las energías y fuerzas morales y materiales de España ante cualquier forma de agresión.

Figura 4-1 Definición de defensa nacional según DPEJ (Autoría propia).

A partir de esta definición, podemos ampliar los ámbitos de interés en materia de defensa, abarcando no solo las actividades directamente vinculadas con las Fuerzas Armadas y el conflicto armado, sino también otras esferas estrechamente ligadas a la seguridad nacional. Esto incluye la investigación y desarrollo de tecnologías avanzadas, la garantía de suministros de materias primas estratégicas, el fortalecimiento de la industria y la protección del territorio. Además, se consideran relevantes todos los escenarios potenciales de conflicto o acciones llevadas a cabo por agentes externos, tales como otros estados, grupos independientes, movimientos político-sociales, empresas internacionales y organizaciones no gubernamentales.

4.1 Prueba del flujo de código.

En primer lugar, se va a realizar una prueba de ejecución para verificar que todos los comandos funcionan correctamente y generan las respuestas y procesos esperados. Para ello se introducirán los comandos implementados con el fin de observar que el *chatbot* devuelve la respuesta esperada y realiza el proceso correspondiente, en este caso con el usuario vicesat.

Proceso	Prueba	Resultado	Observaciones
start	Funcionamiento	Positivo	NQR
info	Funcionamiento	Positivo	NQR
help	Funcionamiento	Positivo	NQR
YouTube	Funcionamiento	Positivo	NQR
Instagram	Funcionamiento	Positivo	NQR
Doc	Funcionamiento	Positivo	NQR

Tabla 4-1 Prueba de funcionamiento de comandos (Autoría propia).

Como podemos observar en la tabla 4-1 todos los comandos funcionen con normalidad y derivan en los mensajes y procesos que se desean ejecutar (véase figuras 4-1, 4-2, 4-3 y 4-4).

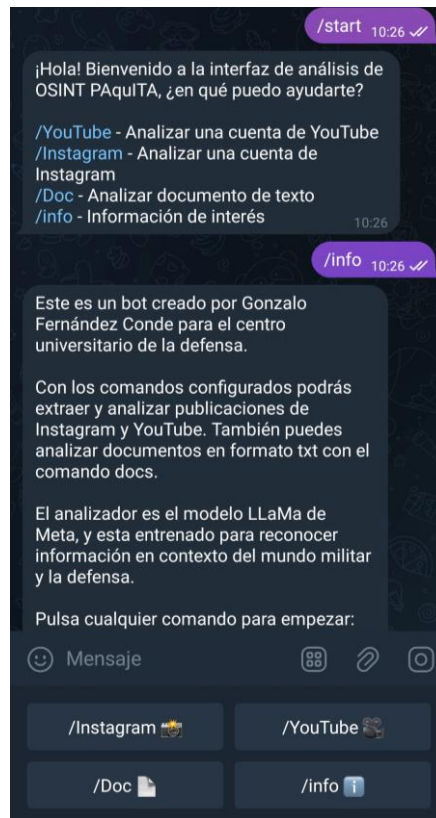


Figura 4-2 Funcionamiento comando “start” y comando “info” (Autoría propia).

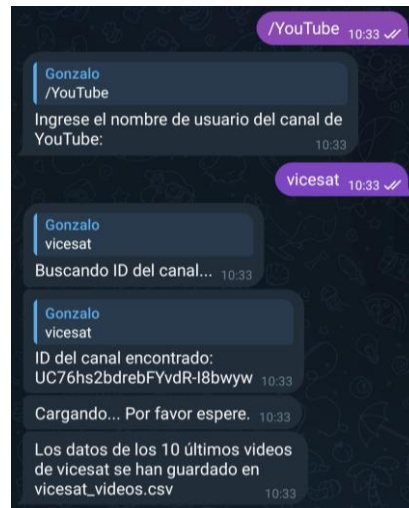


Figura 4-3 Funcionamiento proceso YouTube (Autoría propia).



Figura 4-4 Funcionamiento proceso Instagram (Autoría propia).



Figura 4-5 Funcionamiento proceso de análisis de documentos (Autoría propia).

4.2 Prueba del capturador de datos.

Una vez observado que los comandos funcionan correctamente y reciben las entradas que proporciona el usuario mediante el chat de Telegram, continuaremos estudiando la eficiencia de cada proceso de captura de datos. Proporcionaremos al bot 10 usuarios aleatorios de YouTube e Instagram para analizar si se producen anomalías en el proceso y observar los tiempos de ejecución.

Proceso	Prueba	Resultado	Canal	Tiempo (s)	Archivo	Observaciones
YouTube	Usuario existente	Positivo	vicesat	9,69	196 kB	NQR
YouTube	Usuario existente	Positivo	CBCTheNational	8,69	62 kB	NQR
YouTube	Usuario existente	Positivo	BBCNews	8,81	29 kB	NQR
YouTube	Usuario existente	Positivo	ArmadaEspanola	6,56	107 kB	Solicita ID
YouTube	Usuario existente	Positivo	usarmy	7,87	12 kB	NQR
YouTube	Usuario existente	Positivo	TODAY	8,69	15 kB	Solicita ID
YouTube	Usuario existente	Positivo	telegraph	8,00	324 kB	Solicita ID
YouTube	Usuario existente	Positivo	RFU	7,75	60 kB	Solicita ID
YouTube	Usuario existente	Positivo	elpais	8,06	7 kB	Solicita ID
YouTube	Usuario existente	Positivo	thewillyrex	7,62	119 kB	NQR
Estadísticas	-	100%	-	8,17	93,1 kB	50% Solicita ID

Tabla 4-2 Análisis de raspado de YouTube (Autoría propia).

Como se puede observar en la tabla 4-2 se han introducido 10 cuentas de YouTube escogidas aleatoriamente. En este ensayo veremos el comportamiento de la herramienta ante la entrada de cuentas de usuario existentes. Analizando los resultados observamos que el proceso ha resultado ser eficaz el 100% de las veces llegando a obtener un archivo con las transcripciones de vídeos publicados por dichos usuarios.

Por otro lado, observamos que los tiempos de ejecución son muy similares entre sí obteniendo una media de tiempo de procesamiento de 8,17 s. Mientras tanto en función de los tamaños de archivo generado deducimos una media de 93 kB de tamaño de archivo solución. Según estos dos últimos datos podemos obtener la tasa media de captura de datos de YouTube que en este caso resultaría de 11,38 kB/s.

En cuanto a observaciones es destacable que, inexplicablemente, la obtención del ID mediante el nombre de usuario falla el aproximadamente el 50% de las veces por lo que en 5 de cada 10 casos deberemos introducir el ID del canal manualmente.

Por lo demás el proceso de raspado es efectivo y tiene una velocidad de transcripción más que aceptable para el uso de la herramienta. Sin embargo, debe estudiarse todavía su comportamiento en unión con el analizador ya que estos tiempos se han calculado únicamente hasta el momento de obtención del archivo de transcripción en formato CSV.

Proceso	Prueba	Resultado	Canal	Tiempo (s)	Archivo	Observaciones
Instagram	Usuario existente	Positivo	vicesat_yt	16,56	6 kB	NQR
Instagram	Usuario existente	Positivo	cbcthenational	9,32	1 kB	NQR
Instagram	Usuario existente	Positivo	bbcnews	13,44	11 kB	NQR
Instagram	Usuario existente	Positivo	armadaesp	8,41	7 kB	NQR
Instagram	Usuario existente	Positivo	usarmy	9,94	1 kB	NQR
Instagram	Usuario existente	Positivo	todayshow	9,00	7 kB	NQR
Instagram	Usuario existente	Positivo	telegraph	14,44	8 kB	NQR
Instagram	Usuario existente	Positivo	RPFUkraine	6,19	1 kB	NQR
Instagram	Usuario existente	Positivo	el_pais	12,87	18 kB	NQR
Instagram	Usuario existente	Positivo	willyrex	4,63	1 kB	NQR
Estadísticas	-	100%	-	10,48		NQR

Tabla 4-3 Análisis de raspado de Instagram (Autoría propia).

Al igual que en el ejemplo anterior se han introducido 10 cuentas de Instagram en la herramienta obteniéndose los resultados reflejados en la Tabla 4-3. Como puede observarse, se obtiene de nuevo un 100% de efectividad introduciendo nombres de usuario existentes extrayendo en todos los casos un archivo CSV con los pies de foto de las 20 últimas publicaciones del usuario introducido. Cabe destacar que las cuentas introducidas son análogas del ejemplo anterior respecto a esta plataforma.

Respecto a los tiempos de procesamiento observados, se obtienen unos resultados más variables con valores comprendidos entre 4 y 16 segundos. Se ha calculado una media de tiempo de procesamiento de 10,48 s junto con una mediana de 9,63 s. Este fenómeno se debe a la variedad de tamaño de los archivos obtenidos ya que en función de la cantidad de texto que escriba cada cuenta en sus publicaciones aumentará el tamaño del archivo y por lo tanto también lo hará el tiempo de procesamiento. En función a estos datos se calcula una tasa media de captura de datos de 0,58 kB/s.

Proceso	Prueba	Resultados	Observaciones
General	Introducir comando desconocido	Neutral	El bot no interactúa ante comandos desconocidos
General	Introducir doble comando	Positivo	El bot no permite introducir comandos durante un proceso en curso
General	Dos o más usuarios simultáneos	Negativo	Los comandos y procesos de ambos usuarios interfieren entre sí
YouTube	Usuario no existente	Positivo	Se indica que hay un error en el ID del canal
Instagram	Credenciales inválidas	Positivo	Se indica el mensaje “credenciales inválidas”
Instagram	Usuario no existente	Positivo	Se indica al usuario que se trata de un perfil inexistente

Tabla 4-4 Análisis de errores de interacción (Autoría propia).

En la tabla 4-4 se encuentran registradas las entradas del usuario que podrían causar errores, bugs o anomalías en los procesos configurados, así como entradas de usuario no existentes. Previamente a esta fase ya se habían tenido en cuenta por lo que el código cuenta con protocolos para mitigarlos. La herramienta esta depurada y no presenta fallos importantes de implementación, sin embargo, al no disponer de un servidor capaz de mantener sesiones de usuario independiente la interfaz no contará con la posibilidad de interactuar con más de un usuario.

4.3 Prueba del analizador.

A continuación, se determinará si el ajuste ha sido efectivo introduciendo datos de perfiles de distintas plataformas tanto al modelo ajustado como al modelo original, con el objetivo de comparar la efectividad en los resultados, la confianza de la respuesta y los tiempos de ejecución. Por cada perfil publicado se estudiarán las últimas 20 publicaciones en el caso de Instagram, el último vídeo publicado en el caso de YouTube y los últimos 100 tuits en el caso de X, escogiendo estos números en función del número de *tokens* aproximado que proporciona cada publicación. Sobre la transcripción de la/s publicación/es/video obtenido, se considerará que un perfil es de interés para la defensa si más del 50% del contenido de sus publicaciones se puede relacionar con dicho contexto.

4.3.1 Modelo sin ajustar.

En cuanto al modelo sin ajustar se han obtenido los siguientes resultados para la plataforma X/Twitter:

Contenido no relacionado con defensa			Contenido relacionado con defensa		
Perfil	Resultado	Tiempo (s)	Perfil	Resultado	Tiempo (s)
BBC World	Falso positivo	30,2	Aviation Intel	Positivo	29,3
CFR org	Falso positivo	30,8	Covert shores	Falso negativo	30,1
Elon Musk	Falso positivo	30,7	Defense U	Positivo	33,6
EPW IN	Negativo	38,7	Dr. Radchenko	Positivo	31
ESPN	Falso positivo	31,5	ELINT News	Positivo	31,1
EWT News	Falso positivo	28,9	Kyiv independent	Positivo	31,1
FIFA	Negativo	31,8	Military Landnet	Positivo	60,1
Greenpeace	Negativo	29,6	NATO	Positivo	31,4
NASA	Falso positivo	31,5	NOEL Reports	Positivo	88,6
NatGeo	Falso positivo	36,4	War Monitor	Positivo	49,5
NBA	Negativo	31,5	OSINT Technical	Positivo	28,4
NY Times	Negativo	44,8	Phillips P Obrien	Positivo	31,3
Science News	Falso positivo	88,9	Rebel44CZ	Falso negativo	33,9
SCMP News	Falso positivo	51,1	Sentdefender	Positivo	89,5
The Print India	Negativo	93	TDF UA	Positivo	35,2
United Nations	Falso positivo	86,2	The study of war	Positivo	30,7

Contenido no relacionado con defensa			Contenido relacionado con defensa		
Perfil	Resultado	Tiempo (s)	Perfil	Resultado	Tiempo (s)
UNICEF	Falso positivo	31,4	Thewarzonewire	Positivo	30,6
WSJ	Negativo	31,7	UAWeapons	Positivo	29,8
YouTube	Falso positivo	91,3	US Air Force	Positivo	31,4
Zara	Falso positivo	28,7	War Institute	Positivo	31,2

Tabla 4-5 Evaluación de respuestas del modelo sin ajustar para la plataforma X (Autoría propia).

Como se observa en la tabla 4-5, la proporción de falsos positivos es extremadamente alta, esto se debe en gran parte al contenido de perfiles que publican noticias e información de conflictos y política internacional la cual puede relacionarse con el contexto de defensa. Por otro lado, determinadas abreviaturas, fechas y palabras clave (ej. campaña) aumentan la probabilidad de obtener falso positivo.

Al analizar los resultados, observamos que la eficacia del modelo en el análisis de perfiles de la plataforma X no relacionados con defensa es del 35%, con un tiempo medio de procesamiento de 44,90 segundos. En cambio, la eficacia en el análisis de contenido relacionado con defensa asciende al 90%, con un tiempo medio de procesamiento de 37,50 segundos. Respecto a los resultados erróneos se ha obtenido un 65% de falsos positivos y un 10% de falsos negativos.

Contenido no relacionado con defensa			Contenido relacionado con defensa		
Perfil	Resultado	Tiempo (s)	Perfil	Resultado	Tiempo (s)
BBC World	Negativo	30,2	Armadaesp	Positivo	45
CFR org	Falso positivo	30,8	Usarmy	Falso negativo	10,4
Elon Musk	Falso positivo	30,7	Navy.seal.swcc	Positivo	31,2
EPW IN	Falso positivo	38,7	Militarydotcom	Positivo	19,3
ESPN	Negativo	31,5	Aljazeeraenglish	Positivo	62,7
EWT News	Falso positivo	28,9	Military	Positivo	42,19
FIFA	Negativo	31,8	Americanmilitarynews	Positivo	16,7
Greenpeace	Negativo	29,6	NATO	Positivo	27,3
NASA	Falso positivo	31,5	Singaporenavy	Positivo	19,5
NatGeo	Negativo	36,4	Militarytimes	Positivo	80,1
NBA	Negativo	31,5	Oficialearm	Positivo	40,2
NY Times	Negativo	44,8	Marinamilitareofficiale	Positivo	23,9
Science News	Falso positivo	88,9	Idf	Positivo	58,6
SCMP News	Falso positivo	51,1	Saf.bmtc	Positivo	52,7
The Print India	Falso positivo	93	2020_soldier	Positivo	47,4
United Nations	Negativo	86,2	The study of war	Positivo	40,1
UNICEF	Falso positivo	31,4	Deptofdefense	Positivo	14,8
WSJ	Negativo	31,7	Internationallegionua	Positivo	18,6
YouTube	Negativo	91,3	Usaf_recuiting	Positivo	35,2

Contenido no relacionado con defensa			Contenido relacionado con defensa		
Perfil	Resultado	Tiempo (s)	Perfil	Resultado	Tiempo (s)
Zara	Falso positivo	28,7	Armapedia	Positivo	80,3

Tabla 4-6 Evaluación de respuestas del modelo sin ajustar para la plataforma Instagram (Autoría propia).

La tabla 4-6 muestra los resultados para el caso de Instagram donde se puede ver que se obtiene una eficacia de detección del 50% para perfiles no relacionados con defensa y un 95% para perfiles relacionados. En cuanto errores de detección se obtienen un 50% de falsos positivos y un 5% de falsos negativos unos resultados algo mejores que en el análisis de X, pero con una tendencia similar. En cuanto a los tiempos de procesamiento obtenemos una media de 43,36 segundos para perfiles no relacionados y 37,55 segundos para perfiles relacionados.

Contenido no relacionado con defensa			Contenido relacionado con defensa		
Perfil	Resultado	Tiempo (s)	Perfil	Resultado	Tiempo (s)
BBC World	Falso positivo	58,1	Armadaesp	Falso negativo	96
CFR org	Falso positivo	4	Usarmy	Positivo	5,3
Elon Musk	Falso positivo	58,3	Navy.seal.swcc	Positivo	4,8
EPW IN	Falso positivo	6,8	Militarydotcom	Falso negativo	61,2
ESPN	Falso positivo	35,3	Aljazeeraenglish	Positivo	26,4
EWT News	Falso positivo	14,2	Military	Positivo	16,1
FIFA	Negativo	90	Americanmilitarynews	Positivo	35,5
Greenpeace	Falso positivo	2,5	NATO	Positivo	19,14
NASA	Negativo	9,3	Singaporenavy	Positivo	8,1
NatGeo	Falso positivo	6,1	Militarytimes	Positivo	7,9
NBA	Negativo	10,6	Oficialearn	Positivo	9,6
NY Times	Negativo	11,6	Marinamilitareofficiale	Falso negativo	14,6
Science News	Negativo	12,3	Idf	Positivo	11,7
SCMP News	Falso positivo	9	Saf.bmtc	Positivo	7
The Print India	Falso positivo	7,3	2020_soldier	Falso negativo	48,9
United Nations	Negativo	5,4	The study of war	Positivo	42,3
UNICEF	Negativo	31,2	Deptofdefense	Falso negativo	12,3
WSJ	Falso positivo	13,2	Internationallegionua	Positivo	24,5
YouTube	Negativo	4,3	Usaf_recuiting	Positivo	6,23
Zara	Negativo	22,12	Armapedia	Falso negativo	14,3

Tabla 4-7 Evaluación de respuestas del modelo sin ajustar para la plataforma YouTube (Autoría propia).

Como se puede observar en la tabla 4-7, para la plataforma YouTube, se obtiene una eficacia de detección de 45% para canales no relacionados con defensa proporcionando un 55% de falsos positivos,

y una eficacia del 70% con un 30% de falsos negativos para canales relacionados con la defensa. Los tiempos de ejecución son notablemente más sesgados ya que dependerán directamente de la duración del vídeo que se analice. Su media resulta en 22,10 segundos (no defensa) y 24,08 segundos (defensa). Por otro lado, vemos un mayor número de falsos negativos debido a la redundancia de información ya que en el resto de plataformas las publicaciones son más directas y concisas.

4.3.2 Modelo ajustado.

Para el modelo ajustado, se han introducido los mismos textos pertenecientes a los mismos perfiles estudiados para el modelo sin ajustar. Se han obtenido los siguientes resultados.

Contenido no relacionado con defensa			Contenido relacionado con defensa		
Perfil	Resultado	Tiempo (s)	Perfil	Resultado	Tiempo (s)
BBC World	Negativo	27,7	Aviation Intel	Positivo	27,3
CFR org	Falso positivo	27,9	Covert shores	Positivo	19,4
Elon Musk	Negativo	27,7	Defense U	Positivo	27,1
EPW IN	Falso positivo	38,1	Dr. Radchenko	Positivo	27,8
ESPN	Negativo	28,7	ELINT News	Positivo	57,7
EWT News	Negativo	36,1	Kyiv independent	Positivo	27
FIFA	Negativo	27,8	Military Landnet	Positivo	42,2
Greenpeace	Negativo	28,4	NATO	Positivo	36,9
NASA	Negativo	26	NOEL Reports	Positivo	28,5
NatGeo	Negativo	36,3	War Monitor	Positivo	31,2
NBA	Negativo	56,8	OSINT Technical	Positivo	27,2
NY Times	Falso positivo	31	Phillips P Obrien	Positivo	27,8
Science News	Negativo	43,1	Rebel44CZ	Positivo	31,1
SCMP News	Negativo	41	Sentdefender	Positivo	27
The Print India	Falso positivo	45,1	TDF UA	Positivo	26,4
United Nations	Negativo	34,9	The study of war	Positivo	32,1
UNICEF	Negativo	26	Thewarzonewire	Positivo	30
WSJ	Falso positivo	36,4	UAWapons	Positivo	22,8
YouTube	Negativo	34,4	US Air Force	Positivo	27,5
Zara	Negativo	28,5	War Institute	Positivo	23,4

Tabla 4-8 Evaluación de respuestas del modelo ajustado para la plataforma X (Autoría propia).

Como se puede observar en la tabla 4-8, analizando la plataforma X con el modelo ajustado, obtenemos un 75% de efectividad para perfiles no relacionados con defensa frente a un 100% para perfiles relacionados. Es destacable que no se obtienen falsos negativos, sin embargo, se mantiene un porcentaje de falsos positivos del 25% debido al contenido de perfiles que publican informativos sobre conflictos y política internacional. El tiempo medio de ejecución se reduce a 34,09 segundos para perfiles no relacionados y 30,02 segundos para perfiles de interés para la defensa.

Contenido no relacionado con defensa			Contenido relacionado con defensa		
Perfil	Resultado	Tiempo (s)	Perfil	Resultado	Tiempo (s)
BBC World	Negativo	17,2	Armadaesp	Positivo	49,3
CFR org	Negativo	98,6	Usarmy	Positivo	9,3
Elon Musk	Negativo	8,5	Navy.seal.swcc	Positivo	46,5
EPW IN	Negativo	53,1	Militarydotcom	Positivo	16,6
ESPN	Negativo	16,3	Aljazeeraenglish	Positivo	23,2
EWT News	Negativo	35,6	Military	Positivo	42,6
FIFA	Negativo	17,1	Americanmilitarynews	Positivo	12,9
Greenpeace	Negativo	30,1	NATO	Positivo	11,4
NASA	Negativo	29,43	Singaporenavy	Positivo	27,3
NatGeo	Negativo	37,95	Militarytimes	Positivo	35,93
NBA	Negativo	11,8	Oficialearn	Positivo	43,24
NY Times	Negativo	28,8	Marinamilitareofficiale	Positivo	16
Science News	Negativo	38,8	Idf	Positivo	32,9
SCMP News	Negativo	38,8	Saf.bmtc	Positivo	40,3
The Print India	Falso positivo	37,1	2020_soldier	Positivo	38,2
United Nations	Falso positivo	53,7	The study of war	Positivo	54,2
UNICEF	Negativo	53,2	Deptofdefense	Positivo	29
WSJ	Negativo	55,38	Internationallegionua	Positivo	16,13
YouTube	Negativo	51,2	Usaf_recuiting	Positivo	29,34
Zara	Negativo	48,21	Armapedia	Positivo	27,6

Tabla 4-9 Evaluación de respuestas del modelo ajustado para la plataforma Instagram (Autoría propia).

Como se puede observar en la tabla 4-9, analizando la plataforma Instagram, el número de falsos positivos se reduce al 10% por lo que se obtiene una efectividad del 90% para el análisis de perfiles no relacionados con la defensa. Para el caso de perfiles relacionados, se obtiene una eficacia del 100% de nuevo sin presencia de falsos negativos. El tiempo medio de ejecución del proceso es de 38,01 segundos para perfiles no relacionados con defensa y 30,97 para perfiles relacionados.

Contenido no relacionado con defensa			Contenido relacionado con defensa		
Perfil	Resultado	Tiempo (s)	Perfil	Resultado	Tiempo (s)
BBC World	Negativo	11,3	Armadaesp	Positivo	28,8
CFR org	Negativo	13,9	Usarmy	Positivo	29,2
Elon Musk	Negativo	7,1	Navy.seal.swcc	Positivo	15,26
EPW IN	Falso positivo	8,1	Militarydotcom	Positivo	9,1
ESPN	Negativo	15,9	Aljazeeraenglish	Positivo	11,8

EWT News	Negativo	9,4	Military	Positivo	17,9
FIFA	Negativo	19,4	Americanmilitarynews	Positivo	27,2
Greenpeace	Negativo	12,3	NATO	Positivo	13,4
NASA	Falso positivo	9,81	Singaporenavy	Positivo	31,6
NatGeo	Negativo	6,13	Militarytimes	Positivo	10,2
NBA	Negativo	9,3	Oficialearm	Positivo	16,4
NY Times	Negativo	11,9	Marinamilitareofficiale	Positivo	14
Science News	Negativo	12,4	Idf	Positivo	14,78
SCMP News	Negativo	10	Saf.bmtc	Positivo	10,26
The Print India	Falso positivo	10,2	2020_soldier	Positivo	23,1
United Nations	Negativo	20,6	The study of war	Positivo	9,6
UNICEF	Falso positivo	26,6	Deptofdefense	Positivo	12,2
WSJ	Falso positivo	20,32	Internationallegionua	Positivo	10
YouTube	Negativo	17,23	Usaf_recuiting	Positivo	15,4
Zara	Negativo	8,23	Armapedia	Positivo	12,5

Tabla 4-10 Evaluación de respuestas del modelo ajustado para la plataforma YouTube (Autoría propia).

Como se observa en la tabla 4-10, analizando la plataforma YouTube, las respuestas del modelo ajustado son el 75% casos correctos para perfiles no relacionados con la defensa y el 100% de las veces para perfiles relacionados con defensa. En cuanto a errores en la detección se producen un 25% de falsos positivos y al igual que en los casos anteriores no se presenta ningún falso negativo. Los tiempos medios de procesamiento son de 13,43 segundos para publicaciones no relacionadas con defensa y 16,63 para publicaciones relacionadas.

4.3.3 Evaluación general.

Para la evaluación general del ajuste utilizaremos una matriz de confusión como la que se puede observar en la figura 4-6, así como sus métricas, unas herramientas fundamentales para medir el rendimiento y monitorear la eficacia de un modelo. Estas métricas ofrecen una visión detallada sobre la calidad de las predicciones del modelo, incluyendo la exactitud (cercanía de la respuesta a la respuesta esperada), la precisión (dispersión del conjunto de respuestas), la sensibilidad (proporción de perfiles relacionados con defensa identificados correctamente) y la especificidad (proporción de perfiles no relacionados con defensa identificados correctamente). Se pueden proporcionar múltiples métricas en forma de lista, permitiendo al modelo calcular y mostrar estos valores a lo largo del proceso de entrenamiento [109].



Figura 4-6 Ejemplo de matriz de confusión [109].

Para calcular estas métricas, se emplean las siguientes ecuaciones, donde T_P representa el número de verdaderos positivos, T_N representa el número de verdaderos negativos, F_P los falsos positivos (casos clasificados incorrectamente como relacionados con defensa) y F_N los falsos negativos (casos incorrectamente clasificados como no relacionados con defensa) [109]:

$$Exactitud = \frac{T_P + T_N}{T_P + T_N + F_P + F_N}$$

$$Precisión = \frac{T_P}{T_P + F_P}$$

$$Sensibilidad = \frac{T_P}{T_P + F_N}$$

$$Especificidad = \frac{T_N}{T_N + F_P}$$

A lo que, para el modelo sin ajustar se obtienen los siguientes resultados:

MODELO SIN AJUSTAR RESPUESTA OBTENIDA	RESPUESTA ESPERADA	
	DEFENSA	NO DEFENSA
DEFENSA	85,00%	56,67%
NO DEFENSA	15,00%	43,33%

Tabla 4-11 Matriz de confusión obtenida para el modelo sin ajustar (Autoría propia).

Habiendo analizado 60 cuentas relacionadas con defensa y 60 cuentas no relacionadas con defensa, se obtiene una exactitud de 0,64, una precisión de 0,60, una sensibilidad de 0,85 y una especificidad de 0,43.

En cuanto al modelo ajustado se obtienen los siguientes resultados:

MODELO AJUSTADO RESPUESTA OBTENIDA	RESPUESTA ESPERADA	
	DEFENSA	NO DEFENSA
DEFENSA	100,00%	20,00%
NO DEFENSA	0,00%	80,00%

Tabla 4-12 Matriz de confusión obtenida para el modelo ajustado (Autoría propia).

Habiendo analizado las mismas cuentas que en el caso anterior, obtenemos una exactitud de 0,90, una precisión de 0,83, una sensibilidad de 1,00 y una especificidad de 0,80, apreciándose unos resultados notablemente mejores después del ajuste.

4.3.4 Conclusión de las pruebas y resultados

En la vista de los resultados de las pruebas podemos determinar los siguientes postulados:

- La eficacia del análisis mejora significativamente tras el ajuste del modelo, obteniendo una sensibilidad media 15% mayor (véase figura 4-7) y una especificidad media 36,67% mayor (véase figura 4-8).

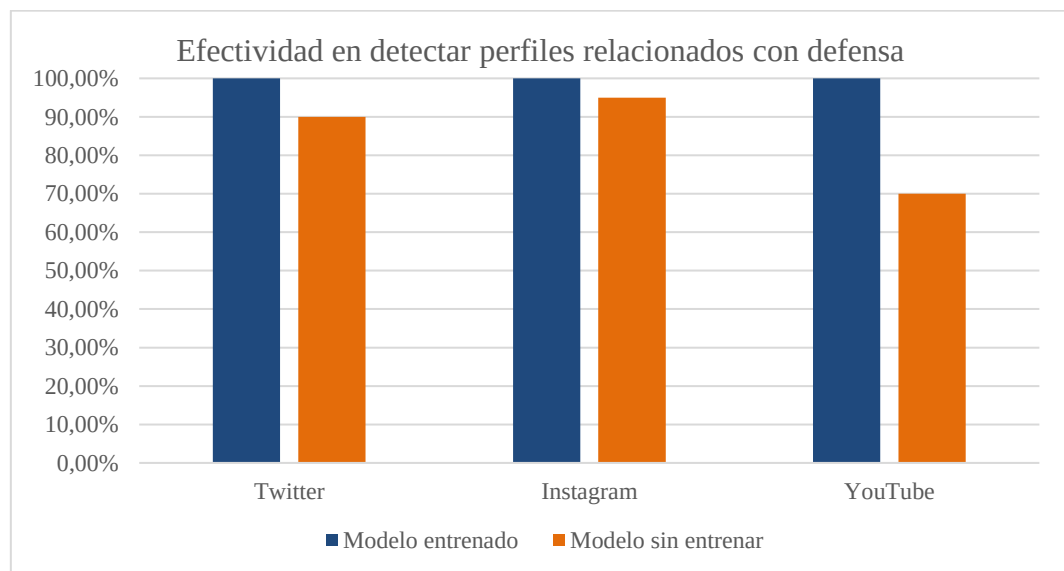


Figura 4-7 Gráfico comparativo de sensibilidad (Autoría propia).

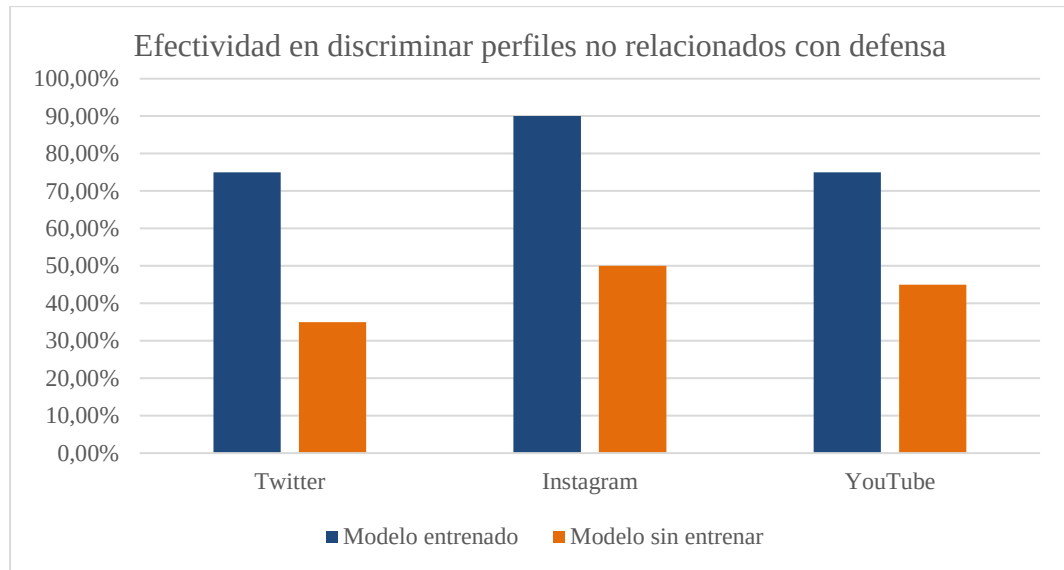


Figura 4-8 Gráfico comparativo de especificidad (Autoría propia).

- El número de errores en el análisis disminuye significativamente tras el ajuste obteniendo un 36,67% menos de falsos positivos (véase figura 4-9) y reduciendo a 0 el número de falsos negativos, un 15% menos que los obtenidos en el modelo sin ajustar (véase figura 4-10). Cabe destacar que el número de falsos positivos está directamente relacionado con el carácter del contenido publicado por los perfiles, por lo que un perfil que publica contenido relacionado con conflictos internacionales, acciones en países en guerra o crisis humanitarias, puede ser clasificado como relacionable a la defensa sin serlo directamente.

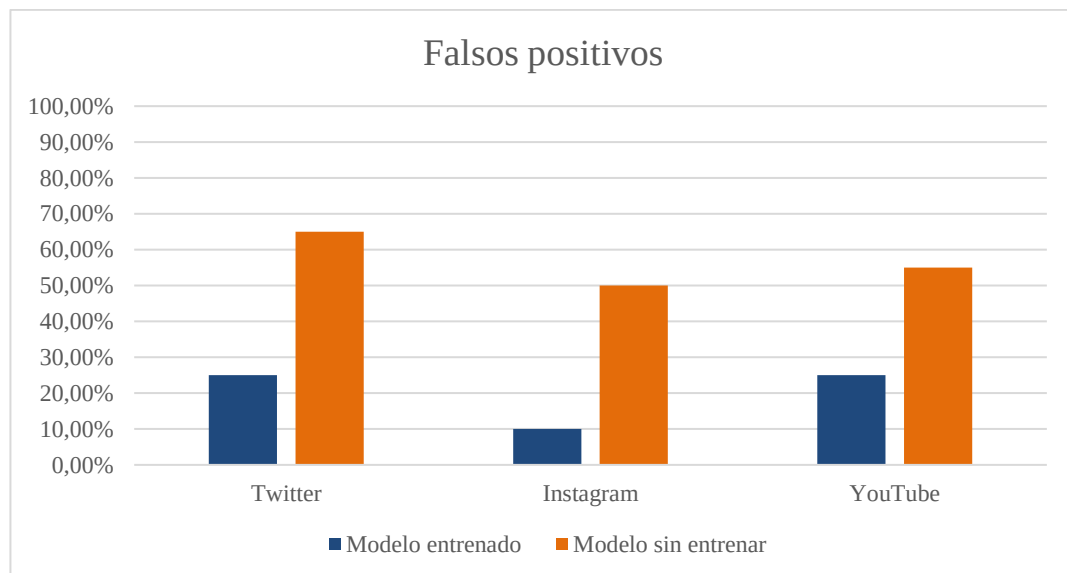


Figura 4-9 Gráfico comparativo de falsos positivos (Autoría propia).

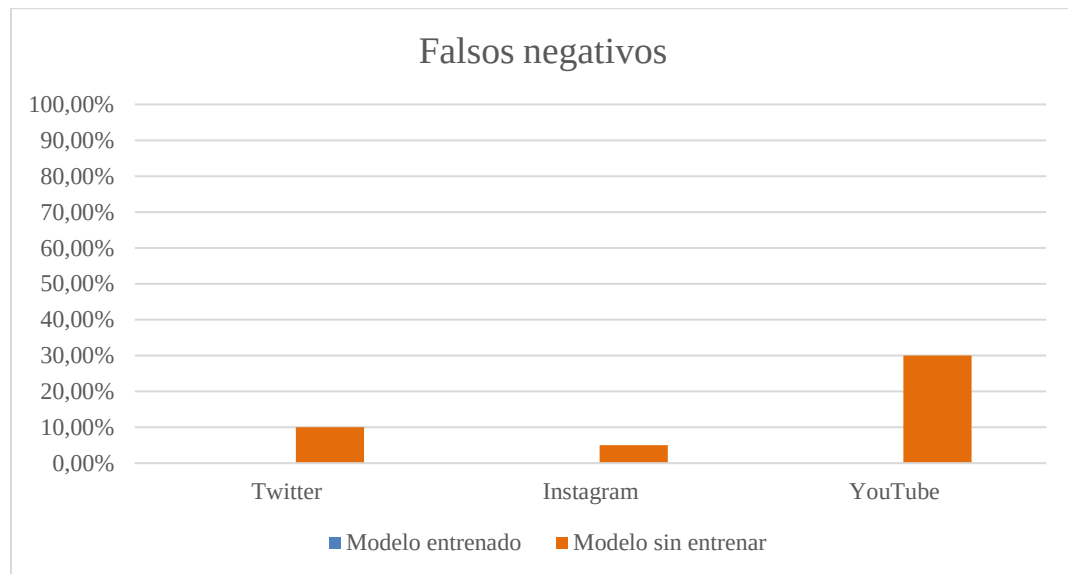


Figura 4-10 Gráfico comparativo de falsos negativos (Autoría propia).

- Los tiempos de procesamiento del análisis se reducen significativamente en el modelo ajustado, pasando de un tiempo medio de $33,68 \pm 3,92$ segundos a $25,58 \pm 2,10$ segundos en el caso de perfiles relacionados con defensa y de $36,81 \pm 4,76$ segundos a $28,51 \pm 3,04$ segundos en el caso de perfiles no relacionados con defensa. Esto se debe por una parte al ajuste del modelo para la clasificación y por otra a la cuantización en 4 bits que reduce el tamaño del modelo.

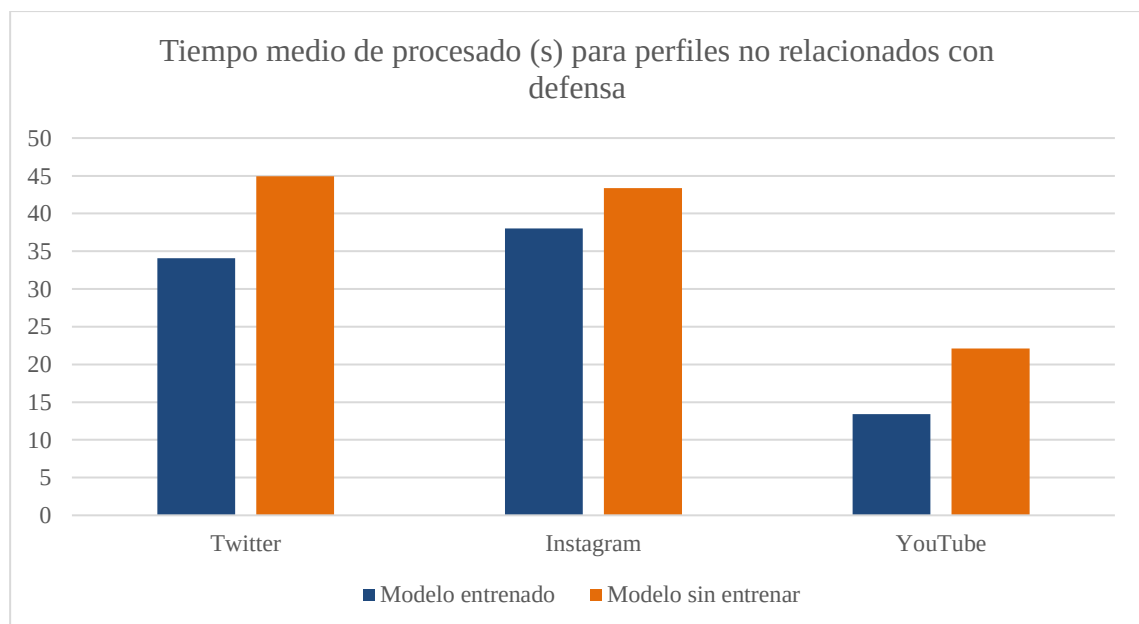


Figura 4-11 Gráfico comparativo de tiempos de procesado para perfiles no relacionados con defensa (Autoría propia).

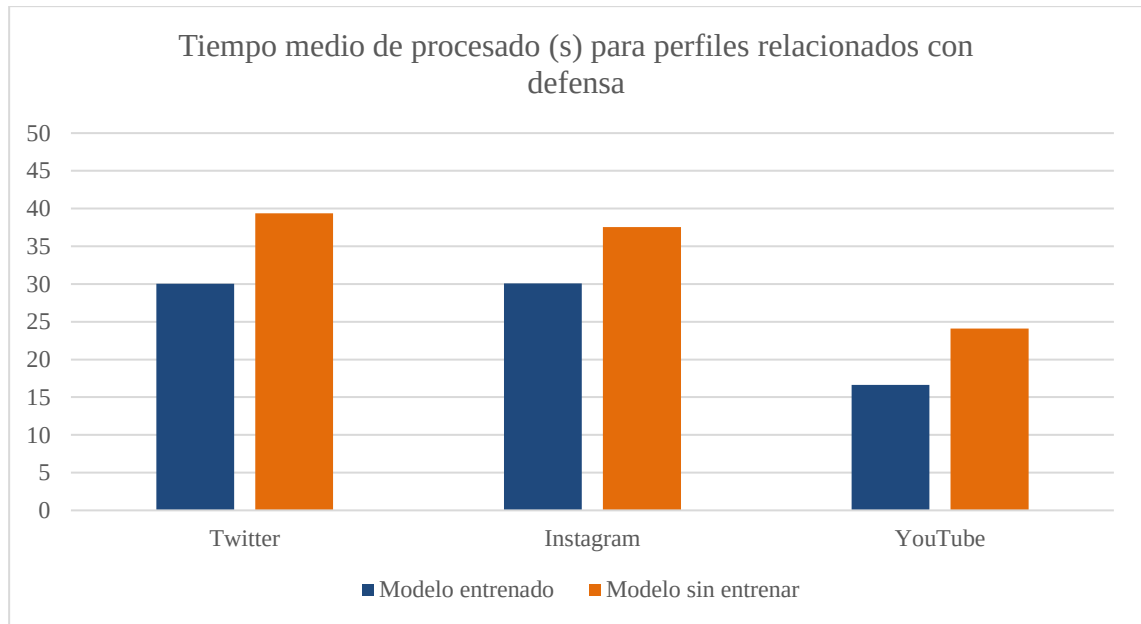


Figura 4-12 Gráfico comparativo de tiempos de procesamiento para perfiles relacionados con defensa (Autoría propia).

5 CONCLUSIONES Y LÍNEAS FUTURAS

5.1 Conclusiones.

En el contexto actual, caracterizado por la rápida evolución de la tecnología y la creciente interconexión digital, el uso de inteligencia de fuentes abiertas se ha convertido en un componente esencial para la comprensión y el análisis de eventos y tendencias a nivel global. Las redes sociales, en particular, han emergido como una valiosa fuente de información, ofreciendo una ventana única hacia las opiniones, actividades y comportamientos de individuos y comunidades en todo el mundo. El papel de las redes sociales en la generación y difusión de contenido relevante no puede subestimarse destacando plataformas como Instagram, X y YouTube no solo en la tarea de facilitar la interacción y el intercambio de ideas entre usuarios, sino que también su uso como canales de expresión y difusión de información sobre una amplia gama de temas.

Por otro lado, la naturaleza dinámica y la amplia adopción de Telegram lo convierten en un canal interesante para la recopilación y el análisis de datos en el contexto de la inteligencia abierta. En este contexto, se ha llevado a cabo la implementación de una interfaz de usuario mediante un bot en Telegram para la herramienta de análisis de OSINT. Esta integración no solo amplía el alcance y la accesibilidad de la herramienta, sino que también aprovecha el potencial de Telegram como plataforma de interacción y colaboración en tiempo real.

En este sentido, el presente trabajo se enfoca en la importancia y la relevancia de las redes sociales, las técnicas de inteligencia artificial, el uso de Telegram como componentes clave en la recopilación, análisis y comprensión de información relacionada con temas de defensa y su aplicación en técnicas de análisis OSINT. A través del desarrollo de una herramienta basada en el uso de grandes modelos de lenguaje, con una interfaz de usuario basada en un *chatbot* de Telegram para aplicaciones OSINT, se busca aprovechar el poder de estas tecnologías para facilitar la vigilancia, el monitoreo, el análisis y la toma de decisiones estratégicas en un entorno digital en constante cambio.

5.1.1 Conclusiones generales.

En la vista general de este proyecto, podemos concluir los siguientes postulados:

- La herramienta ha resultado tener una funcionalidad robusta y razonablemente fiable. La capacidad para extraer datos de manera ágil y precisa de las plataformas, especialmente

Instagram y YouTube, facilita la interacción con dichas plataformas sin la necesidad de realizar captura de datos manual, agilizando el proceso y facilitando el análisis.

- La integración de un LLM en el análisis de los datos recopilados ha sido una elección acertada. Estos modelos han permitido un análisis del lenguaje natural utilizado en las publicaciones, lo que ha facilitado la identificación de menciones y referencias a temas relacionados con defensa y militares con un alto grado de precisión.
- Mediante el uso de ajuste fino y técnicas de aprendizaje automático, se ha realizado una adaptación más precisa de la herramienta para enfocar el LLM a identificar las características específicas del contenido relacionado con defensa y responder de la manera deseada. Este enfoque ha contribuido significativamente a la calidad y precisión de los análisis realizados, fortaleciendo así la utilidad y la fiabilidad del asistente.
- La herramienta desarrollada tiene una amplia gama de aplicaciones prácticas en diversos ámbitos, desde la seguridad nacional hasta la toma de decisiones estratégicas. Su capacidad para monitorear y analizar el discurso en línea puede ser utilizada para la anticipación de amenazas, la evaluación de la opinión pública y la planificación de estrategias de comunicación, entre otros usos.
- A pesar de los resultados obtenidos, se reconocen áreas de mejora y optimización en el proyecto. Es fundamental seguir refinando los algoritmos de análisis, así como explorar nuevas técnicas y enfoques para mejorar la precisión y la eficiencia del proceso.

5.2 Líneas futuras.

Una vez sintetizadas las conclusiones principales del proyecto, podemos determinar que se han cumplido satisfactoriamente los objetivos propuestos. Se ha logrado implementar un asistente virtual robusto y fiable con el que obtener y analizar datos publicados por perfiles de diversas plataformas de interacción social. Además, se ha conseguido ajustar el asistente para que facilite la interpretación de la información y los datos previamente obtenidos. Sin embargo, los campos tecnológicos en los que se ha trabajado para la construcción de esta herramienta se encuentran en constante crecimiento, por lo que a continuación, se mencionarán una serie de detalles para continuar desarrollando esta línea de investigación.

- El uso de LLM más potentes permitirían la obtención de una respuesta más detallada, así como mayor precisión en la detección.
- Podrían disponerse de herramientas de interpretación de imágenes, vídeo y audio para aumentar el alcance de la herramienta en plataformas de difusión de contenido multimedia.
- El uso de un entorno de trabajo con memoria de GPU limitada dificulta el uso de LLM y métodos de aprendizaje automático como el ajuste fino. Con el fin de construir analizadores más fiables sería recomendable disponer de dispositivos con alta capacidad de GPU.
- La herramienta desarrollada analiza cuentas introducidas manualmente. Para una mayor capacidad de detección y la automatización de los procesos podrían introducirse protocolos de análisis automático mediante listados de perfiles, obteniendo respuestas en forma de alarma para los perfiles de interés en vez de una conclusión por perfil analizado.

- En este caso el ajuste está enfocado en detectar contenido relacionado con defensa, sin embargo, el ajuste puede ser configurado para tareas más diversas como detectar patrones y comportamientos poco éticos en las plataformas.
- Además del uso del ajuste fino, se podrían utilizar métodos RAG (*Retrieval-Augmented Generation*). Elaborando un sistema en el que el asistente tenga acceso a fuentes de datos para mejorar el análisis y proporcionar respuestas más detalladas.
- Los términos y condiciones de uso de las redes sociales, así como las de sus API's sufren cambios constantes por lo que se deberán actualizar los procesos cuando sea requerido para mantener la herramienta en funcionamiento.

6 BIBLIOGRAFÍA

- [1] «La Operación “Mincemeat” en la Segunda Guerra Mundial», National Geographic Accedido: 30 de enero de 2024. [En línea]. Disponible en: https://historia.nationalgeographic.com.es/a/operacion-mincemeat-segunda-guerra-mundial_15257
- [2] «Breve historia del espionaje». J. C. H. Hermosilla. Nowtilus, 2012, ISBN: 9788413053042.
- [3] «Diez de los espías más famosos de la historia», Diario ABC. Accedido: 30 de enero de 2024. [En línea]. Disponible en: https://www.abc.es/internacional/abci-diez-espias-mas-famosos-historia-201701140159_noticia.html
- [4] «National defense authorization act for fiscal year 2006». 109th Congress. Accedido: 7 de marzo de 2024. [En línea]. Disponible en: <https://www.govinfo.gov/content/pkg/PLAW-109publ163/html/PLAW-109publ163.htm>
- [5] «Inteligencia de fuentes abiertas (OSINT): características, debilidades y engaño. Yago Rodríguez». Accedido: 7 de marzo de 2024. [En línea]. Disponible en: <https://www.seguridadinternacional.es/?q=es/content/inteligencia-de-fuentes-abiertas-osint-caracter%C3%ADsticas-debilidades-y-enga%C3%B1o>
- [6] «Las redes sociales, la principal arma terrorista durante la pandemia de COVID-19 | Noticias ONU». Accedido: 7 de marzo de 2024. [En línea]. Disponible en: <https://news.un.org/es/story/2020/11/1484342>
- [7] «ISIS Recruiters, Propagandists, and Inciters to Violence Operating on Twitter», Counter Extremism Project. Accedido: 7 de marzo de 2024. [En línea]. Disponible en: <https://www.counterextremism.com/content/isis-recruiters-propagandists-and-inciters-violence-operating-twitter>
- [8] «ChatGPT». openAI. Accedido: 18 de marzo de 2024. [En línea]. Disponible en: <https://chat.openai.com>
- [9] «Llama 2: Open Foundation and Fine-Tuned Chat Models». H. Touvron, L. Martin, y K. Stone.
- [10] «Ciclo de inteligencia: qué es, para qué sirve y cuáles son sus límites», LISA Institute. Accedido: 30 de enero de 2024. [En línea]. Disponible en: <https://www.lisainstitute.com/blogs/blog/ciclo-de-inteligencia>
- [11] «Gestión del conocimiento y servicios de inteligencia: la dimensión estratégica de la información». Artículo académico. M. A. Esteban Navarro y D. Navarro Monilla, ago. 2003.

- [12] «Tipos de Inteligencia». R. J. Villalonga. Global Strategy. Accedido: 30 de enero de 2024. [En línea]. Disponible en: <https://global-strategy.org/tipos-de-inteligencia/>
- [13] «How Much Data Do We Create Every Day? The Mind-Blowing Stats Everyone Should Read». B. Marr. Forbes. Accedido: 31 de enero de 2024. [En línea]. Disponible en: <https://www.forbes.com/sites/bernardmarr/2018/05/21/how-much-data-do-we-create-every-day-the-mind-blowing-stats-everyone-should-read/>
- [14] «Current Status and Security Trend of OSINT». Y.-W. Hwang, I.-Y. Lee, H. Kim, H. Lee, y D. Kim. Wirel. Commun. Mob. Comput., vol. 2022, p. e1290129, feb. 2022, doi: 10.1155/2022/1290129.
- [15] «OSINT (Inteligencia de Fuentes Abiertas): tipos, métodos y salidas profesionales», LISA Institute. Accedido: 21 de febrero de 2024. [En línea]. Disponible en: <https://www.lisainstitute.com/blogs/blog/osint-inteligencia-fuentes-abiertas>
- [16] «Google - Acerca de». Google. Accedido: 18 de marzo de 2024. [En línea]. Disponible en: https://about.google/intl/es-419_ALL/
- [17] «XPro», X (formerly Twitter). Elon Musk. Accedido: 18 de marzo de 2024. [En línea]. Disponible en: <https://twitter.com/Pro>
- [18] «Maltego». Paterva. Accedido: 18 de marzo de 2024. [En línea]. Disponible en: <https://www.maltego.com/>
- [19] «Google News». Google. Accedido: 18 de marzo de 2024. [En línea]. Disponible en: <https://news.google.com/home?hl=en-US&gl=US&ceid=US:en>
- [20] «Google Lens: Busca lo que ves», Google. Accedido: 18 de marzo de 2024. [En línea]. Disponible en: <https://lens.google/intl/es/>
- [21] «Wireshark». The Wireshark team. Accedido: 18 de marzo de 2024. [En línea]. Disponible en: <https://www.wireshark.org/download.html>
- [22] «Mwtracker, detect emerging document threats». Malwaretracker team. Accedido: 18 de marzo de 2024. [En línea]. Disponible en: <https://malwaretracker.com/>
- [23] «Real-Time Event and Risk Detection», Dataminr. Accedido: 18 de marzo de 2024. [En línea]. Disponible en: <https://www.dataminr.com/>
- [24] «Whois Lookup, Domain Availability & IP Search» - DomainTools. Accedido: 18 de marzo de 2024. [En línea]. Disponible en: <https://whois.domaintools.com/>
- [25] «El Proyecto Tor | Privacidad & Libertad en línea». Accedido: 18 de marzo de 2024. [En línea]. Disponible en: <https://torproject.org>
- [26] «ExifTool». Phil Harvey. Accedido: 18 de marzo de 2024. [En línea]. Disponible en: <https://exiftool.org/>
- [27] «NetWitness Investigator Freeware». CTX Corporation. Accedido: 18 de marzo de 2024. [En línea]. Disponible en: <https://www.netwitness.com/contact-us/netwitness-investigator-freeware/>
- [28] «UnkL4b/GitMiner». UnkL4b, 16 de marzo de 2024. Accedido: 18 de marzo de 2024. [En línea]. Disponible en: <https://github.com/UnkL4b/GitMiner>
- [29] «WikiLeaks». Julian Assange. Accedido: 18 de marzo de 2024. [En línea]. Disponible en: <https://wikileaks.org/>
- [30] «Social-analyzer: API, CLI & Web App for analyzing & finding a person's profile across 300+ social media websites (Detections are updated regularly)». Accedido: 18 de marzo de 2024. [En línea]. Disponible en: <https://github.com/qeeqbox/social-analyzer>

- [31] «Owler», Owler. Accedido: 18 de marzo de 2024. [En línea]. Disponible en: <https://corp.owler.com>
- [32] «NVD - Home». MITRE Corporation. Accedido: 18 de marzo de 2024. [En línea]. Disponible en: <https://nvd.nist.gov/>
- [33] «Analytics». Google. Accedido: 18 de marzo de 2024. [En línea]. Disponible en: <https://analytics.google.com/analytics/web/provision/?hl=es#/provision>
- [34] «A Brief History of Artificial Intelligence: On the Past, Present, and Future of Artificial Intelligence». M. Haenlein y A. Kaplan, , California Management Review, jul. 2019, DOI: 10.1177/0008125619864925.
- [35] «Perceptrons: An Introduction to Computational Geometry». M. Minsky y S. A. Papert. 1969. ISBN electronic: 9780262343930
- [36] «Machine Learning | Qué es, tipos, ejemplos y cómo implementarlo», GraphEverywhere. Accedido: 27 de febrero de 2024. [En línea]. Disponible en: <https://www.grapheverywhere.com/machine-learning-que-es-tipos-ejemplos-y-como-implementarlo/>
- [37] «Los 10 Algoritmos esenciales en Machine Learning», Raona. Accedido: 18 de marzo de 2024. [En línea]. Disponible en: <https://www.raona.com/los-10-algoritmos-esenciales-machine-learning/>
- [38] «An Introduction to Deep Learning». L. Arnold, S. Rebecchi, S. Chevallier, y H. Paugam-Moisy. Conference: ESANN, abr. 2011. ISBN: 978-2-87419-044-5
- [39] «Deep Neural Network: The 3 Popular Types (MLP, CNN and RNN)». G. Boesch. Accedido: 28 de febrero de 2024. [En línea]. Disponible en: <https://viso.ai/deep-learning/deep-neural-network-three-popular-types/>
- [40] «Algoritmo adam en la inteligencia artificial». A. R. Aguilar, R. S. Moreno, L. Miranda, y W. Ojeda. Artículo: COMEII-21005
- [41] «Xavier Initialization | PDF | Red neuronal artificial | Optimización Matemática», Elvin Ramos. Accedido: 18 de marzo de 2024. [En línea]. Disponible en: <https://es.scribd.com/document/656689610/XAVIER-INITIALIZATION>
- [42] «Deep Learning». L. Centeno Franco. TFG. Universidad de Sevilla. 2019. Disponible en: <https://idus.us.es/bitstream/handle/11441/90004/Centeno%20Franco%20Alba%20TFG.pdf?sequence=1&isAllowed=y>
- [43] «NLP (natural language processing): ¿qué es y para qué sirve?», UNIR. Accedido: 4 de marzo de 2024. [En línea]. Disponible en: <https://www.unir.net/marketing-comunicacion/revista/nlp-procesamiento-lenguaje-natural/>
- [44] «¿Cómo funcionan los Transformers? en español | Aprende Machine Learning». Usuario: Na8. Accedido: 4 de marzo de 2024. [En línea]. Disponible en: <https://www.aprendemachinelearning.com/como-funcionan-los-transformers-espanol-nlp-gpt-bert/>
- [45] «Word Embedding Demo: Tutorial». Accedido: 4 de marzo de 2024. [En línea]. Disponible en: <https://www.cs.cmu.edu/~dst/WordEmbeddingDemo/tutorial.html>
- [46] «AI Dungeon». Nick Walton, Latitude. Accedido: 18 de marzo de 2024. [En línea]. Disponible en: <https://aidungeon.com/>
- [47] «Otter.ai - AI Meeting Note Taker & Real-time AI Transcription». Sam Liang. Accedido: 18 de marzo de 2024. [En línea]. Disponible en: <https://otter.ai/>
- [48] «Traductor de Google». Google. Accedido: 18 de marzo de 2024. [En línea]. Disponible en: <https://translate.google.es/?hl=es>

- [49] «Text Analytics», MonkeyLearn. Accedido: 18 de marzo de 2024. [En línea]. Disponible en: <https://monkeylearn.com/>
- [50] «GPT-4». OpenAI. Accedido: 18 de marzo de 2024. [En línea]. Disponible en: <https://openai.com/gpt-4>
- [51] «What are Large Language Models? | NVIDIA Glossary», NVIDIA. Accedido: 6 de marzo de 2024. [En línea]. Disponible en: <https://www.nvidia.com/en-us/glossary/large-language-models/>
- [52] «word2vec | Text», TensorFlow. Accedido: 18 de marzo de 2024. [En línea]. Disponible en: <https://www.tensorflow.org/text/tutorials/word2vec>
- [53] «GloVe: Global Vectors for Word Representation». Jeffrey Pennington, Richard Socher, Christopher D. Manning. Accedido: 19 de marzo de 2024. [En línea]. Disponible en: <https://nlp.stanford.edu/projects/glove/>
- [54] «google-bert/bert-base-uncased · Hugging Face». Google. Accedido: 18 de marzo de 2024. [En línea]. Disponible en: <https://huggingface.co/google-bert/bert-base-uncased>
- [55] «Gemini - Habla para dar rienda suelta a tus ideas», Gemini. Accedido: 18 de marzo de 2024. [En línea]. Disponible en: <https://gemini.google.com>
- [56] «Llama», Meta. Accedido: 18 de marzo de 2024. [En línea]. Disponible en: <https://llama.meta.com/>
- [57] «LLaMA: Open and Efficient Foundation Language Models». H. Touvron et al., , arXiv:2302.13971, feb. 2023. Disponible en: <https://arxiv.org/pdf/2302.13971>.
- [58] «Introducing LLaMA: A foundational, 65-billion-parameter language model». Meta. Accedido: 6 de marzo de 2024. [En línea]. Disponible en: <https://ai.meta.com/blog/large-language-model-llama-meta-ai/>
- [59] «Russia-Ukraine War Through the Eyes of Social Media | GJIA». Magdalene Karalis. Georgetown Journal of International Affairs. Accedido: 6 de marzo de 2024. [En línea]. Disponible en: <https://gjia.georgetown.edu/2024/02/02/russia-ukraine-through-the-eyes-of-social-media/>
- [60] «Citizen journalism | Benefits, Challenges & Impact» Britannica. Accedido: 6 de marzo de 2024. [En línea]. Disponible en: <https://www.britannica.com/topic/citizen-journalism>
- [61] «The Evolution of Social Media: How Did It Begin, and Where Could It Go Next?», Maryville University Online. Accedido: 7 de marzo de 2024. [En línea]. Disponible en: <https://online.maryville.edu/blog/evolution-social-media/>
- [62] «Qué es la “teoría de los seis grados” y cómo llevó a la creación de la primera red social en internet», BBC News Mundo. Accedido: 19 de marzo de 2024. [En línea]. Disponible en: <https://www.bbc.com/mundo/noticias-48558989>
- [63] «Inicio / X», X (formerly Twitter). Accedido: 19 de marzo de 2024. [En línea]. Disponible en: <https://twitter.com/home?lang=es>
- [64] «Twitter y la argumentación tecno-comunicativa en la nueva ola de la primavera árabe en argelia». N. Irakrak en Actas del III Congreso Internacional Move.net sobre Movimientos Sociales y TIC, Universidad de Sevilla: Universidad de Sevilla, nov. 2019. ISBN: 978-84-09-20916-3. Disponible en: <https://idus.us.es/bitstream/handle/11441/97825/Twitter.pdf?sequence=1&isAllowed=y>
- [65] «X | Company, History, Twitter, Elon Musk, & Uses». Britannica. Accedido: 7 de marzo de 2024. [En línea]. Disponible en: <https://www.britannica.com/topic/Twitter>
- [66] «Twitter amplía el número de caracteres por tweet, pero con una condición | Tecnología». LOS40. Accedido: 7 de marzo de 2024. [En línea]. Disponible en: https://los40.com/los40/2023/02/09/tecnologia/1675957593_376262.html

- [67] «Esto es todo lo que Elon Musk ha destruido en X (Twitter) en solo un año». R. Badillo. *elconfidencial.com*. Accedido: 7 de marzo de 2024. [En línea]. Disponible en: https://www.elconfidencial.com/tecnologia/2023-10-27/elon-musk-destruir-twitter_3763008/
- [68] «Instagram | History, Features, Description, & Facts». *Britannica*. Accedido: 7 de marzo de 2024. [En línea]. Disponible en: <https://www.britannica.com/topic/Instagram>
- [69] «YouTube». Accedido: 19 de marzo de 2024. [En línea]. Disponible en: <https://www.youtube.com/>
- [70] «(Un)founded fear towards the algorithm: YouTube recommendations and polarisation». J. García-Marín y I.-J. Serrano-Contreras. *Comunicar*, vol. 31, n.o 74, pp. 61-70. ene. 2023, doi: 10.3916/C74-2023-05.
- [71] «Los youtubers se mudan a Twitch». L. Marketing. *Libo Marketing*. Accedido: 2 de abril de 2024. [En línea]. Disponible en: <https://libomarketing.com/los-youtubers-se-mudan-a-twitch/>
- [72] «Netflix España - Ver series en línea, ver películas en línea». *Netflix España*. Accedido: 19 de marzo de 2024. [En línea]. Disponible en: <https://www.netflix.com/es/>
- [73] «Watch Thousands of Shows and Movies». *Hulu*. Accedido: 19 de marzo de 2024. [En línea]. Disponible en: https://www.hulu.com/welcome?orig_referrer=https%3A%2F%2Fwww.google.com%2F
- [74] «Preguntas frecuentes», *Telegram*. Accedido: 7 de marzo de 2024. [En línea]. Disponible en: <https://telegram.org/faq/es#p-como-cifran-los-datos>
- [75] «Telegram | Overview, History, & Facts». *Britannica*. Accedido: 7 de marzo de 2024. [En línea]. Disponible en: <https://www.britannica.com/topic/Telegram-software>
- [76] «Meta | Social Metaverse Company», *Meta | Social Metaverse Company*. Accedido: 19 de marzo de 2024. [En línea]. Disponible en: <https://about.meta.com/es/>
- [77] «The World’s Most Important App (For Now)». C. Warzel. *The Atlantic*. Accedido: 7 de marzo de 2024. [En línea]. Disponible en: <https://www.theatlantic.com/technology/archive/2023/06/telegram-app-encrypted-messaging-russia/674558/>
- [78] «Introduction to the Telegram API». J. Yi. *Medium*. Accedido: 7 de marzo de 2024. [En línea]. Disponible en: <https://towardsdatascience.com/introduction-to-the-telegram-api-b0cd220dbed2>
- [79] «Telegram APIs». *Telegram*. Accedido: 7 de marzo de 2024. [En línea]. Disponible en: <https://core.telegram.org/>
- [80] «Sistema de detección inteligente de usuarios de twitter interesados en la temática de defensa». M. Fernández Torrejón, N. (advisor) Fernández García, y M. (advisor) Fernández Gavilanes. abr. 2023. Accedido: 2 de abril de 2024. [En línea]. Disponible en: <http://calderon.cud.uvigo.es/handle/123456789/678>
- [81] «Herramientas de OSINT para la obtención de información en Twitter sobre el conflicto Ucrania-Rusia». Á. Hernández Ruiz y N. Fernández García (advisor). abr. 2022. Accedido: 2 de abril de 2024. [En línea]. Disponible en: <http://calderon.cud.uvigo.es/handle/123456789/649>
- [82] «Plataforma OSINT para el almacenamiento y análisis inteligente de la red social Twitter». J. Jiménez-Cervantes Pérez, N. (advisor) Fernández García, y M. (advisor) Rodelgo Lacruz. abr. 2023. Accedido: 2 de abril de 2024. [En línea]. Disponible en: <http://calderon.cud.uvigo.es/handle/123456789/726>
- [83] «Evaluation of LLM Chatbots for OSINT-based Cyber Threat Awareness». S. Shafee, A. Bessani, y P. M. Ferreira. 13 de marzo de 2024. doi: 10.48550/arXiv.2401.15127.

- [84] «ChatGPT Act as an Intelligence Officer». Lauro Snidaro. November 2023. doi: 10.1109/TechDefense59795.2023.10380914
- [85] «Google Colaboratory». Google. Accedido: 21 de marzo de 2024. [En línea]. Disponible en: <https://colab.research.google.com/>
- [86] «Project Jupyter». IPython. Accedido: 21 de marzo de 2024. [En línea]. Disponible en: <https://jupyter.org>
- [87] «Python España». IPython Accedido: 21 de marzo de 2024. [En línea]. Disponible en: <https://es.python.org/>
- [88] «Unleash AI Innovation and Value»| Anaconda. Accedido: 21 de marzo de 2024. [En línea]. Disponible en: <https://www.anaconda.com/>
- [89] «Ventajas y Desventajas de Python: Lo que Necesitas Saber». María Ortiz. Accedido: 8 de marzo de 2024. [En línea]. Disponible en: <https://stride.com.co/blog/ventajas-desventajas-python/>
- [90] «pyTelegramBotAPI: Python Telegram bot api.» Usuario: Badiboy. Accedido: 21 de marzo de 2024. [En línea]. Disponible en: <https://github.com/eternnoir/pyTelegramBotAPI>
- [91] «Strange behaviour of MessageHandler in nested ConversationHandlers · Issue #2348 · python-telegram-bot/python-telegram-bot», GitHub. Accedido: 12 de marzo de 2024. [En línea]. Disponible en: <https://github.com/python-telegram-bot/python-telegram-bot/issues/2348>
- [92] «API de datos de YouTube: auditorías de cuotas y cumplimiento». YouTube Data API. Google for Developers. Accedido: 13 de marzo de 2024. [En línea]. Disponible en: https://developers.google.com/youtube/v3/guides/quota_and_compliance_audits?hl=es-419
- [93] «jdepoix/youtube-transcript-api». J. Depoix. 13 de marzo de 2024. Accedido: 13 de marzo de 2024. [En línea]. Disponible en: <https://github.com/jdepoix/youtube-transcript-api>
- [94] «gonferspanish/OSINT · Datasets at Hugging Face». Gonzalo Fernández. Accedido: 18 de marzo de 2024. [En línea]. Disponible en: <https://huggingface.co/datasets/gonferspanish/OSINT>
- [95] «Research», AI at Meta. Accedido: 21 de marzo de 2024. [En línea]. Disponible en: <https://ai.meta.com/research/>
- [96] «TRL - Transformer Reinforcement Learning». Hugging Face. Accedido: 13 de marzo de 2024. [En línea]. Disponible en: <https://huggingface.co/docs/trl/main/en/index>
- [97] «LoRA: Low-Rank Adaptation of Large Language Models». E. J. Hu et al. arXiv, 16 de octubre de 2021. Accedido: 18 de marzo de 2024. [En línea]. Disponible en: <http://arxiv.org/abs/2106.09685>
- [98] «QLoRA: Efficient Finetuning of Quantized LLMs». T. Dettmers, A. Pagnoni, A. Holtzman, y L. Zettlemoyer. arXiv: 2305.14314. 23 de mayo de 2023. Accedido: 18 de marzo de 2024. [En línea]. Disponible en: <http://arxiv.org/abs/2305.14314>
- [99] «PEFT». Hugging Face. Accedido: 22 de marzo de 2024. [En línea]. Disponible en: <https://huggingface.co/docs/peft/index>
- [100] «facebookresearch/bitsandbytes». Meta Research. 29 de febrero de 2024. Accedido: 22 de marzo de 2024. [En línea]. Disponible en: <https://github.com/facebookresearch/bitsandbytes>
- [101] «gonferspanish/PAQUITA · Hugging Face». Gonzalo Fernández. Accedido: 25 de marzo de 2024. [En línea]. Disponible en: <https://huggingface.co/gonferspanish/PAQUITA>
- [102] «xFormers». Hugging Face. Accedido: 25 de marzo de 2024. [En línea]. Disponible en: <https://huggingface.co/docs/diffusers/optimization/xformers>

- [103] «Transformers: State-of-the-Art Natural Language Processing». T. Wolf et al. Association for Computational Linguistics, pp. 38-45. octubre de 2020. Accedido: 25 de marzo de 2024. [En línea]. Disponible en: <https://www.aclweb.org/anthology/2020.emnlp-demos.6>
- [104] «huggingface/accelerate». Hugging Face, 2024. Accedido: 25 de marzo de 2024. [En línea]. Disponible en: <https://github.com/huggingface/accelerate>
- [105] «einops». Usuario: ollema. Accedido: 25 de marzo de 2024. [En línea]. Disponible en: <https://github.com/arogozhnikov/einops/blob/master/docs/1-einops-basics.ipynb>
- [106] «huggingface/safetensors». Hugging Face. Accedido: 25 de marzo de 2024. [En línea]. Disponible en: <https://github.com/huggingface/safetensors>
- [107] «Use with PyTorch». Hugging Face. Accedido: 25 de marzo de 2024. [En línea]. Disponible en: https://huggingface.co/docs/datasets/use_with_pytorch
- [108] «Definición de defensa nacional - Diccionario panhispánico del español jurídico». RAE. Diccionario panhispánico del español jurídico - Real Academia Española. Accedido: 19 de marzo de 2024. [En línea]. Disponible en: <https://dpej.rae.es/lema/defensa-nacional>
- [109] «La matriz de confusión y sus métricas – Inteligencia Artificial –», Juan Barrios. Accedido: 2 de abril de 2024. [En línea]. Disponible en: <https://www.juanbarrios.com/la-matriz-de-confusion-y-sus-metricas/>

ANEXO I: IMPLICACIONES SOCIALES, ECONÓMICAS, AMBIENTALES

La construcción de una herramienta de análisis OSINT conlleva una serie de implicaciones en diversos aspectos sociales, medioambientales y económicos que deben ser considerados cuidadosamente.

– Implicaciones Sociales:

En primer lugar, se debe tener en cuenta que el análisis de perfiles de redes sociales implica el acceso y tratamiento de información personal y actividades en línea. Este proceso puede generar ciertas cuestiones sobre la privacidad y seguridad de los usuarios. Las herramientas de análisis de OSINT deben garantizar el respeto de los derechos de privacidad de los individuos y la protección de sus datos personales, siguiendo siempre la legislación vigente sobre protección de datos.

Por otro lado, un control excesivo de las redes sociales puede llegar a considerarse como una privación de la libertad de expresión de los usuarios, por lo que este tipo de tecnologías deben utilizarse con precaución y siempre para combatir comportamientos ilícitos y no con otros fines como obtención de datos, estudios de mercado o espionaje.

No obstante, pese a las cuestiones negativas, el análisis OSINT puede facilitar tareas orientadas a la seguridad nacional, la lucha contra el terrorismo y la desinformación en las redes, trayendo consigo grandes beneficios no solo para las fuerzas y cuerpos de seguridad del estado, sino que también para la sociedad en general.

– Implicaciones Medioambientales:

Pese a que las implicaciones medioambientales pueden no ser de tanta relevancia para este proyecto, existen una serie de cuestiones a tener en cuenta. En primer lugar, el consumo de recursos computacionales ya que la ejecución de procesos de raspado de datos y análisis de grandes volúmenes de información en la nube puede requerir una cantidad significativa de recursos computacionales y energía contribuyendo directamente a la huella de carbono asociada con la actividad informática. Por este motivo, se deben explorar enfoques para optimizar el uso de recursos y reducir el impacto ambiental de la herramienta.

En otro ámbito, cabe destacar que el desarrollo y uso de tecnología digital contribuyen a la generación de residuos electrónicos cuando los dispositivos informáticos alcanzan el final de su vida útil. El desecho inadecuado de estos residuos puede tener impactos negativos en el medio ambiente y la salud humana. Es fundamental adoptar prácticas responsables de gestión de residuos electrónicos, como el reciclaje y la reutilización de componentes, para mitigar este impacto.

– Implicaciones Económicas:

La construcción y utilización de este tipo de herramientas impulsan el desarrollo tecnológico en campos de programación, seguridad e inteligencia artificial. Esto puede generar nuevas oportunidades de empleo en sectores emergentes de la industria tecnológica, así como también demanda de habilidades especializadas en análisis de datos y programación, contribuyendo al crecimiento económico y a la innovación.

Adicionalmente, aunque en este caso ha sido prácticamente despreciable debido a las condiciones de trabajo, es destacable para el ámbito económico el coste monetario que conlleva el desarrollo de una herramienta de este tipo. Esto incluye los salarios y honorarios del personal, el coste de las herramientas y licencias de software, los costos de infraestructura y hardware, servicios en la nube y posibles suscripciones a APIs.

ANEXO II: REFLEXIONES ÉTICAS Y SOCIALES

La captura y análisis de datos en redes sociales plantea preocupaciones sobre la privacidad y la protección de los datos personales. Es crucial asegurarse de que el uso de la herramienta cumpla con las leyes y regulaciones de privacidad de datos, así como con los términos de servicio de las plataformas de redes sociales. Por ello, tras el desarrollo del presente proyecto, cabe resaltar los siguientes postulados:

- Es de implícita exclusión en los términos de uso de las API's almacenar los datos extraídos de las plataformas sociales. Por lo que la herramienta debe eliminar los archivos obtenidos en el módulo de captura de datos tras su análisis.
- Los términos de uso de Llama 2 limitan su empleo con fines militares o de espionaje, por lo que cabe resaltar que el presente proyecto es una prueba de concepto que se ha realizado únicamente con fines científicos y de investigación, no estando previsto el paso de la herramienta a un entorno de producción.
- Debe resaltarse también que esta herramienta únicamente se centra en el análisis de perfiles e información de carácter público, por lo que en ningún momento viola la privacidad de los usuarios de las plataformas.
- La herramienta no ofrece acceso a información personal ni de carácter sensible.

ANEXO III: CÓDIGO

Fine Tuning

```
!pip install -q accelerate==0.21.0 peft==0.4.0 bitsandbytes==0.40.2 trl==0.4.7
!pip install git+https://github.com/huggingface/transformers@v4.31-release
```

```
import os
import torch
from datasets import load_dataset
from transformers import (
    AutoModelForCausalLM,
    AutoTokenizer,
    BitsAndBytesConfig,
    HfArgumentParser,
    TrainingArguments,
    pipeline,
    logging,)
from peft import LoraConfig, PeftModel
from trl import SFTTrainer
from huggingface_hub import notebook_login

notebook_login()
```

```
model_name = "meta-llama/Llama-2-7b-chat-hf" #Modelo inicial
```

```
dataset_name = "gonferspanish/OSINTJSON" #Nombre del dataset
```

```
new_model = "PAQUITA" #Modelo ajustado
```

```
#####
QLoRA parameters
```

```
#####
lora_r = 64
```

```
lora_alpha = 16
```

```
lora_dropout = 0.1
```

```
#####
bitsandbytes parameters
```

```
#####
use_4bit = True #Activar cuantización en 4 bits
```

```
bnb_4bit_compute_dtype = "float16"
```

```
bnb_4bit_quant_type = "nf4" #Cuantización normalizada en 4 bits
```

```
use_nested_quant = False
```

```
#####
TrainingArguments parameters
```

```
#####
```

```
num_train_epochs = 3 #Número de vueltas al dataset

per_device_train_batch_size = 2 #Tamaño de GPU para entrenamiento

per_device_eval_batch_size = 2 #Tamaño de GPU para evaluación

gradient_accumulation_steps = 1 #Gradiente acumulado en pasos

gradient_checkpointing = True #Punto de control de gradiente

max_grad_norm = 0.3 #Gradiente máximo

learning_rate = 2e-4 #Tasa de aprendizaje

weight_decay = 0.001 #Penalización de pesos

optim = "paged_adamw_32bit" #Optimizador

lr_scheduler_type = "cosine" #Programador de tasa de aprendizaje

max_steps = -1 #Número de pasos

warmup_ratio = 0.03 #Tasa de calentamiento

group_by_length = False #Agrupamiento de secuencias de misma longitud

save_steps = 0 #Almacenamiento de puntos de control

logging_steps = 20 #Pasos para cada registro

#####
SFT parameters
#####
max_seq_length = None #Longitud máxima de secuencia

packing = False #Empaquetamiento en lotes

device_map = {"": 0}

dataset = load_dataset(dataset_name, split="train") #Carga el dataset

def formatting_func(example): #Función formato
    text = f'''<s>[INST] <<SYS>>Classify whether the following text
    is explicitly related to defense or military topics or not.<<SYS>>
    {{Text}}: {example['text']}
    [/INST]{{Answer}}: {example['answer']}</s>'''
    return text

compute_dtype = getattr(torch, bnb_4bit_compute_dtype) #Configuración de
cuantización

bnb_config = BitsAndBytesConfig(
    load_in_4bit=use_4bit,
```

```

bnb_4bit_quant_type=bnb_4bit_quant_type,
bnb_4bit_compute_dtype=compute_dtype,
bnb_4bit_use_double_quant=use_nested_quant,)

if compute_dtype == torch.float16 and use_4bit:
    major, _ = torch.cuda.get_device_capability()
    if major >= 8:
        print("=" * 80)
        print("Your GPU supports bfloat16: accelerate training with bf16=True")
        print("=" * 80)

model = AutoModelForCausalLM.from_pretrained( #Cargar el modelo preentrenado
    model_name,
    quantization_config=bnb_config,
    device_map=device_map)
model.config.use_cache = False
model.config.pretraining_tp = 1

tokenizer = AutoTokenizer.from_pretrained(model_name, trust_remote_code=True)
#Configuración del tokenizador
tokenizer.pad_token = tokenizer.eos_token
tokenizer.padding_side = "right"

peft_config = LoraConfig( #Configuración de QLoRA
    lora_alpha=lora_alpha,
    lora_dropout=lora_dropout,
    r=lora_r,
    bias="none",
    task_type="CAUSAL_LM",)

training_arguments = TrainingArguments( #Configuración de Los parámetros de
entrenamiento
    output_dir=output_dir,
    num_train_epochs=num_train_epochs,
    per_device_train_batch_size=per_device_train_batch_size,
    gradient_accumulation_steps=gradient_accumulation_steps,
    optim=optim,
    save_steps=save_steps,
    logging_steps=logging_steps,
    learning_rate=learning_rate,
    weight_decay=weight_decay,
    fp16=fp16,
    bf16=bf16,
    max_grad_norm=max_grad_norm,
    max_steps=max_steps,
    warmup_ratio=warmup_ratio,
    group_by_length=group_by_length,
    lr_scheduler_type=lr_scheduler_type,
    report_to="tensorboard")

trainer = SFTTrainer(
    model=model,
    train_dataset=dataset,
    peft_config=peft_config,
    max_seq_length=max_seq_length,

```

```
tokenizer=tokenizer,
args=training_arguments,
packing=packing,
formatting_func=formatting_func)

trainer.train()

trainer.model.save_pretrained(new_model)

base_model = AutoModelForCausalLM.from_pretrained( #Carga el modelo en FP16 y lo mezcla con los pesos de LoRA 16
    model_name,
    low_cpu_mem_usage=True,
    return_dict=True,
    torch_dtype=torch.float16,
    device_map=device_map,
)
model = PeftModel.from_pretrained(base_model, new_model)
model = model.merge_and_unload()

tokenizer = AutoTokenizer.from_pretrained(model_name, trust_remote_code=True)
#Carga el tokenizador en 16 bits
tokenizer.pad_token = tokenizer.eos_token
tokenizer.padding_side = "right"

model.push_to_hub(new_model, use_temp_dir=False, token="TOKEN") #Sube el modelo a Hugging face
tokenizer.push_to_hub(new_model, use_temp_dir=False, token="TOKEN") #Sube el tokenizador a Hugging Face
```

Inferencia y asistente

```
!pip install bitsandbytes
!pip install git+https://github.com/huggingface/transformers.git
!pip install git+https://github.com/huggingface/accelerate.git
!pip install einops
!pip install safetensors
!pip install torch
!pip install xformers
!pip install instaloader
!pip install youtube_transcript_api
!pip install pytelegrambotapi

import transformers
import torch
import json
from transformers import AutoTokenizer, AutoModelForCausalLM, BitsAndBytesConfig, pipeline, TextStreamer
from huggingface_hub import notebook_login
import instaloader
import csv
import re
import os
from googleapiclient.discovery import build
from youtube_transcript_api import YouTubeTranscriptApi
```

```

from telebot import types
from transformers import LlamaTokenizer
import csv
import random

notebook_login() #Verificarse en Hugging Face

model_id = "gonferspanish/PAQUITA" #Nombre del modelo en Hugging Face

bnb_config = BitsAndBytesConfig( #Activa la cuantización en 4 bits
    load_in_4bit=True,
    bnb_4bit_use_double_quant=True,
    bnb_4bit_quant_type="nf4",
    bnb_4bit_compute_dtype=torch.bfloat16,)
model = AutoModelForCausalLM.from_pretrained(
    model_id,
    quantization_config=bnb_config,
    device_map=runtimeFlag,
    trust_remote_code=True,
    cache_dir=cache_dir)

def limitar_tokens(tokenizador, texto, max_tokens):
    tokens = tokenizador.tokenize(texto)

    if len(tokens) > max_tokens:
        tokens = tokens[:max_tokens] #Trunca los tokens si exceden el límite

    return tokenizador.convert_tokens_to_string(tokens)

tokenizer = LlamaTokenizer.from_pretrained(model_id, cache_dir=cache_dir)

chat_id = None #Inicializar variables
is_running = False
iusername = None
user = None
password = None

#Credenciales de la API de Telegram
bot = telebot.TeleBot("TOKEN")

#Credenciales de la API de YouTube
DEVELOPER_KEY = "TOKEN"
YOUTUBE_API_SERVICE_NAME = "youtube"
YOUTUBE_API_VERSION = "v3"

#Personaliza el teclado
teclado_personalizado = types.ReplyKeyboardMarkup(resize_keyboard=True)
teclado_personalizado.row('/Instagram 📷', '/YouTube 📺')
teclado_personalizado.row('/Doc 📄', '/info ⓘ')

#####
YOUTUBE SCRAPER
#####
@bot.message_handler(commands=['YouTube']) #Recibir el comando /YouTube
def youtube(message):

```

```
global is_running
global chat_id
chat_id = message.chat.id
if is_running==False:
    is_running = True
    bot.reply_to(message, "Ingrese el nombre de usuario del canal de YouTube:")
    bot.register_next_step_handler(message, handle_youtube_username)
else:
    bot.reply_to(message, "Ya hay un proceso en curso. Por favor, espere a que termine.") #Protocolo error

def handle_youtube_username(message): #Recibir el nombre de un canal de YouTube
    global yusername
    global is_running
    yusername = message.text
    if message.text.startswith('/'):
        bot.reply_to(message, "Ya hay un comando en curso. Por favor, reinicie el proceso") #Protocolo error
        send_welcome(message)
        is_running = False
        return
    else:
        bot.reply_to(message, "Buscando ID del canal...")
        channel_id = get_channel_id(yusername)
        if channel_id:
            bot.reply_to(message, f"ID del canal encontrado: {channel_id}")
            bot.send_message(chat_id, "Cargando... Por favor espere.")
            youtubescraper(channel_id, yusername)
        else:
            bot.reply_to(message, "No se pudo encontrar el ID del canal. Introduzca el ID manualmente:")
            bot.register_next_step_handler(message, handle_manual_input)

def handle_manual_input(message): #Pide al usuario el ID del canal manualmente
    global is_running
    chat_id = message.chat.id
    channel_id = message.text
    if message.text.startswith('/'):
        bot.reply_to(message, "Ya hay un comando en curso. Por favor, reinicie el proceso") #Protocolo error
        send_welcome(message)
        is_running = False
        return
    else:
        bot.send_message(chat_id, "Cargando... Por favor espere.")
        youtubescraper(channel_id, yusername)

def get_channel_id(yusername): #Obtiene el ID de un canal con el nombre del canal
    try:
        youtube = build(YOUTUBE_API_SERVICE_NAME, YOUTUBE_API_VERSION,
developerKey=DEVELOPER_KEY)
        request = youtube.channels().list(part="id", forUsername=yusername)
        response = request.execute()
        if 'items' in response and response['items']:
            return response['items'][0]['id']
```

```

        else:
            return None
    except Exception as e:
        print(f"Error al obtener el ID del canal: {e}") #Protocolo error
        return None

def youtubescraper(channel_id, yusername): #Extrae y guarda los subtítulos de un
canal de YouTube en archivo CSV
    global is_running
    if channel_id:
        videos = get_channel_videos(channel_id)
        if len(videos)!=0:
            filename = f'{yusername}_videos.csv'
            save_to_csv(videos, filename)
            print(f"Los datos de los {len(videos)} últimos videos de {yusername} se
han guardado en {filename}")
            bot.send_message(chat_id,f"Los datos de los {len(videos)} últimos videos
de {yusername} se han guardado en {filename}")
            csv_filename = filename
            main(csv_filename)
            is_running = False
            return
        else:
            bot.send_message(chat_id,"Error en ID del canal.")#Protocolo error
            is_running = False
            return
    else:
        bot.send_message(chat_id,"Error en el proceso por favor vuelva a empezar.")
#Protocolo error
        is_running = False
        return

def get_channel_videos(channel_id, max_results=1): #Obtiene la lista de vídeos de
un canal
    try:
        youtube = build(YOUTUBE_API_SERVICE_NAME, YOUTUBE_API_VERSION,
developerKey=DEVELOPER_KEY)
        videos = []
        next_page_token = None
        while len(videos) < max_results:
            request = youtube.search().list(part="snippet", channelId=channel_id,
maxResults=min(50, max_results - len(videos)),
pageToken=next_page_token,
order="date", type="video")
            response = request.execute()
            videos.extend(response.get('items', []))
            next_page_token = response.get('nextPageToken')
            if not next_page_token:
                break
        return videos[:max_results]

    except Exception as e:
        print(f"Error al obtener videos del canal: {e}") #Protocolo error
        return []

```

```
def get_video_transcript(video_id): #Extrae los subtítulos del vídeo
    try:
        transcript_list = YouTubeTranscriptApi.list_transcripts(video_id)
        for lang in transcript_list:
            if lang.language_code:
                transcript = lang.fetch()
                text = '\n'.join([line['text'] for line in transcript])
                return text
        print(f"No se encontró transcripción para el video {video_id}")
        return None
    except Exception as e:
        print(f"Error al obtener transcripción para el video {video_id}: {e}")
#Protocolo error
    return None

def save_to_csv(videos, filename): #Guarda en archivo CSV
    try:
        with open(filename, 'w', newline='', encoding='utf-8') as csvfile:
            fieldnames = ['Title', 'Video ID', 'Published At', 'Transcript']
            writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
            writer.writeheader()
            for video in videos:
                transcript = get_video_transcript(video['id']['videoId'])
                writer.writerow({'Title': video['snippet']['title'], 'Video ID':
video['id']['videoId'],
                                'Published At': video['snippet']['publishedAt'],
                                'Transcript': transcript})
            print(f"Datos de los últimos {len(videos)} videos guardados en
{filename}")
    except Exception as e:
        print(f"Error al guardar datos en el archivo CSV: {e}") #Protocolo de
error

#####
MENÚ PRINCIPAL
#####
@bot.message_handler(commands=['start']) #Recibir comando /start
def send_welcome(message):
    chat_id = message.chat.id
    global is_running
    is_running = False
    response = "¡Hola! Bienvenido a la interfaz de análisis de OSINT PAQuITA, ¿en
qué puedo ayudarte?\n\n"
    response += "/YouTube - Analizar una cuenta de YouTube\n"
    response += "/Instagram - Analizar una cuenta de Instagram\n"
    response += "/Doc - Analizar documento de texto\n"
    response += "/info - Información de interés"
    bot.send_message(chat_id, response, reply_markup=teclado_personalizado)

#####
COMANDOS AYUDA
#####
@bot.message_handler(commands=['help']) #Recibir comando /help
def send_help(message):
    chat_id = message.chat.id
```

```

response = "Aquí tienes una lista de comandos disponibles:\n\n"
response += "/help - Muestra esta lista de comandos\n"
response += "/start - Reinicia el bot\n"
response += "/info - Muestra información sobre el bot\n"
response += "/YouTube - Analizar una cuenta de YouTube\n"
response += "/Instagram - Analizar una cuenta de Instagram\n"
response += "/Doc - Analizar documento de texto\n"
bot.send_message(chat_id, response)

@bot.message_handler(commands=['info']) #Recibir comando /info
def send_help(message):
    chat_id = message.chat.id
    response = '''Este es un bot creado por Gonzalo Fernández Conde para el centro
universitario de la defensa.\n\nCon los comandos configurados podrás extraer y
analizar publicaciones de Instagram y YouTube. También puedes analizar documentos
en formato txt con el comando docs.\n\nEl analizador es el modelo LLaMa de Meta, y
esta entrenado para reconocer información en contexto del mundo militar y la
defensa.\n\nPulsa cualquier comando para empezar:
'''
    bot.send_message(chat_id, response)
#####
INSTAGRAM SCRAPER
#####
@bot.message_handler(commands=['Instagram']) #Recibir comando /Instagram
def instagram(message):
    global chat_id, is_running
    chat_id = message.chat.id
    if not is_running:
        is_running = True
        bot.reply_to(message, "Ingrese el nombre de usuario de Instagram que
quiera analizar sin @:")
        bot.register_next_step_handler(message, handle_instagram_username)
    else:
        bot.reply_to(message, "Ya hay un proceso en curso. Por favor, espere a que
termine.")

def handle_instagram_username(message): #Recibir perfil
    global iusername, is_running
    if message.text.startswith('/'):
        bot.reply_to(message, "Ya hay un comando en curso. Por favor, reinicie el
proceso") #Protocolo de error
        send_welcome(message)
        is_running = False
        return
    else:
        iusername = message.text
        bot.reply_to(message, "Se requiere iniciar sesión en Instagram. Por favor,
introduzca su nombre de usuario:")
        bot.register_next_step_handler(message, handle_instagram_user)

def handle_instagram_user(message): #Recibir usuario
    global user, is_running
    if message.text.startswith('/'):
        bot.reply_to(message, "Ya hay un comando en curso. Por favor, reinicie el
proceso") #Protocolo de error

```

```
        is_running = False
        send_welcome(message)
        return
    else:
        user = message.text
        bot.reply_to(message, "Por favor, introduzca su contraseña:")
        bot.register_next_step_handler(message, handle_instagram_password)

def handle_instagram_password(message): #Recibir contraseña
    global password, is_running
    if message.text.startswith('/'):
        bot.reply_to(message, "Ya hay un comando en curso. Por favor, reinicie el
proceso") #Protocolo de error
        send_welcome(message)
        is_running = False
        return
    else:
        password = message.text
        if iniciar_sesion(user, password):
            bot.reply_to(message, "Sesión iniciada correctamente")
            bot.reply_to(message, f"Obteniendo las 20 primeras publicaciones de
{iusername}")
            scrape_instagram(iusername, user, password)
            is_running = False
        else:
            bot.reply_to(message, "Credenciales inválidas. Por favor, inténtelo de
nuevo.") #Protocolo de error
            is_running = False

def clean_caption(caption):
    try:
        clean_text = re.sub(r'^\x00-\x7F+', '', caption) #Elimina caracteres de
nueva línea
        clean_text = clean_text.replace('\n', ' ') #Elimina caracteres de nueva
línea
        return clean_text
    except Exception as e:
        return ""

def iniciar_sesion(user, password): #Función para iniciar sesión en Instagram
    L = instaloder.Instaloder()
    try:
        L.context.log("Intentando iniciar sesión en Instagram...")
        L.login(user, password)
        L.context.log("¡Inicio de sesión exitoso!")
        return L
    except instaloder.exceptions.ProfileNotExistsException as e: #Protocolos de
error
        bot.send_message(chat_id, "Error: El perfil no existe.")
        return None
    except instaloder.exceptions.BadCredentialsException as e:
        return None
    except Exception as e:
        bot.send_message(chat_id, f"Error inesperado: {str(e)}")
        return None
```

```

def scrape_instagram(iusername, user, password):
    L = iniciar_sesion(user, password)
    global is_running
    if L is None:
        return

    try:
        profile = instaloader.Profile.from_username(L.context, iusername)
        bot.send_message(chat_id, f"Perfil encontrado: {profile.username}")
        bot.send_message(chat_id, f"Seguidores: {profile.followers}")
        bot.send_message(chat_id, f"Siguiendo: {profile.followees}")
        bot.send_message(chat_id, f"Número de publicaciones:
{profile.mediacount}")
        csv_file = f"{iusername}_captions.csv"
        with open(csv_file, mode='w', newline='', encoding='utf-8') as file:
            #Guarda en archivo CSV
            writer = csv.writer(file)
            writer.writerow(["Caption"]) #Escribe encabezado del archivo CSV
            count = 0
            for post in profile.get_posts():
                caption = clean_caption(post.caption)
                if caption: #Verifica si la leyenda no está vacía antes de
escribirla
                    writer.writerow([caption]) #Escribe el pie de foto limpio en
el archivo CSV
                    count += 1
                    if count >= 20:
                        break
            print(f"Los pies de foto se han guardado en el archivo {csv_file}")
            csv_filename = csv_file
            main(f"{csv_filename}") #Entrada al analizador
            is_running = False
        return
    except instaloader.exceptions.ProfileNotExistsException as e: #Protocolos de
error
        bot.send_message(chat_id, "Error: El perfil no existe.")
    except instaloader.exceptions.BadCredentialsException as e:
        bot.send_message(chat_id, "Error: Credenciales inválidas.")
    except Exception as e:
        bot.send_message(chat_id, f"Error inesperado: {str(e)}")

#####
ANALIZADOR DE DOCUMENTOS
#####

@bot.message_handler(commands=['Doc']) #Recibir el comando /Doc
def doc(message):
    global is_running #Inicia variables
    global chat_id
    global prompt
    prompt = ""
    chat_id = message.chat.id
    if not is_running:
        is_running = True

```

```
        bot.reply_to(message, "Ingrese un documento en formato txt:") #Pide un
documento al usuario
        bot.register_next_step_handler(message, handle_document)
    else:
        bot.reply_to(message, "Ya hay un proceso en curso. Por favor, espere a que
termine.") #Protocolo para doble comando

def handle_document(message):
    if message.document:
        document_id = message.document.file_id
        file_info = bot.get_file(document_id)
        file_path = file_info.file_path
        downloaded_file = bot.download_file(file_path)
        try:
            file_chunks = [downloaded_file[i:i+4096] for i in range(0,
len(downloaded_file), 4096)] #Divide el archivo en partes más pequeñas
            for chunk in file_chunks: # Decodifica y procesa cada parte
                file_content = chunk.decode("utf-8") #Introduce el contenido en
La variable prompt
                if prompt:
                    prompt += file_content
                else:
                    prompt = file_content
            bot.reply_to(message, "Documento recibido, analizando contenido:")
            main(prompt) #Entrada al analizador
        except Exception as e: #Protocolos de error
            print("Error al decodificar el archivo:", e)
            bot.reply_to(message, "Lo siento, ocurrió un error al procesar el
documento.")
        finally:
            is_running = False
            send_welcome(message)
    else:
        bot.reply_to(message, "Debe enviar un documento en formato txt. Por favor
reinicie el proceso") #Protocolo de error
        is_running = False
        send_welcome(message)

#####
LLAMA 2
#####
def read_csv_file(file_path): #Abre y lee el archivo csv
    try:
        with open(file_path, mode='r', newline='', encoding='utf-8') as file:
            csv_reader = csv.reader(file)
            csv_content = list(csv_reader)
            return csv_content
    except FileNotFoundError:
        print("El archivo no se encontró.")
        return None
    except Exception as e:
        print("Error al leer el archivo CSV:", e)
        return None

def main(file_path): #Introduce el csv en una variable entrada y lo pasa por el
```

LLM

```

csv_content = read_csv_file(file_path)
if csv_content is not None:
    entrada = csv_content
    llama('\n'.join([' '.join(row) for row in entrada]))

def llama(user_prompt):
    system_prompt = '<<SYS>>Classify whether the following text is explicitly
related to defense or military topics or not. Say only yes or no and a brief
argue<<SYS>>'''
    B_INST, E_INST = "[INST]", "[/INST]"
    B_SYS, E_SYS = "<s>", "</s>"
    max_tokens = 4000 #Número máximo de tokens permitidos en el prompt
    limited_user_prompt = limitar_tokens(tokenizer, user_prompt, max_tokens)
    prompt = f"{B_SYS}
{B_INST}{system_prompt.strip()}{E_SYS}{limited_user_prompt.strip()}{E_INST}\n\n"
#Hace el formato del prompt
    inputs = tokenizer([prompt], return_tensors="pt").to(runtimeFlag)
    streamer = TextStreamer(tokenizer)
    mensaje= model.generate(**inputs, streamer=streamer, max_new_tokens=500)
#Inferencia
    texto_mensaje = tokenizer.decode(mensaje[0], skip_special_tokens=True)
#Convierte el mensaje generado en texto
    contenido_despues_de_inst = texto_mensaje.split(E_INST)[-1] #Extrae el
contenido después de [/INST]
    bot.send_message(chat_id, contenido_despues_de_inst) #Envia la respuesta

bot.polling() #Inicia el bot

```