Universidad Internacional de La Rioja
Escuela Superior de Ingeniería y Tecnología

Máster Universitario en Inteligencia Artificial

SoundHive: una propuesta de herramienta de monitorización para verificadores de información

Trabajo fin de estudio presentado por:	Fco. Javier Cantón Correa
Tipo de trabajo:	Desarrollo software
Director/a:	Dra. Marta Mª Arguedas Lafuente
Fecha:	13/9/2023

Resumen

Este Trabajo de Fin de Máster (TFM), perteneciente al Máster de Inteligencia Artificial de la Universidad Internacional de La Rioja (UNIR), aborda el desafío de la desinformación en plataformas digitales, con un enfoque particular en Telegram. El propósito principal de esta investigación es desarrollar una herramienta basada en técnicas avanzadas de procesamiento del lenguaje natural y aprendizaje automático para combatir la desinformación. La metodología adoptada combina un análisis exhaustivo del estado del arte, la identificación de requisitos y el desarrollo iterativo de la herramienta. Como resultado, se ha creado una solución eficaz que permite monitorizar y verificar contenidos en Telegram, contribuyendo significativamente al campo del *fact-checking*. Las conclusiones destacan la relevancia y potencial de la herramienta en la lucha contra la desinformación, sugiriendo líneas de trabajo futuro para mejorar y expandir su alcance.

Palabras clave: aprendizaje semi-supervisado, desinformación, fact-checking, Inteligencia Artificial, Telegram

Abstract

This Master's Thesis (TFM), belonging to the Master in Artificial Intelligence of the International University of La Rioja (UNIR), addresses the challenge of disinformation in digital platforms, with a particular focus on Telegram. The main purpose of this research is to develop a tool based on advanced natural language processing and machine learning techniques to combat disinformation. The methodology adopted combines a thorough analysis of the state of the art, requirements identification and iterative development of the tool. As a result, an effective solution has been created that allows monitoring and verifying content on Telegram, contributing significantly to the field of fact-checking. The conclusions highlight the relevance and potential of the tool in the fight against disinformation, suggesting lines of future work to improve and expand its scope.

Keywords: Artificial Intelligence, disinformation, fact-checking, semi-supervised learning, Telegram

Índice de contenidos

Índice de figuras

"It's easier to fool people than to convince them that they have been fooled"

-Mark Twain

1. INTRODUCCIÓN

El presente documento corresponde al Trabajo de Fin de Máster (TFM) del Máster de Inteligencia Artificial de la Universidad Internacional de La Rioja (UNIR). Este trabajo representa la culminación de un periodo de formación académica y práctica, donde se han adquirido y aplicado conocimientos avanzados en el campo de la Inteligencia Artificial.

Un TFM es una pieza esencial en la formación de posgrado, ya que permite al estudiante demostrar su capacidad para abordar y resolver un problema específico, utilizando las herramientas, técnicas y conocimientos adquiridos durante el curso del máster. Es una oportunidad para llevar a cabo una investigación autónoma, bajo la supervisión de expertos en el campo, y contribuir al avance del conocimiento o la solución de problemas prácticos en el área de estudio.

En el contexto del Máster de Inteligencia Artificial de la UNIR, este TFM se centra en el desafío de la desinformación y la posverdad, especialmente en plataformas digitales como Telegram. A través de este trabajo, se busca desarrollar y validar una herramienta basada en técnicas avanzadas de procesamiento del lenguaje natural y aprendizaje automático para abordar este problema. La elección de este tema no es casual, ya que la desinformación se ha convertido en uno de los desafíos más apremiantes de nuestra era digital, con implicaciones significativas para la sociedad, la política y la cultura.

Esta introducción técnica tiene como objetivo proporcionar un marco general del trabajo realizado, sentando las bases para los capítulos subsiguientes que detallarán el contexto, la metodología, el desarrollo y las conclusiones del proyecto. Es una invitación a adentrarse en un viaje de investigación y desarrollo en el fascinante mundo de la Inteligencia Artificial y su aplicación en la lucha contra la desinformación.

1.1. MOTIVACIÓN

A principios del siglo XXI, con la explosión y disrupción que supuso Internet, cuando se hablaba de la incipiente Era de la Comunicación y la Información, y el nacimiento de la figura del "prosumidor", tal y como predijo Alvin Toffler (1980) décadas antes, nadie podía pensar que el exceso de información iba a convertirse en un auténtico problema. Lo que conocemos como

"infoxicación" (Cornella, 2000), además, ha implicado la aparición de desórdenes de la información, situaciones en las que la información se corrompe y pervierte, voluntaria e involuntariamente, debido a diferentes causas y con distintos objetivos: uno de estos desórdenes es la desinformación.

No es el único desorden, por tanto, pero sí el más visible, el que ha cobrado importancia para entender lo que se ha definido como posverdad (Echevarría, 2016), consecuencia de las ideas procedentes del posmodernismo (Baudrillard, 1998; Bauman, 1996, 2001; Lyotard, 2008), y que ha marcado las agendas políticas, mediáticas, culturales y sociales durante, al menos, los últimos 10 años. Por si fuera poco, también ha dado origen al nacimiento de una nueva disciplina en el mundo del periodismo: la verificación de información y lo que se conoce como el *fact-checking* o comprobación de hechos (vertiente más centrada si cabe en el ámbito político). Verificar información, contrastar y comprobar los hechos, ya era una parte intrínseca y fundamental del propio periodismo, pero la irrupción y emergencia de un fenómeno como el de la desinformación ha hecho que esta parte del proceso se desgaje como una especialidad propia dentro del periodismo y las Ciencias de la Información.

En los últimos años hemos visto cómo la desinformación ha aumentado no sólo en volumen, también en intensidad y en forma: hay más desinformación, cada vez más elaborada, distribuida por cada vez más canales y multimodal, en múltiples formatos: texto, audio y vídeo. La aparición de nuevos modelos generativos de contenidos basados en Inteligencia Artificial no ha hecho más que agravar el problema. El uso de modelos generativos de Inteligencia Artificial en la desinformación plantea diversos retos, como la capacidad de crear contenidos falsos y la rapidez con la que se pueden propagar en línea. Además, estas IAs pueden verse influenciadas por sesgos en los datos de entrenamiento. Sin embargo, también podrían convertirse en parte de la solución a la desinformación al ayudar en la detección, verificación y mitigación de la desinformación. Ya existen numerosas herramientas basadas en IA para luchar contra la desinformación, en manos de verificadores y *fact-checkers* de todo el mundo.

Las necesidades tecnológicas, con este aumento en la complejidad de la desinformación producida, han aumentado consecuentemente también para los verificadores. Aunque son varias las fases en las que las herramientas tecnológicas pueden ser de gran ayuda, es quizás

en la primera fase de monitorización (esto es, la escucha activa en redes sociales y plataformas web de contenidos susceptibles de ser catalogados como desinformación potencial), donde se hace necesaria la incorporación en los flujos de trabajo de una herramienta que unifique enfoques y aúne diferentes usuarios y canales, de diferentes fuentes y procedencias, facilitando así la monitorización y, con ella, los tiempos de detección de la desinformación y su posible impacto social.

1.2. PLANTEAMIENTO DEL PROBLEMA

Lo que se propone en este TFM es el desarrollo de un ***software* modular, a modo de prototipo**, capaz de ayudar a los periodistas dedicados a la verificación en las tareas de monitorización, de tal manera que, en una misma interfaz, el verificador tenga disponibles distintos mensajes procedentes de distintas fuentes (que pueden venir de un *kit* preestablecido o ser añadidas por él mismo), y que puedan ser susceptibles de ser verificados. Para la selección de dichos contenidos se pretende la construcción primero de **extractores de contenido** para cada fuente, bien mediante acceso API en el caso que esto sea posible, bien mediante extractores de web *scraping* elaborados ex profeso. El tratamiento de esos datos habría de ser **multi-modal**, puesto que hablamos de contenidos en formatos de texto y audiovisual, lo que conllevaría el uso de modelos de **NLP** (Natural Language Processing) y **procesamiento de imagen**.

A continuación, un algoritmo de **aprendizaje auto-supervisado** o **semi-supervisado** se encargaría de filtrar esa información en base a una serie de indicadores como viralidad (una métrica que variaría en función del canal) o impacto social (en función de los términos usados en el posible 'claim' y llevar a cabo una primera clasificación. A continuación, mediante aprendizaje **por refuerzo**, a modo de sistema de recomendación, el periodista podría seguir entrenando al algoritmo para que pueda afinar y mejorar los resultados obtenidos en esta escucha activa. Una vez localizado un contenido verificable, la interfaz permitiría enviar dicho contenido directamente a la plataforma interna que la entidad verificadora utilice, preferentemente también vía API (Slack, Telegram, hoja de cálculo, correo…). De manera escalada, podría plantearse la conveniencia de un sistema **multi-usuario**, con diferentes recomendaciones según el verificador o la temática (hay equipos especializados en temáticas concretas, por lo que la monitorización podría adaptarse a cada necesidad).

El desarrollo sería **progresivo y modular**, de manera que pueda comenzarse por una sola fuente e ir completando las funcionalidades de la herramienta con la incorporación de nuevos módulos para los canales. Se prevé que, con las primeras iteraciones funcionales, pueda abrirse a la comunidad de periodistas verificadores una primera beta para el testeo por parte de profesionales (publicación en cerrado con identificación de usuarios). La idea a medio o largo plazo es la **publicación en abierto** de la herramienta, ya sea como plugin en navegadores o como página web publicada en abierto.

En consonancia, un posible nombre propuesto para esta herramienta, obviamente sujeto a cambios, sería ***SoundHive***, que hace referencia a la escucha (*sound*) con el concepto de colmena (*hive*), es decir, la escucha activa del sonido de la colmena, la monitorización, por parte de los verificadores de información, de la conversación social que se produce en internet, para ser capaces de distinguir las posibles señales desinformadoras del ruido diario que se produce en redes.

Para este primer desarrollo, se centrarán los esfuerzos en la plataforma Telegram. Es una de las aplicaciones de mensajería instantánea que ha ganado una enorme popularidad en los últimos años. Con 700 millones de usuarios mensuales activos en todo el mundo, Telegram se ha visto siempre relegada a la sombra de otras aplicaciones como WhatsApp (2.000 millones), WeChat (1.309 millones) o Facebook Messenger (931 millones de usuarios), según datos de Statista a enero de 2023. Pero ha compensado esa falta de implantación y número de usuarios con una adaptabilidad mayor,

Si bien Telegram se ha destacado por sus características de privacidad y seguridad, también ha tenido un impacto significativo en la propagación de la desinformación en línea. En este trabajo, examinaremos la trascendencia de Telegram para la desinformación, analizando cómo esta plataforma ha influido en la forma en que la información errónea se difunde y los desafíos que plantea.

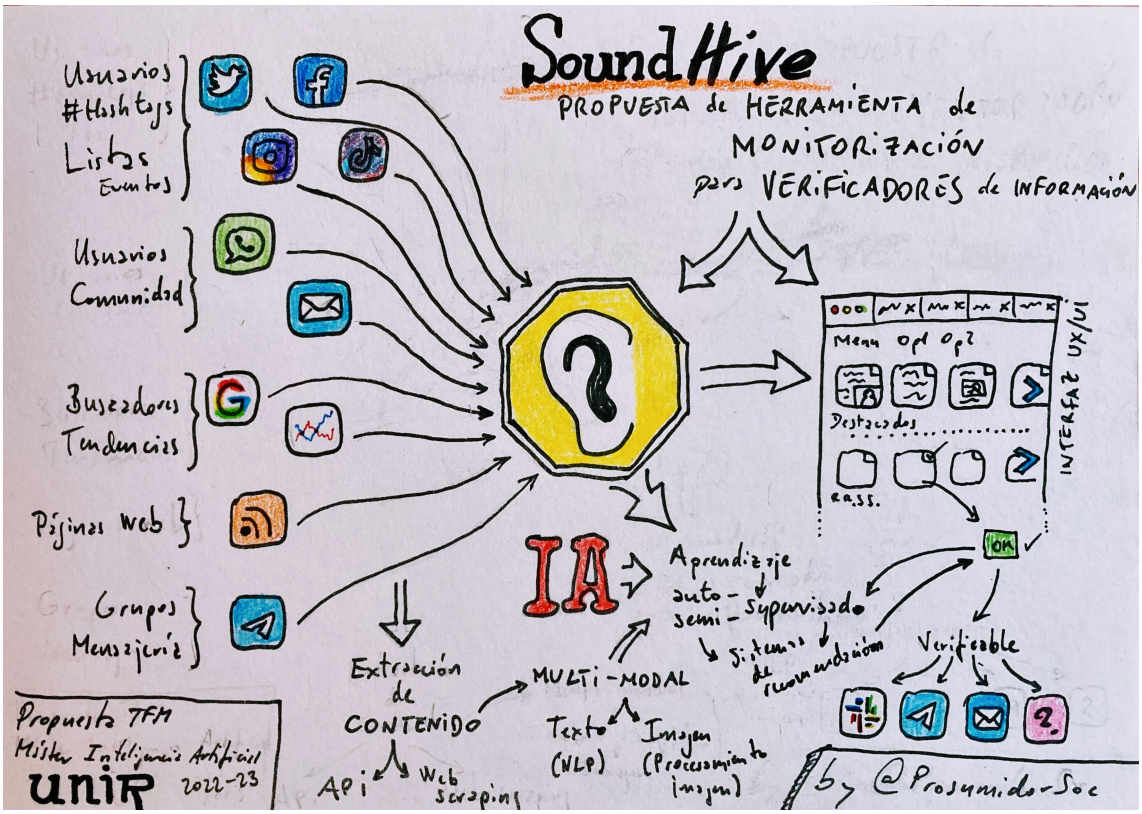

Figura 1.- Sketchnote resumen de la propuesta de SoundHive. Elaboración propia.

1.3. ESTRUCTURA DE LA MEMORIA

En el próximo capítulo, sobre el contexto y estado del arte, se ofrece una visión profunda sobre la desinformación y la posverdad, el periodismo de verificación, y cómo la inteligencia artificial y el aprendizaje automático están siendo utilizados en la verificación de hechos. Se destaca la plataforma Telegram y su papel en la desinformación, especialmente en España. Además, se revisan herramientas y *datasets* relevantes en el ámbito de estudio.

En el capítulo dedicado a los objetivos y metodología de trabajo se definen los objetivos generales y específicos del proyecto. Se presenta una metodología detallada que incluye la justificación, las fases del proyecto y las herramientas y técnicas utilizadas. Igualmente, en el siguiente capítulo, centrado en la identificación de requisitos se contextualiza el problema, se identifican los usuarios objetivo y se describen escenarios habituales. Se detalla la metodología utilizada para identificar los requisitos, que incluye la participación de expertos, entrevistas y observación participante. Finalmente, se listan los requisitos identificados.

A continuación, se describe la herramienta *software* desarrollada, así como el proceso de desarrollo de la herramienta, desde el diseño y prototipado hasta la propuesta de lanzamiento y monitorización. Se detalla la arquitectura de la herramienta, sus componentes principales y el flujo de datos. Además, se explica el funcionamiento de la herramienta, incluyendo la extracción de datos de Telegram y el procesamiento y análisis de estos datos.

En el capítulo dedicado a la evaluación se propone un modelo para evaluar la herramienta desde diferentes perspectivas: usabilidad, aplicabilidad y la opinión de expertos. Se proponen metodologías para cada tipo de evaluación, se describen los participantes propuestos y se anticipan los resultados esperados. Finalmente, en las conclusiones y trabajo futuro se resumen las principales conclusiones derivadas del proyecto y se proponen líneas de trabajo futuro para mejorar y expandir la herramienta desarrollada.

2. CONTEXTO Y ESTADO DEL ARTE

2.1. DESINFORMACIÓN Y POSVERDAD

El fenómeno de la **desinformación**, si bien no es nuevo en la Historia de la Humanidad, sí ha adoptado, en una nueva época tecnológica y cultural, caracterizada por la aparición del fenómeno de la **posverdad**, nuevas y peligrosas formas de desarrollo y expansión (Kakutani, 2018; Lewandowsky et al., 2017). Fenómenos como las mal llamadas 'fake news'[1] se enmarcan en lo que expertas como Claire Wardle (2017) consideran 'desórdenes de la información' (information disorders), en los que podemos diferenciar la 'disinformation' o información engañosa, de la 'misinformation' o información errónea y la 'malinformation' o información improcedente. El reciente 'I Estudio sobre Desinformación en España', realizado por UTECA y la Universidad de Navarra, (Sádaba-Chalezquer & Salaverría-Aliaga, 2022) muestra que la **preocupación social** en torno a la desinformación en España es amplia: un 96% de la población lo considera un problema y un 91%, un peligro para la democracia y la estabilidad nacional. Un 72% reconoce que en algún momento ha dado como cierto algún tipo de contenido falso.

[1] 'Fake news', a pesar de su extendido uso, resulta ya un término denostado por los estudiosos en la materia y que prefiero no usar en detrimento de 'desinformación', debido, por un lado, al abuso del término por parte de conocidos desinformadores como Donald Trump y a que si un contenido informativo, por definición, no puede ser una falsedad. Hablaremos entonces de contenidos falsos o desinformadores, pero no informativos. Las mentiras no informan, engañan.

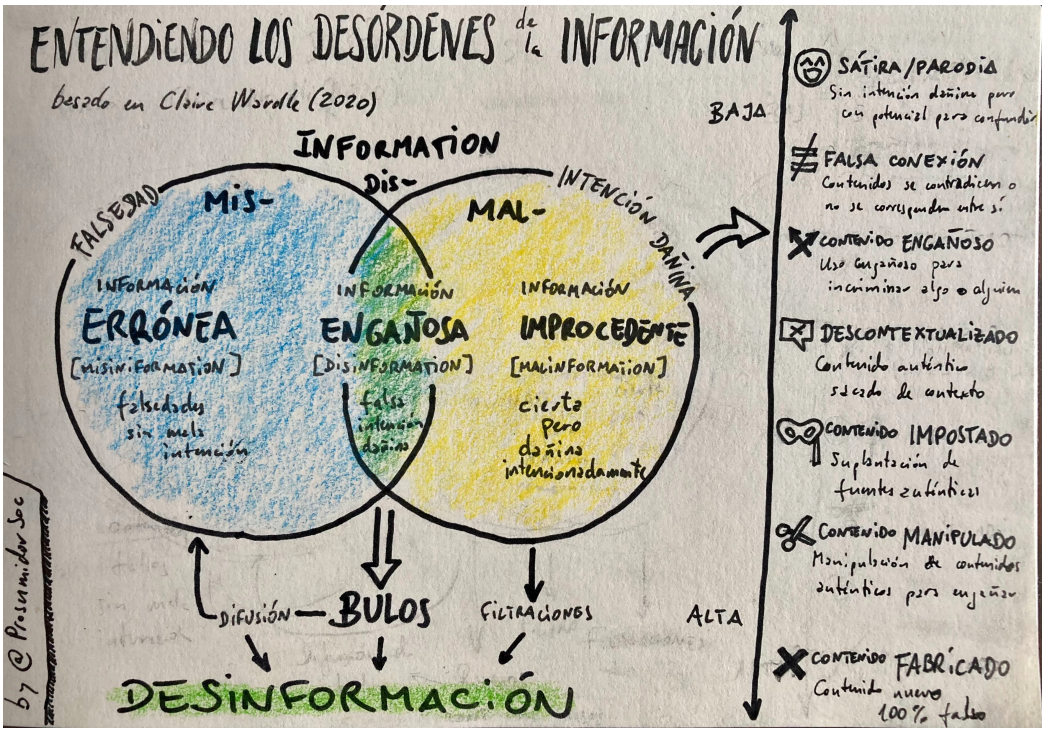

Figura 2.- Tipología de desórdenes de la información, según Claire Wardle, según la traducción al español propuesta por Ramón Salaverría. Elaboración propia.

En estos entornos infoxicados, inundados de información, se torna más complicado que nunca discernir la información fiable de la falsa, manipulada o descontextualizada. Por eso es más importante la labor de una especialización del periodismo producto de esta nueva situación: la **verificación de información**, una rama del periodismo especializada en comprobar la veracidad de contenidos sospechosos que circulan por internet, algunos de ellos con un alto riesgo de peligrosidad social y viralidad emocional. La verificación de información, o *fact-checking*, supone un **reto para el periodismo actual** y se puede entender desde una perspectiva doble: son una serie de nuevas competencias para los periodistas pero también una **nueva disciplina de especialización** para los periodistas (Ufarte-Ruiz et al., 2018).

2.2. PERIODISMO DE VERIFICACIÓN Y *FACT-CHECKING*

Definimos el *fact-checking* como un tipo de periodismo dedicado a la comprobación de hechos y afirmaciones (Graves et al., 2016) mediante el uso de técnicas básicas de investigación periodística y, en ocasiones, de técnicas avanzadas que requieren el uso de métodos computacionales. Pero la comprobación de hechos es una parte indisoluble y fundacional del propio periodismo. El *fact-checking* es, por tanto, la potenciación y especialización en una

disciplina separada de una de las capacidades y responsabilidades fundamentales de todo periodista: comprobar y verificar que lo que se está contando al gran público tiene visos de certidumbre, de verdad.

La expansión y el aumento en las capacidades de manejo de información que dieron origen al fenómeno conocido como Big Data influenció todas las esferas sociales, también al periodismo. Las primeras propuestas para automatizar de alguna manera el *fact-checking* online aparecieron hace más de 15 años (Graves, 2018), junto con diversas iniciativas para incentivar y explorar la investigación en esta línea. Ya entonces, a pesar de identificar los elementos del flujo de trabajo del *fact-checking* susceptibles de automatizarse, los teóricos tenían claro que conseguir una automatización completa era prácticamente imposible, por la dificultad implícita del propio trabajo del *fact-checker* y el juicio crítico, sensibilidad y experiencia necesarias para tomar una decisión que, como dijimos antes, no es simplemente binaria. Las técnicas de Inteligencia Artificial tendrían que ser una ayuda, no un sustituto, para el periodista dedicado a la verificación de información.

No obstante, la elección de Trump como presidente aceleró esta línea de investigación.

La comprobación de hechos automatizada se consideraba una gran solución para combatir la desinformación, pero informes recientes indican que los verificadores de hechos desconfían en general de los métodos totalmente automatizados (Arnold, 2020). Aunque la rápida difusión de información falsa presenta problemas de escalabilidad para la comprobación humana de hechos (Vosoughi et al., 2018), esto no debería socavar la complejidad y precisión del proceso de comprobación de hechos. Por lo tanto, el foco de la investigación se ha desplazado hacia métodos híbridos que ayudan a los fact-checkers. Hasta ahora, según Nakov et al. (2021), las aplicaciones de ayuda a la comprobación de hechos se han desarrollado para cuatro tareas principales: identificar afirmaciones que merecen ser comprobadas, encontrar afirmaciones previamente verificadas, recuperar pruebas relevantes para la comprobación y la comprobación automática.

La comprobación de la veracidad de las afirmaciones se ha abordado mediante diversos métodos en investigaciones anteriores. Algunos lo han planteado como una tarea de clasificación, dando prioridad a las afirmaciones que deben comprobarse mediante una predicción de puntuación. Otros han utilizado una tarea de clasificación, en la que el sistema

predice si una afirmación es objetiva, numérica o se basa en creencias personales. Alternativamente, se ha utilizado una tarea de respuesta a preguntas, asumiendo que existen afirmaciones verificadas en el conjunto de datos. Sin embargo, algunos trabajos ponen de relieve que los sistemas automatizados pueden introducir sesgos en la elección de las afirmaciones para la comprobación de hechos, y sugieren que el desarrollo de herramientas como las alertas de noticias, el reconocimiento de voz y los modelos de traducción pueden ayudar a los verificadores de hechos a filtrar las afirmaciones con mayor eficacia.

La detección de afirmaciones verificadas anteriormente implica encontrar afirmaciones verificadas en el pasado, así como afirmaciones verificadas en otros idiomas o países. Para lograrlo, algunos investigadores han desarrollado un método de aprendizaje que compara las nuevas afirmaciones con las verificadas. Facter-Check, por ejemplo, utiliza la similitud semántica textual para abordar esta tarea en español.

La recuperación de pruebas ha sido otro campo de investigación en la comprobación semiautomática de hechos. Algunos investigadores se han centrado en extraer el conjunto de pruebas más valioso de una base de datos predeterminada de fuentes fiables. Esto se ha llevado a cabo tanto a nivel de documento como de frase, así como con datos estructurados en tablas. Otras herramientas útiles para la recuperación de pruebas son el reconocimiento de voz, la búsqueda inversa de imágenes y la búsqueda de consultas complejas.

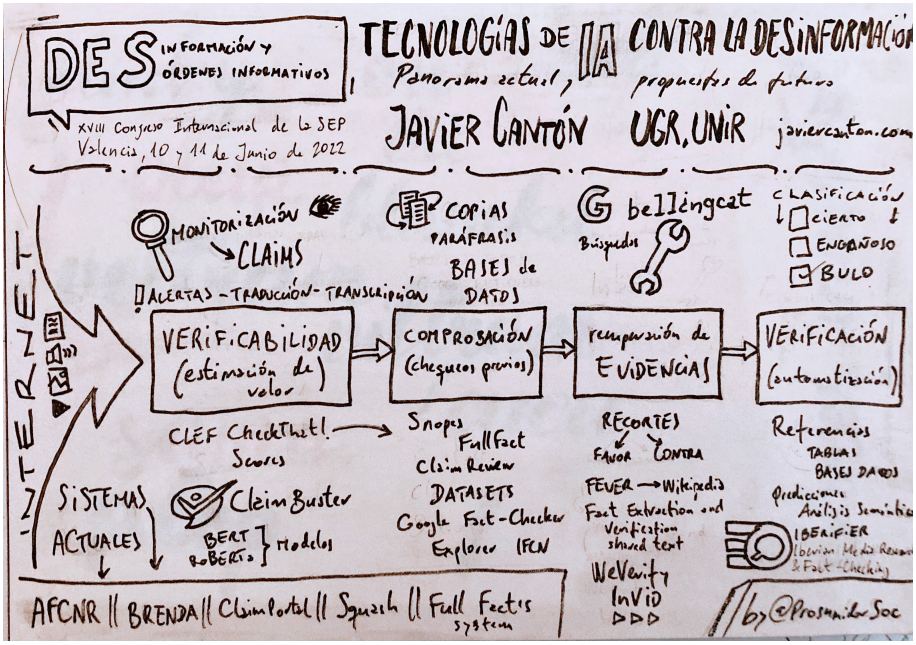

Figura 3.- Distintas tecnologías actuales basadas en IA usadas contra la desinformación, en función del momento en el flujo de trabajo de los verificadores en el que se usan. Elaboración propia.

2.3. INTELIGENCIA ARTIFICIAL PARA LA VERIFICACIÓN

El uso de **herramientas de inteligencia artificial** va extendiéndose poco a poco entre los verificadores de información, no solo en el ámbito anglosajón, también en el latinoamericano, con diferentes iniciativas y experiencias dentro del consorcio IFCN (International *Fact-checking* Network). Existen diferentes iniciativas centradas en alguna de las fases del flujo de trabajo de la verificación de información: desde la **verificabilidad** del contenido falso, a la **comprobación** de la existencia previa de desmentidos sobre la afirmación a verificar o 'claim', la **recuperación** de evidencias para la verificación (la investigación propiamente dicha, con profuso uso en ocasiones de herramientas OSINT), y la **clasificación** en alguna de las categorías existentes del desmentido publicado (Fco. J. Cantón-Correa, 2022).

Sin embargo, los proyectos innovadores existentes suelen centrarse en los procedimientos propios de la verificación, como la investigación de evidencias, la gestión de bases de datos, la respuesta automatizada de peticiones ciudadanas o, las menos, en la generación automática de contenido (Sánchez González et al., 2022). Una de las tareas para las que no existen herramientas específicas es la de la **monitorización de información**: la escucha activa de espacios digitales para la búsqueda de 'claims' o contenidos falsos verificables. Si bien

existen herramientas como Factmata, que realizan una escucha activa similar, estas están más enfocadas al campo del marketing y la analítica de social media, normalmente para la monitorización de la conversación global, la calidad de contenidos y la evaluación de amenazas para marcas y empresas.

La **escucha activa en verificación** normalmente se centra en fuentes concretas, establecidas previamente por el equipo de verificación: seguimiento de listas, usuarios y *hashtags* concretos en redes sociales, recepción de materiales y contenidos a través de los medios de contacto con las diferentes entidades dedicadas a la verificación, observación de tendencias generales en buscadores, seguimiento de publicaciones en webs consideradas desinformadoras e infiltración en canales públicos concretos en plataformas de mensajería. Estas tareas suelen consumir gran parte del tiempo de trabajo de los periodistas de verificación, además de que produce tal saturación y fatiga mental (por la dureza en algunos casos de los contenidos falsos a los que se ve expuesto el periodista) que suele rotarse la realización de dicha tarea entre las diferentes personas del equipo de trabajo.

2.3.1. Un ejemplo aplicado: ClaimCheck, la herramienta de Newtral

ClaimCheck (Newtral, 2022) es una herramienta innovadora desarrollada en colaboración entre Newtral.es y ABC Australia. Esta herramienta nació como parte del JournalismAI Fellowship Programme, que fue impulsado por la London School of Economics y el Knight Center. El principal objetivo de ClaimCheck es combatir mentiras y desinformaciones lanzadas en discursos y declaraciones públicas.

La herramienta utiliza técnicas de inteligencia artificial para detectar cuando se repite una afirmación, incluso si se utiliza un lenguaje o expresiones diferentes. Para lograr esto, se emplea un algoritmo especializado en detectar similitud semántica y rescatar verificaciones previas. Este sistema no solo identifica palabras clave que se repiten, sino que también analiza estructuras semánticas, la sintaxis de las oraciones y otros elementos que componen las afirmaciones.

ClaimCheck está conectado a los mensajes que publican numerosos políticos en X (anteriormente Twitter). Si detecta que una frase se repite o coincide con una verificación anterior o con otro tuit, envía una alerta a los periodistas para su revisión. Esto permite una

rápida respuesta ante la difusión de desinformación. La herramienta utiliza una base de datos compuesta por más de 250.000 verificaciones en diferentes idiomas, que se emplea para identificar mentiras recurrentes.

Además, la herramienta es multilingüe, lo que amplía su alcance y permite detectar mentiras repetidas en otros idiomas. Esto es especialmente útil para temas globales, como la pandemia del coronavirus o conflictos internacionales como la guerra en Ucrania. En estos contextos, el algoritmo puede identificar una mentira que ya ha sido refutada por fact-checkers en otros países. Salvando las distancias, y en otra escala, esta herramienta sirve de ejemplo de las posibilidades de las técnicas de IA y NLP aplicadas a la lucha contra la desinformación.

2.4. APRENDIZAJE AUTOMÁTICO PARA EL ANÁLISIS DE DESINFORMACIÓN

2.4.1. Identificación de desinformación mediante aprendizaje supervisado

La identificación automática de la desinformación mediante aprendizaje supervisado es uno de los enfoques más utilizados en el ámbito de la IA. Este método modela el problema como una clasificación binaria, es decir, determinar si una información es veraz o no. Para ello, se toma un conjunto de características representativas de un elemento de información y se predice su veracidad. La selección de estas características y la manera de combinarlas pueden hacerse de forma manual o automática. En el enfoque manual, se puede aplicar la toma de decisiones multicriterio para definir criterios y pesos de probabilidad con los que calcular una puntuación de credibilidad de la información (Pasi et al., 2020; Viviani & Pasi, 2017). Por otro lado, en el enfoque automático, se utiliza el aprendizaje automático para aprender de forma autónoma las características y los pesos con los que clasificar la veracidad de la información (López et al., 2019; Molina-Solana et al., 2018).

Sin embargo, es importante destacar que la desinformación no se presenta en términos absolutos de blanco o negro, sino que se encuentra en un amplio espectro de grises. En la literatura, se han propuesto etiquetas más detalladas para captar estos matices sutiles entre diferentes grados de desinformación. Por ejemplo, Wang (2017) elaboró un conjunto de datos anotado manualmente con seis etiquetas, evaluando el grado de veracidad de miles de afirmaciones con categorías como mentira, falsedad, algo de cierto, media verdad, mayormente cierto y verdad. Nakamura, Levy y Wang (2020) utilizaron una jerarquía de

etiquetado de dos, tres y seis categorías para cada muestra de su conjunto de datos multimodal, lo cual permitía la implementación de modelos de clasificación a diferentes niveles de granularidad.

El rendimiento de los métodos de aprendizaje supervisado depende directamente de la calidad de los datos etiquetados. Dado que estos datos suelen capturar situaciones y eventos específicos, la aplicación de estos modelos a otros dominios puede no ser efectiva (Shu et al., 2017).

Los métodos de detección de desinformación requieren de características representativas y relevantes los elementos que se van a analizar para ser efectivos. Tradicionalmente, las características empleadas para la clasificación de desinformación se han dividido en dos categorías: basadas en el contenido y basadas en el contexto (Bondielli & Marcelloni, 2019): las características basadas en el contenido son atributos relevantes extraídos directamente del ítem desinformativo, normalmente un texto que afirma o apoya el posible engaño, y que a menudo suele estar asociado con imágenes o vídeos que lo refuerzan; las características basadas en el contexto se refieren a los datos o metadatos que rodean el ítem desinformativo.

2.4.2. Procesamiento del Lenguaje Natural (NLP) para caracterización estilística

La caracterización estilística de mensajes mediante el procesamiento del lenguaje natural (PLN) es una herramienta esencial en la lucha contra la desinformación. Esta técnica se centra en analizar las características lingüísticas de la información, abarcando tanto aspectos sintácticos como semánticos (Ruffo et al., 2023). Desde una perspectiva sintáctica, se subraya la relevancia del etiquetado gramatical y la identificación de grupos de palabras, como bigramas, trigramas o n-gramas. Estos elementos son cruciales para desentrañar la estructura y el flujo del contenido, permitiendo una comprensión más detallada de los mensajes (Zhou et al., 2020).

En el ámbito semántico, se utilizan herramientas como el análisis de sentimientos, la detección de temas y las codificaciones con *word embeddings*. Estas técnicas ofrecen una visión profunda del significado y el contexto del contenido, proporcionando una perspectiva más rica sobre la información presentada. Un aspecto destacado es el análisis basado en el estilo de escritura. Se ha observado que los actores maliciosos emplean estilos distintivos al redactar

contenidos desinformativos. Para identificar estos estilos, se consideran criterios como la frecuencia de patrones morfológicos específicos (Castelo et al., 2019), la presencia de elementos estructurales en el texto (Bonet-Jover et al., 2021), la variedad léxica y el uso particular de símbolos de puntuación (Azevedo et al., 2021). Estas características estilísticas sirven como indicadores para detectar y contrarrestar la desinformación en diversos medios.

2.4.3. Aspectos contextuales de la generación y diseminación de desinformación

En las redes sociales, el contexto en el que se presenta y se difunde la desinformación es esencial para su identificación y comprensión. Las características del usuario, como el número de publicaciones, seguidores, datos demográficos, verificación de la cuenta y antigüedad, pueden ofrecer pistas sobre la credibilidad del emisor y, por ende, sobre la veracidad de la información compartida (Shu et al., 2019). La credibilidad de un usuario puede ser un indicador crucial para determinar la probabilidad de que comparta contenido falso o engañoso.

El contenido de los mensajes y su contexto también son vitales. Los metadatos asociados a las publicaciones y los recursos multimedia que las acompañan pueden enriquecer el análisis y ofrecer una perspectiva más detallada sobre la naturaleza de la desinformación (Della Vedova et al., 2018; B. Guo et al., 2020). Estos elementos, cuando se analizan en conjunto, pueden revelar patrones y tendencias que ayuden a identificar y combatir la desinformación de manera más efectiva.

Además, la estructura de la red social en sí misma, incluyendo la forma en que los usuarios están interconectados y cómo se propaga la información, puede proporcionar *insights* valiosos sobre la difusión de la desinformación. En conjunto, al considerar estos aspectos contextuales en las redes sociales, es posible obtener una comprensión más profunda y precisa de la desinformación, permitiendo desarrollar estrategias más efectivas para detectarla y combatirla.

2.4.4. *Fact-checking* semi-automático: un enfoque “human-in-the-loop”

La verificación de hechos, o *fact-checking*, es la rama del periodismo que se centra en la comprobación de afirmaciones públicas (Graves et al., 2016). Aunque la verificación de

información siempre ha sido una parte esencial del periodismo, el auge de las redes sociales y la velocidad con la que se difunde la información han hecho que el *fact-checking* adquiera una relevancia especial. Sin embargo, la automatización completa de este proceso es prácticamente imposible debido a la necesidad de juicio crítico, sensibilidad y experiencia para tomar decisiones que no sean meramente binarias (Arnold, 2020).

Dada la rapidez con la que se difunde la desinformación, se reconoce que desacreditar una mentira lleva más tiempo que difundirla (Vosoughi et al., 2018). Por ello, la tendencia actual se inclina hacia la verificación asistida por herramientas de inteligencia artificial en lugar de una comprobación de hechos completamente automatizada. Estos sistemas, denominados "human-in-the-loop" (personas en el proceso), integran la capacidad humana de juicio con las capacidades de procesamiento y análisis de la IA (La Barbera et al., 2022; Shabani et al., 2021; Yang et al., 2021).

Las técnicas de IA pueden apoyar la verificación de información en diferentes fases del flujo de trabajo de verificación, desde la detección inicial de afirmaciones hasta la comprobación final y la publicación de resultados (Z. Guo et al., 2022; Nakov et al., 2021). Estas herramientas, combinadas con la experiencia y el juicio humano, ofrecen una solución más robusta y eficiente para combatir la desinformación en la era digital.

2.4.5. Contenidos generados por ordenador y desinformación

La generación automática de desinformación mediante modelos masivos de lenguaje, como GPT-3 y ChatGPT, representa uno de los desafíos más significativos en la actualidad. Estos modelos tienen la capacidad de producir desinformación textual a gran escala y pueden ser utilizados de diversas maneras, como camuflar contenidos falsos bajo una apariencia de información real, crear bots y sitios web que perpetúan narrativas desinformativas, o incluso evadir detectores basados en características estilísticas (Solaiman et al., 2019). Una preocupación adicional es que, debido a la falta de un control riguroso sobre las fuentes utilizadas para entrenar estos modelos, muchos de los contenidos que generan pueden ser inexactos o sesgados (Marcus, 2022).

Además de los desafíos asociados con la desinformación textual, la generación automática de desinformación no se limita solo al texto. Existen técnicas avanzadas de inteligencia artificial

que pueden crear imágenes, videos y audios altamente realistas. Estos contenidos multimedia, conocidos como “ultrafalsificaciones” o “deepfakes”, son generados mediante técnicas como las redes generativas adversariales (Goodfellow et al., 2016). Estas ultrafalsificaciones pueden ser aún más perjudiciales que la desinformación textual, ya que su realismo puede engañar fácilmente al ojo y al oído humanos.

Dada la capacidad de estos modelos para generar desinformación de manera efectiva y convincente, es esencial desarrollar métodos robustos para detectar y contrarrestar la desinformación generada por ellos. Sin embargo, los esfuerzos realizados hasta ahora para abordar este problema han demostrado ser insuficientemente efectivos (Mitchell et al., 2023). Por lo tanto, la lucha contra la desinformación generada automáticamente sigue siendo un área de investigación activa y crítica.

2.5. TELEGRAM Y LA DESINFORMACIÓN

Telegram es una aplicación de mensajería instantánea fundada en 2013 por los hermanos rusos Nikolái y Pável Dúrov. Es una de las aplicaciones de mensajería instantánea que ha ganado una enorme popularidad en los últimos años. Con 700 millones de usuarios mensuales activos en todo el mundo, Telegram se ha visto siempre relegada a la sombra de otras aplicaciones como WhatsApp (2.000 millones), WeChat (1.309 millones) o Facebook Messenger (931 millones de usuarios), según datos de Statista a enero de 2023. Pero ha compensado esa falta de implantación y número de usuarios con una adaptabilidad mayor o unos mayores niveles de encriptación y seguridad para las conversaciones privadas.

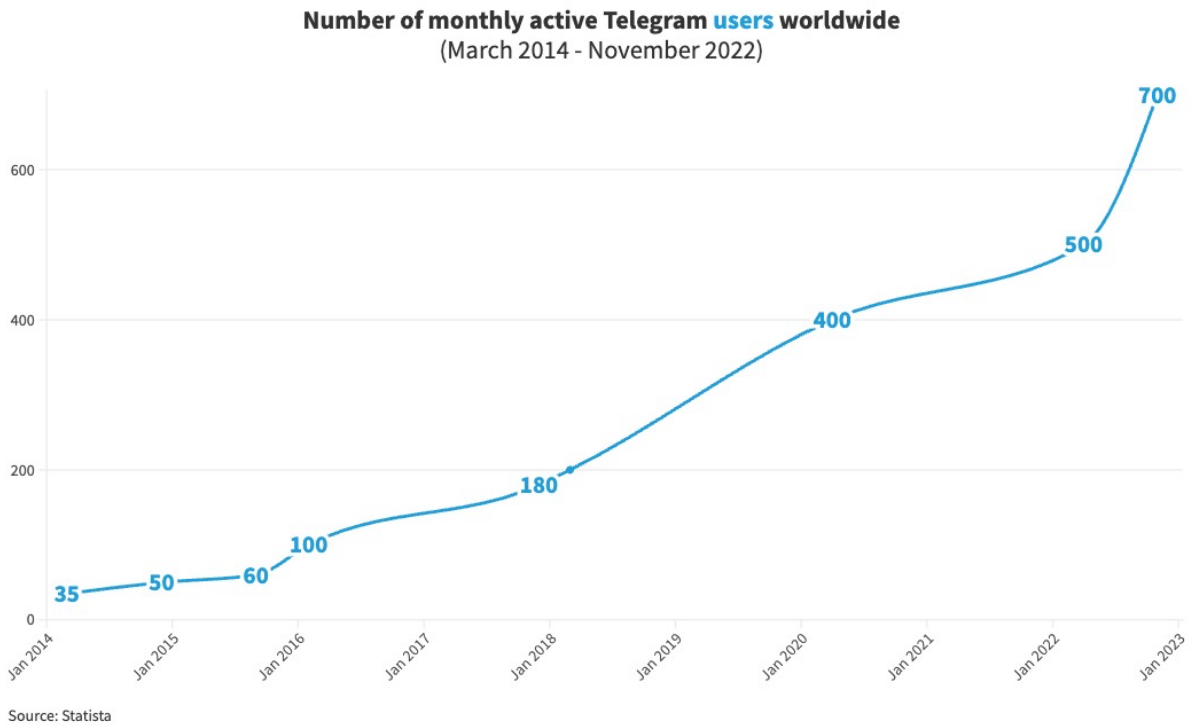

Figura 4.- Evolución en el número de usuarios de Telegram (marzo de 2014-noviembre 2022). Fuente: Statista. Versión web disponible.

Algunas de las posibles causas por las que un usuario podría preferir Telegram frente al resto de productos de la competencia (aspectos que la propia compañía ha potenciado para su propia promoción) podrían ser:

1. Privacidad y seguridad: Telegram se destaca por su enfoque en la privacidad y seguridad. Ofrece una encriptación de extremo a extremo opcional en las conversaciones, lo que significa que los mensajes se pueden cifrar para que solo los remitentes y destinatarios puedan leerlos. Además, Telegram permite la creación de chats secretos que cuentan con una mayor capa de seguridad, ya que los mensajes se autodestruyen después de un cierto período de tiempo y no se pueden reenviar. Telegram también ofrece opciones avanzadas de autenticación de dos factores y no almacena datos del usuario en sus servidores, lo que brinda mayor confidencialidad en comparación con WhatsApp o Messenger.

2. Flexibilidad en el uso multiplataforma: Telegram está disponible en una amplia variedad de plataformas, incluyendo dispositivos móviles (Android, iOS), computadoras de escritorio (Windows, macOS, Linux) y navegadores web. Esto

permite a los usuarios acceder a sus conversaciones y archivos desde diferentes dispositivos de manera sincronizada, lo que resulta conveniente para aquellos que necesitan utilizar Telegram en varios dispositivos.

3. Capacidad de almacenamiento ilimitado: Telegram permite a los usuarios enviar y recibir archivos de gran tamaño, incluyendo fotos, videos, documentos y otros tipos de archivos, con un límite de tamaño de archivo de hasta 2 GB. Además, ofrece almacenamiento en la nube gratuito e ilimitado para guardar archivos y mensajes, lo que lo convierte en una opción atractiva para aquellos que necesitan compartir y almacenar grandes cantidades de datos.

4. Funciones de chat en grupo avanzadas: Telegram ofrece una amplia gama de funciones para los chats en grupo, lo que lo hace especialmente popular para comunidades y grupos de trabajo. Los grupos de Telegram pueden tener hasta 200.000 miembros, lo que permite una comunicación masiva. Además, Telegram ofrece opciones para administradores de grupo, como la posibilidad de fijar mensajes importantes, crear encuestas, programar publicaciones, moderar miembros y establecer roles, lo que permite una gestión eficiente de los grupos.

5. Personalización y control de notificaciones: Telegram permite a los usuarios personalizar su experiencia de uso al ofrecer una amplia gama de opciones de configuración, como la posibilidad de cambiar la apariencia de la interfaz, personalizar los tonos de notificación, habilitar o deshabilitar las notificaciones de chats o grupos específicos, y silenciar chats o grupos temporalmente. Esto proporciona un mayor control sobre las notificaciones y permite adaptar la aplicación a las preferencias individuales de cada usuario.

6. Amplia gama de bots y complementos: Telegram cuenta con una gran cantidad de bots y complementos desarrollados por la comunidad, que ofrecen una amplia variedad de funciones adicionales, como la traducción de mensajes, la generación de memes, la gestión de tareas, la consulta de noticias, y muchas más. Esto brinda una experiencia de uso más enriquecedora y personalizada.

Estas características, especialmente las referentes a privacidad, como la capacidad de enviar mensajes encriptados y auto destruibles, ha llevado a un aumento en su popularidad entre aquellos que buscan una mayor seguridad en su comunicación en línea. Sin embargo, paradójicamente, esta misma privacidad también ha permitido la propagación de la desinformación en la plataforma.

Figura 5.- Diferentes muestras de la interfaz de usuario de la aplicación de mensajería instantánea Telegram. Su facilidad de uso y la configuración de bots para simplificar tareas son algunas de las claves del éxito de la aplicación. Fuente: Adobe Stock.

Una de las formas en que Telegram ha influido en la propagación de la desinformación es a través de los canales y grupos de Telegram. Los canales de Telegram son una herramienta que permite a los administradores difundir información a un gran número de seguidores de manera rápida y eficiente. Los grupos de Telegram, por otro lado, permiten la creación de comunidades en línea donde los usuarios pueden unirse y compartir información con otros miembros. Estos canales y grupos de Telegram han sido utilizados por actores malintencionados para difundir información errónea y desinformación en una escala masiva.

Uno de los mayores desafíos en la lucha contra la desinformación en Telegram es la dificultad para verificar la veracidad de la información compartida. A diferencia de otras plataformas de redes sociales, Telegram no tiene políticas de verificación de hechos o mecanismos de control de contenido integrados. Esto ha permitido que la desinformación se propague sin restricciones, lo que dificulta la identificación de noticias falsas y la corrección de la información errónea.

Además, Telegram ha sido utilizado como una plataforma para la coordinación y planificación de campañas de desinformación. Los actores malintencionados pueden utilizar grupos y

canales de Telegram para coordinar la difusión de información falsa en línea, lo que hace que sea difícil rastrear y combatir la propagación de la desinformación en tiempo real. Esto ha llevado a la rápida propagación de noticias falsas y teorías de conspiración en la plataforma, lo que puede tener un impacto significativo en la opinión pública y socavar la confianza en las instituciones y en la información verificada.

2.6. DESINFORMACIÓN EN ESPAÑA A TRAVÉS DE TELEGRAM

En España, Telegram es la sexta red social más popular y se ha convertido en un caldo de cultivo para la propagación de la desinformación, debido principalmente a las pocas medidas que adoptan para atajar la desinformación en su plataforma y el potencial de intercambio sin trabas de mensajería cifrada. Los canales públicos permiten que los contenidos de cuentas no auténticas lleguen a grandes audiencias, alimentando redes de desinformación de gran alcance que suelen estar prohibidas en otras plataformas. En España, canales como Rafapal News y La Quinta Columna tienen 140.000 y 280.000 seguidores, respectivamente.

Investigaciones recientes realizadas por *fact-checkers* relevantes (Andrino, 2021; Giménez, 2022; Madrigal, 2021a, 2021b; Maldita, 2022) comienzan a demostrar la transformación que se ha producido en las narrativas de desinformación en los canales españoles a través de diversos temas con grupos de desinformación que se retroalimentan entre sí, amplificando sus mensajes a través de las redes de Telegram replicando o reenviando contenido a otros canales. En muchos casos, el contenido en estos canales parece surgir allí primero, antes de saltar a otras redes sociales.

Se necesitan más investigaciones que investiguen más a fondo cómo los canales de desinformación de Telegram y sus narrativas se relacionan entre sí, surgen y se transforman con el tiempo. Esto es aún más apremiante dada la amplia popularidad de la plataforma tanto en España como en Europa en su conjunto.

Tampoco parece haber iniciativas europeas de promoción y políticas públicas que aborden la cuestión de la desinformación en plataformas de mensajería privada como Telegram. Los recientes debates políticos de la UE en torno a la Ley de Servicios Digitales y el Código de Prácticas sobre Desinformación se han centrado únicamente en las redes sociales públicas, dejando las plataformas privadas en gran medida sin control. A pesar de los crecientes

llamamientos a la acción contra Telegram, el establecimiento de posiciones políticas claras para las partes interesadas sigue siendo problemático cuando se trata de la regulación de la desinformación en cuentas de mensajería privadas.

Otro aspecto importante de la trascendencia de Telegram para la desinformación es su alcance global. Telegram tiene una base de usuarios diversa y se utiliza en todo el mundo, lo que permite la propagación de la desinformación en diferentes idiomas y regiones. Esto hace que sea más difícil controlar y abordar la desinformación en Telegram, ya que puede tener un impacto en múltiples países y comunidades en línea.

2.7. ESTADO DEL ARTE

Quizás debido a que el auge de Telegram en número de usuarios no se ha producido hasta hace poco, el interés de la comunidad académica por investigar esta herramienta es relativamente reciente. Búsquedas sencillas en Google Scholar o Semantic Scholar de la coincidencia de los términos "Telegram" y "disinformation" arroja apenas unos 15.000 resultados (poco más de 2.500 con el término en español "desinformación"). Para hacerse una idea, baste decir que la búsqueda de los términos "desinformación" y "disinformation" arrojan, por sí solos, más de 100.000 y 220.000 resultados, respectivamente.

A través de la herramienta Research Rabbit se ha podido establecer una panorámica sobre los principales trabajos de este campo en los últimos años. El grafo resultante permite ver la escasez de artículos de importancia y la estructuración de los existentes en torno a varias áreas de interés. Las principales temáticas de los trabajos académicos realizados en los últimos años versan sobre diversas materias, pero principalmente:

1. Análisis técnicos sobre la plataforma, que buscan conocer determinadas características, así como el desarrollo de herramientas específicas diseñadas para capturar y analizar datos de Telegram. También estudios que se centran en las características técnicas y funcionales de Telegram como plataforma, analizando su estructura en cuanto a datos y mensajes virales (Dargahi Nobari et al., 2017, 2021). Dentro de este grupo, es destacable la publicación del Pushshift Telegram Dataset, uno de los pocos de este tipo existentes, del que se hablará a continuación. También es destacable el desarrollo de la herramienta 4CAT (Peeters & Hagen, 2022), pionera en

el establecimiento de una herramienta estandarizada para la extracción de datos de redes sociales, entre ellas Telegram.

2. Otro importante grupo de trabajos se centra en el uso de Telegram para el mundo del activismo y la protesta política, estudiando cómo se utilizan este tipo de plataformas de mensajería para la movilización social, como en el caso del movimiento anti-ELAB en Hong-Kong (Aleksandra Urman et al., 2021; Su et al., 2022). Este tipo de movilización político-social se ha visto especialmente activa en Telegram por parte de grupos de extrema derecha y afines a teorías conspiracionistas, en múltiples países, por lo que este tipo de investigaciones indaga en la organización, evolución y dinámica de estas redes (Bovet & Grindrod, 2022; Garrido et al., 2021; Schulze et al., 2022; Urman & Katz, 2022).
3. Por último, fuera del mundo académico, en el ámbito periodístico, sí son muchas las piezas, ya sea procedentes de *fact-checkers* (Giménez, 2022; Madrigal, 2021a, 2021b) o de medios generalistas (Andrino, 2021; F.-J. Cantón-Correa & VerificaRTVE, 2021; Chen & Roose, 2021), que se han ocupado de explicar el papel de Telegram en el mundo de la desinformación. Este tipo de piezas, en formato de reportaje en profundidad, analizan cómo se propaga la desinformación en Telegram a través de sus redes y el flujo de difusión de noticias falsas por ellas.

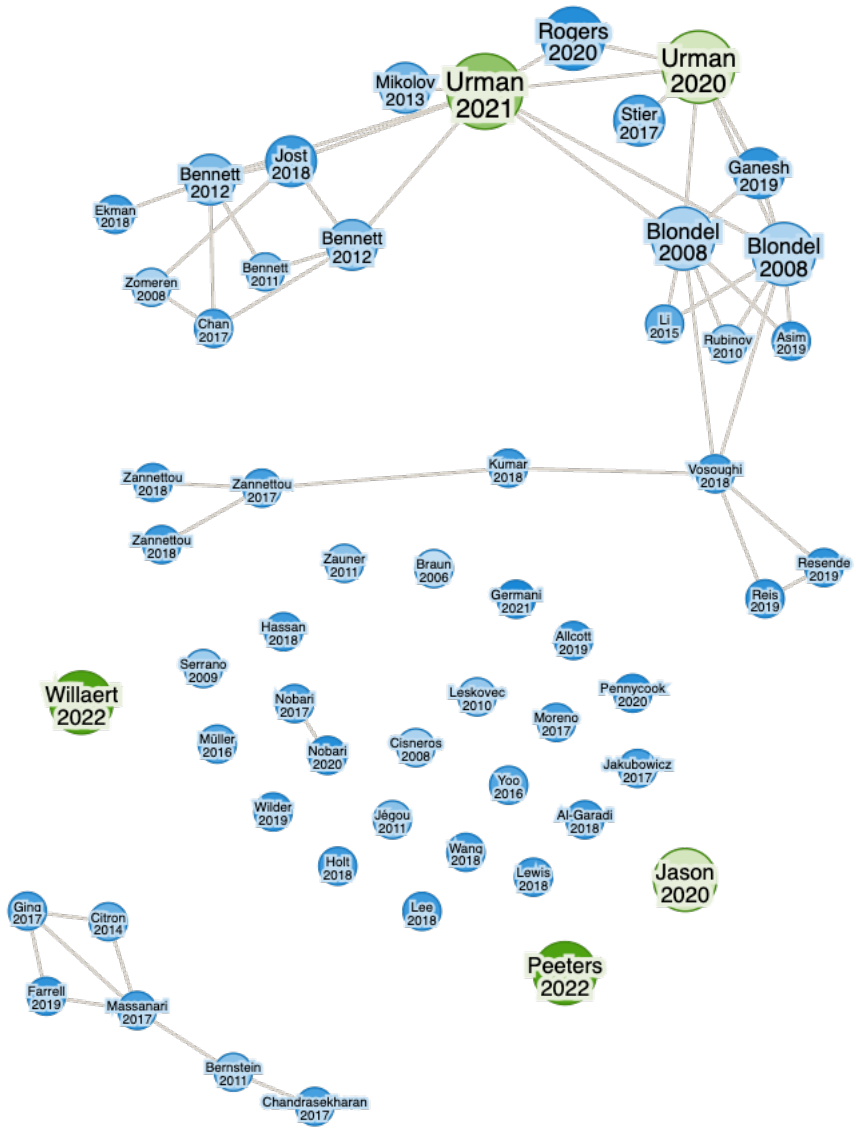

Figura 6.- Grafo de conexiones entre diversos trabajos académicos sobre Telegram y desinformación, aparecidos en los últimos años. Fuente: Research Rabbit.

2.7.1. Pushshift Telegram Dataset

El conjunto de datos denominado "Pushshift Telegram Dataset"[2], es una recopilación masiva de información de la plataforma de mensajería Telegram. Específicamente, contiene datos de

[2] Disponible en el repositorio abierto Zenodo: https://zenodo.org/record/3607497

más de 27.800 canales y 317 millones de mensajes enviados por 2,2 millones de usuarios únicos. Los autores han publicado este conjunto de datos con el objetivo de proporcionar a la comunidad de investigación un recurso valioso para estudiar diversos fenómenos sociales en línea. Dado que Telegram es una plataforma que ha sido utilizada para una variedad de propósitos, desde la organización de protestas hasta la propagación de desinformación y extremismo, un conjunto de datos de este tipo puede ofrecer *insights* sobre cómo se forman y propagan estos movimientos en línea. Al hacerlo público, los autores (Baumgartner et al., 2020) esperaban fomentar investigaciones en áreas como movimientos sociales, protestas, extremismo político y desinformación.

Para recopilar el conjunto de datos, los autores utilizaron técnicas de *web scraping* para extraer información de los canales públicos de Telegram. Aunque no se detallan todos los aspectos técnicos en el resumen, se menciona que han proporcionado el código fuente utilizado para la recopilación, lo que significa que otros investigadores pueden revisar y utilizar este código para ejecutar su propia instancia de recopilación de datos o para entender mejor el proceso de extracción.

Es importante destacar que, dado que el dataset se basa en canales públicos, respeta la privacidad de los usuarios al no incluir conversaciones privadas o información personal identificable. Además, antes de publicar cualquier conjunto de datos, es común que los investigadores procesen y anonimicen la información para garantizar que no se violen las normas de privacidad.

Este dataset, por tanto, es una herramienta valiosa para la comunidad de investigación que busca entender mejor el funcionamiento social dentro de la aplicación de mensajería, y ha supuesto también una aportación importante que se ha recogido en este trabajo.

2.7.2. 4CAT: Capture and Analysis Toolkit

Por otro lado, el 4CAT Capture and Analysis Toolkit[3] (Peeters & Hagen, 2022) es una herramienta de investigación basada en código abierto y diseñada para capturar y analizar

[3] Disponible en GitHub: https://github.com/digitalmethodsinitiative/4cat

datos de diversas fuentes en línea. Estas fuentes incluyen plataformas populares como Twitter, Telegram, Reddit, 4chan, 8kun, BitChute, Douban y Parler. La herramienta permite a los investigadores capturar datos y realizar análisis sin necesidad de tener habilidades avanzadas de programación. Además, 4CAT se ha diseñado con tres características clave en mente: modularidad, transparencia y trazabilidad.

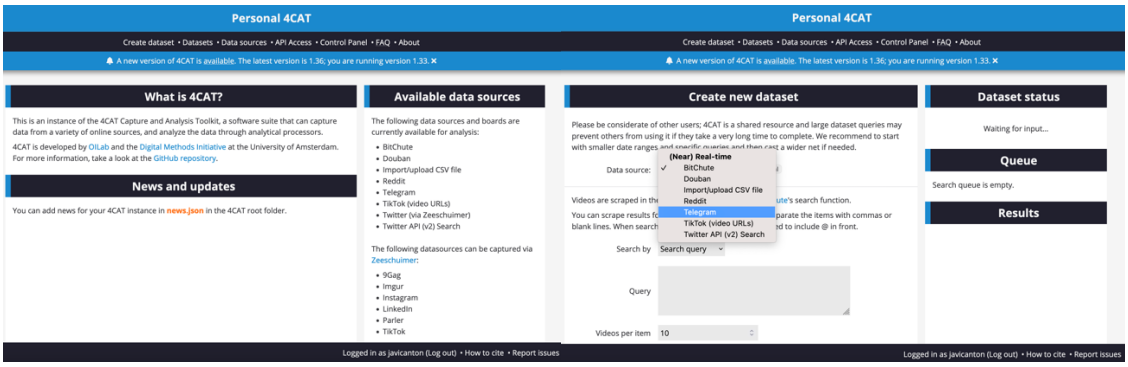

Figura 7.- Varias capturas de pantalla de la interfaz de la herramienta 4CAT. Fuente: elaboración propia.

Los autores han desarrollado y publicado 4CAT para abordar la necesidad de herramientas de investigación transparentes y trazables en el campo de la investigación social computacional. Reconocen que, aunque hay muchas herramientas disponibles para la captura y análisis de datos en línea, a menudo carecen de transparencia en sus métodos y no son accesibles para investigadores sin habilidades de programación. Al publicar 4CAT, los autores buscan ofrecer una herramienta que sea tanto robusta como transparente, permitiendo a los investigadores no solo capturar y analizar datos, sino también entender y comunicar claramente sus métodos de investigación.

4CAT opera en dos etapas principales: captura de datos y análisis. En la etapa de captura, los usuarios pueden seleccionar una fuente en línea y definir parámetros específicos para la recopilación de datos. Una vez recopilados, los datos se almacenan y están listos para el análisis. En la etapa de análisis, los usuarios pueden aplicar una variedad de "procesadores" a los datos capturados para realizar diferentes tipos de análisis. Estos procesadores pueden variar desde análisis de texto básico hasta visualizaciones complejas. La herramienta es modular, lo que significa que se pueden agregar fácilmente nuevas fuentes de datos y procesadores analíticos.

Al igual que muchas herramientas de captura de datos, 4CAT depende de las APIs proporcionadas por las plataformas en línea. Si una plataforma cambia su API o restringe el acceso, puede afectar la capacidad de 4CAT para recopilar datos de esa fuente. Dado que 4CAT puede capturar una amplia variedad de datos, los usuarios pueden necesitar cierto nivel de familiaridad con los tipos de datos y las estructuras para realizar análisis efectivos. Además, aunque 4CAT se esfuerza por una "ética por diseño", la captura y análisis de datos en línea siempre plantea preocupaciones éticas, especialmente en relación con la privacidad de los usuarios y el consentimiento para el uso de sus datos.

El auge de Telegram y su papel en la difusión de información y desinformación ha capturado la atención tanto de la comunidad académica como del mundo periodístico. Aunque la investigación académica sobre Telegram es aún incipiente en comparación con otros temas, es evidente que la plataforma está emergiendo como un área de estudio crucial. Los trabajos académicos se han centrado en aspectos técnicos, la movilización política y social, y la dinámica de las redes de extrema derecha y teorías conspirativas. Pero, por otro lado, el periodismo ha desempeñado un papel vital en arrojar luz sobre cómo se difunde la desinformación en Telegram.

A medida que Telegram continúa creciendo en popularidad, es esencial que se realicen más investigaciones y reportajes para comprender mejor su impacto en la sociedad y cómo se puede abordar la desinformación en la plataforma. Es un recordatorio de que, en la era digital, las plataformas de comunicación juegan un papel fundamental en la formación de la percepción pública y, por lo tanto, deben ser estudiadas y comprendidas a fondo.

Este trabajo forma parte del primer grupo de trabajos, centrados en el desarrollo de herramientas que permitan capturar y analizar la aplicación, pero pretende no perder el foco de que se persigue un fin mayor: el análisis de la desinformación dentro de la red para ahondar en la lucha contra las noticias falsas.

3. OBJETIVOS Y METODOLOGÍA DE TRABAJO

3.1. OBJETIVO GENERAL

El objetivo general de este TFM, por tanto, es prototipar un módulo para una herramienta de monitorización de desinformación centrado en Telegram que sea capaz de detectar posibles bulos mediante la selección de aquellos mensajes potencialmente dañinos y/o virales. Para ello, se perseguirán los siguientes objetivos específicos:

3.2. OBJETIVOS ESPECÍFICOS

- O1: Revisar la literatura científica y técnica más reciente en el campo de la IA aplicada a la lucha contra la desinformación, analizando los enfoques y técnicas utilizadas en investigaciones previas para detectar y clasificar información falsa en Telegram.
- O2: Desarrollar una herramienta de monitorización de desinformación en Telegram basada en técnicas de Inteligencia Artificial, que trate de detectar y clasificar automáticamente información falsa o engañosa en tiempo real en los mensajes y grupos de Telegram, utilizando algoritmos de procesamiento de lenguaje natural, aprendizaje automático y análisis de patrones de propagación.
- O3: Integrar en una herramienta de monitorización de desinformación: El módulo de detección de desinformación desarrollado se integrará en una herramienta de monitorización de desinformación existente o se desarrollará una nueva herramienta que lo incorpore. Esto implicará la integración del módulo en el flujo de trabajo de la herramienta, la interfaz de usuario, la configuración de opciones y la generación de informes. El objetivo es crear una herramienta completa y funcional que permita a los usuarios monitorizar la desinformación en Telegram de manera efectiva.

3.3. METODOLOGÍA

3.3.1. Introducción

En la era de la información digital, las plataformas de mensajería como Telegram se han convertido en fuentes primordiales de comunicación y difusión de información. Sin embargo, esta facilidad de difusión también ha llevado a la propagación de desinformación o “bulos”,

que pueden tener consecuencias perjudiciales para la sociedad. La necesidad de herramientas que puedan monitorizar, detectar y clasificar automáticamente la desinformación en tiempo real es más crucial que nunca.

El desarrollo de una herramienta de monitorización de desinformación en Telegram que utiliza técnicas de Inteligencia Artificial (IA) para abordar este desafío es una tarea intrincada. No solo implica la comprensión y aplicación de algoritmos avanzados, sino también la integración con plataformas de mensajería en tiempo real y la presentación de información de manera accesible para los usuarios. Además, el campo de la detección de desinformación es dinámico, con nuevos bulos emergiendo y evolucionando constantemente, lo que requiere que cualquier solución sea adaptable y evolutiva.

Dada la naturaleza compleja y dinámica del problema, se ha seleccionado una metodología **ágil**, específicamente el marco de trabajo **Scrum**, para el desarrollo del proyecto. Esta metodología, con su enfoque iterativo y adaptativo, es ideal para abordar los desafíos y objetivos del proyecto.

3.3.2. Justificación

La elección de una metodología ágil, en particular Scrum, se basa en varios factores clave que se alinean con la naturaleza y los objetivos del proyecto:

1. **Naturaleza dinámica del problema:** La desinformación es un fenómeno en constante evolución. Los bulos cambian, se adaptan y se propagan de formas nuevas y diferentes. Una metodología ágil permite al equipo adaptarse rápidamente a estos cambios, reevaluar prioridades y ajustar el enfoque del proyecto según sea necesario.

2. **Desarrollo incremental:** Los objetivos del proyecto son multifacéticos, desde la revisión de la literatura hasta el desarrollo y la integración de un módulo de detección. Scrum, con sus *sprints* y entregas incrementales, permite abordar cada objetivo paso a paso, garantizando que cada fase reciba la atención y el esfuerzo adecuados.

3. **Feedback continuo:** Una de las ventajas clave de Scrum es la retroalimentación continua. Al final de cada sprint, se puede obtener feedback tanto del equipo como de

los *stakeholders*. Esto es esencial para un proyecto como este, donde la eficacia de la herramienta debe ser probada y validada continuamente.

4. **Colaboración y comunicación:** Scrum fomenta la colaboración y la comunicación entre los miembros del equipo. Dado que este proyecto combina áreas de *expertise*, desde el procesamiento de lenguaje natural hasta el desarrollo web, es crucial que haya una comunicación fluida entre los expertos en cada área.

5. **Flexibilidad y adaptabilidad:** A medida que se avanza en el desarrollo, es probable que surjan nuevos desafíos o se identifiquen oportunidades de mejora. Scrum permite al equipo reaccionar a estos cambios, reorganizando prioridades y adaptando el plan de proyecto según sea necesario.

En resumen, la metodología ágil y el marco Scrum ofrecen la estructura, flexibilidad y enfoque centrado en el feedback que son esenciales para el éxito de un proyecto de esta magnitud y complejidad.

3.3.3. Fases del Proyecto

1. Planificación y definición de requisitos:
    a. Investigación preliminar: Antes de embarcarse en el desarrollo, se realiza una revisión exhaustiva de la literatura científica y técnica más reciente en el campo de la IA aplicada a la lucha contra la desinformación (Objetivo O1).
    b. Definición de requisitos: Basándose en la investigación, se definen los requisitos técnicos y funcionales del sistema. Esto incluye identificar las características clave, las funcionalidades y las expectativas de rendimiento de la herramienta.
2. Diseño y prototipado:
    a. Arquitectura del sistema: Se diseña la estructura general de la herramienta, identificando los componentes clave y cómo interactuarán entre sí.
    b. Prototipado inicial: Se desarrolla un prototipo inicial basado en técnicas de IA que sirve como versión alfa de la herramienta (Objetivo O2). Este prototipo se utiliza para validar ideas y obtener feedback temprano.
3. Desarrollo Iterativo:

   a. *Sprints* de desarrollo: Se desarrollan y prueban características en *sprints* de 2-3 semanas. Cada sprint tiene un objetivo claro y se centra en un conjunto específico de características o mejoras.
   b. Revisión y feedback: Al final de cada sprint, se realiza una revisión para evaluar el progreso, identificar áreas de mejora y planificar el siguiente sprint.
4. Integración:
   a. Integración modular: Se integra el módulo de detección de desinformación en una herramienta existente o se desarrolla una nueva (Objetivo O3).
   b. Pruebas de integración: Una vez integrados todos los componentes, se realizan pruebas exhaustivas para asegurar que la herramienta funcione de manera cohesiva y sin errores.
5. Pruebas y validación:
   a. Entorno de pruebas: Se configura un entorno de producción simulado para probar todas las características de la herramienta.
   b. Feedback de usuarios: Se invita a un grupo de usuarios a probar la herramienta y proporcionar feedback. Esto permite identificar y corregir problemas antes del lanzamiento oficial.
6. Lanzamiento y monitorización:
   a. Despliegue: Una vez que la herramienta ha sido probada y validada, se lanza al público objetivo.
   b. Monitorización continua: Se establecen herramientas y procesos para monitorizar el rendimiento de la herramienta en tiempo real, identificar problemas y recopilar feedback de los usuarios para futuras iteraciones.

3.3.4. Herramientas y Técnicas

1. Scraping de Telegram:
   a. Herramienta: Telethon.
   b. Descripción: Telethon es una biblioteca de Python que permite interactuar con la API de Telegram. Es esencial para extraer mensajes de canales y grupos de Telegram en tiempo real.

   c. Uso en el proyecto: Se utiliza para rastrear y extraer mensajes de diferentes canales y grupos de Telegram basándose en criterios específicos, como la frecuencia de publicación y el número de vistas.

2. Procesamiento de Lenguaje Natural (NLP):
   a. Herramienta: NLTK, spaCy.
   b. Descripción: Estas son bibliotecas de Python diseñadas para trabajar con datos de lenguaje humano. Ofrecen herramientas para tokenizar, etiquetar y analizar estructuras gramaticales en el texto.
   c. Uso en el proyecto: Se emplean para limpiar y procesar el texto de los mensajes, identificar entidades clave, y preparar los datos para el entrenamiento y la clasificación del modelo.
3. Machine Learning:
   a. Herramienta: Scikit-learn, TensorFlow.
   b. Descripción: Scikit-learn es una biblioteca de Python para aprendizaje automático que ofrece herramientas simples y eficientes para análisis de datos. TensorFlow es una plataforma de código abierto para *machine learning*.
   c. Uso en el proyecto: Se utiliza la regresión logística y otros algoritmos de Scikit-learn para clasificar los mensajes basándose en su contenido y otros atributos. TensorFlow se utiliza para modelos más avanzados y redes neuronales.
4. Desarrollo Web y UI:
   a. Herramienta: Flask, Jinja2.
   b. Descripción: Flask es un micro *framework* de Python para desarrollar aplicaciones web. Jinja2 es un motor de plantillas moderno y diseñado para Python.
   c. Uso en el proyecto: Flask se utiliza para desarrollar la interfaz web de la herramienta, permitiendo a los usuarios visualizar y etiquetar mensajes. Jinja2 se utiliza para renderizar las vistas y presentar los datos al usuario de manera estructurada.
5. Base de Datos:
   a. Herramienta: SQLite.
   b. Descripción: SQLite es una biblioteca en lenguaje C que proporciona una base de datos ligera basada en disco.

   c. Uso en el proyecto: Se utilizaría para almacenar y gestionar los mensajes extraídos de Telegram, así como las etiquetas y clasificaciones asociadas.
6. Control de Versiones:
   a. Herramienta: Git, GitHub.
   b. Descripción: Git es un sistema de control de versiones distribuido. GitHub es una plataforma de hosting para control de versiones y colaboración.
   c. Uso en el proyecto: Se utiliza para gestionar el código fuente del proyecto, rastrear cambios, y colaborar entre miembros del equipo.
7. Editor de texto:
   a. Herramienta: Visual Studio Code (VS Code).
   b. Descripción: Visual Studio Code es un editor de código fuente desarrollado por Microsoft que ofrece funcionalidades como resaltado de sintaxis, finalización inteligente de código, fragmentos de código y depuración integrada.
   c. Uso en el proyecto: Se utilizó como el principal editor de texto para escribir y depurar el código. Gracias a su amplia gama de extensiones y su integración con Git, VS Code facilitó el desarrollo, permitiendo al equipo escribir código de manera eficiente, identificar y corregir errores rápidamente, y gestionar versiones con facilidad.

Por otro lado, como se ha indicado anteriormente, la metodología ágil, con su enfoque iterativo y adaptativo, es ideal para el desarrollo de una herramienta de monitorización de desinformación en Telegram. Permite abordar los desafíos de manera sistemática, adaptarse a los cambios y entregar una solución robusta y eficaz.

1. **Adaptabilidad y flexibilidad:** La elección de una metodología ágil, en particular Scrum, ha demostrado ser esencial para el éxito del proyecto. La naturaleza dinámica de la desinformación y las constantes evoluciones en el campo de la Inteligencia Artificial requieren un enfoque que pueda adaptarse rápidamente a nuevos desafíos y descubrimientos. La capacidad de Scrum para reevaluar y ajustar el enfoque del proyecto en cada sprint ha permitido al equipo mantenerse al día con los cambios y garantizar que la herramienta desarrollada sea relevante y efectiva.

2. **Desarrollo centrado en el usuario:** Una de las fortalezas clave de la metodología ágil es su enfoque en el feedback del usuario. Al desarrollar la herramienta en iteraciones y obtener retroalimentación regularmente, el equipo pudo asegurarse de que la herramienta no solo cumpliera con los objetivos técnicos, sino que también fuera intuitiva y útil para los usuarios finales.

3. **Colaboración mejorada:** Scrum promueve la comunicación y colaboración entre los miembros del equipo. Dado que este proyecto abarcó varias disciplinas, desde el procesamiento de lenguaje natural hasta el desarrollo web, la comunicación regular y las reuniones de revisión aseguraron que todos los miembros del equipo estuvieran alineados y trabajaran hacia un objetivo común.

4. **Entregas incrementales:** La capacidad de entregar características y mejoras de manera incremental permitió al equipo validar ideas rápidamente y garantizar que el desarrollo estuviera en el camino correcto. Esto no solo mejoró la eficiencia del equipo, sino que también redujo el riesgo de invertir tiempo en características que no agregaran valor.

5. **Reflexión continua:** Una parte integral de Scrum es la reflexión y mejora continua. Las retrospectivas al final de cada sprint permitieron al equipo identificar áreas de mejora, tanto en términos de proceso como de producto, y adaptar su enfoque en consecuencia.

6. **Resultado final:** La combinación de una metodología ágil con un equipo comprometido y herramientas y técnicas adecuadas resultó en una herramienta de monitorización de desinformación robusta, eficaz y centrada en el usuario. La herramienta no solo cumple con los objetivos establecidos, sino que también está bien posicionada para adaptarse y evolucionar en respuesta a los desafíos futuros en el campo de la desinformación.

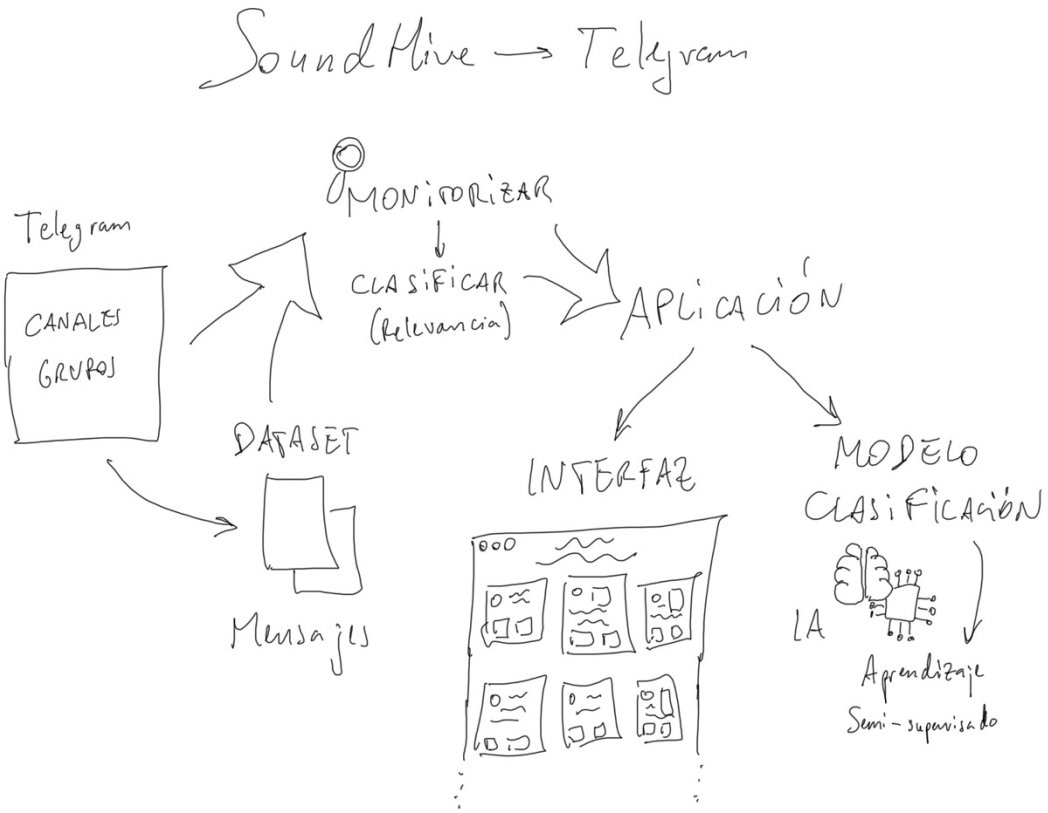

Figura 8.- Primer boceto de la estructura del proyecto. Fuente: elaboración propia.

4. Identificación de requisitos

La identificación de requisitos es un paso fundamental en el proceso de desarrollo de cualquier software o sistema. Es la fase en la que se determinan y documentan las necesidades y expectativas de los usuarios y otras partes interesadas con respecto al sistema propuesto. Esta fase es crucial porque establece el fundamento sobre el cual se construirá todo el proyecto, y cualquier error o ambigüedad en esta etapa puede tener repercusiones significativas en las etapas posteriores.

En el contexto de nuestra herramienta de monitorización de desinformación en Telegram, la identificación de requisitos no solo implica entender las características técnicas que debe tener el software, sino también comprender el complejo panorama de la desinformación, sus manifestaciones en plataformas como Telegram y las necesidades de aquellos que buscan combatirla.

Este capítulo se dedica a detallar el proceso mediante el cual se identificaron los requisitos para el desarrollo de nuestra herramienta. Se abordará la naturaleza del problema de la desinformación en Telegram, el contexto en el que se espera que funcione la herramienta y cómo, con la ayuda de expertos y a través de métodos como entrevistas y observación participante, se llegó a una comprensión clara de lo que se necesita.

A medida que avancemos en este capítulo, se destacará la importancia de un enfoque colaborativo y basado en la evidencia para la identificación de requisitos, y cómo el proyecto IBERIFIER, en el que participa el autor, ha sido instrumental en este proceso.

4.1. CONTEXTUALIZACIÓN DEL PROBLEMA

Como ya se ha comentado, la desinformación, entendida como la difusión deliberada de información falsa o engañosa, ha emergido como uno de los desafíos más apremiantes de nuestra era digital. Las plataformas de mensajería instantánea, como Telegram, se han convertido en vehículos poderosos para la propagación de esta desinformación, dada su naturaleza privada y la capacidad de llegar a grandes audiencias en poco tiempo.

4.1.1. Naturaleza del problema

Telegram, a diferencia de otras plataformas de redes sociales, ofrece canales y grupos que pueden albergar a miles de usuarios, permitiendo la rápida difusión de mensajes. Estos canales y grupos, a menudo administrados por individuos o entidades anónimas, pueden ser utilizados para difundir información errónea con fines políticos, económicos o sociales. La velocidad y el alcance de Telegram, combinados con la falta de supervisión editorial o fact-checking, lo convierten en un medio ideal para la propagación de bulos y noticias falsas.

Además, la naturaleza encriptada de Telegram brinda a los usuarios un sentido de privacidad y seguridad, lo que puede hacer que sean menos críticos o escépticos con respecto a la información que reciben. Esto, a su vez, puede aumentar la susceptibilidad a creer y compartir desinformación.

4.1.2. Impacto y relevancia

El impacto de la desinformación en Telegram no se limita a la plataforma en sí. Los mensajes y las narrativas que ganan tracción en Telegram a menudo se difunden a otras plataformas y medios, amplificando su alcance y potencial daño. Las consecuencias de la desinformación pueden ser variadas, desde la polarización social y el debilitamiento de la confianza en las instituciones, hasta riesgos para la salud pública, como hemos visto con la desinformación relacionada con la COVID-19.

Es esencial reconocer que la desinformación no es simplemente un problema de información errónea; es un fenómeno complejo que se entrelaza con la psicología humana, las dinámicas sociales y las tecnologías de la información. Combatir la desinformación requiere no solo herramientas técnicas, sino también una comprensión profunda de su naturaleza y contexto.

4.2. CONTEXTO DE USO

Para desarrollar una herramienta eficaz, es esencial comprender no solo el problema en sí, sino también el contexto en el que se utilizará la solución. Esto implica identificar a los usuarios objetivo, entender sus necesidades y expectativas, y reconocer los escenarios habituales en los que se empleará la herramienta.

4.2.1. Usuarios objetivo

La herramienta de monitorización de desinformación en Telegram está diseñada para ser utilizada por una variedad de actores interesados en combatir la desinformación:

- **Investigadores**: Académicos y otros profesionales que estudian la propagación de la desinformación y buscan herramientas para recolectar y analizar datos de plataformas como Telegram.
- **Periodistas y *fact-checkers***: Profesionales de los medios de comunicación que buscan verificar la autenticidad de la información que circula en Telegram y otros canales similares.
- **Organizaciones**: Entidades gubernamentales, oenegés y otras organizaciones que trabajan en la lucha contra la desinformación y que requieren herramientas para monitorizar y responder a las campañas de desinformación en tiempo real.

4.2.2. Escenarios habituales

- **Monitorización diaria**: La herramienta puede ser utilizada para supervisar activamente los canales y grupos de Telegram, identificando y alertando sobre posibles bulos o campañas de desinformación en tiempo real.
- **Análisis retrospectivo**: Los usuarios pueden emplear la herramienta para analizar datos históricos, identificando patrones, tendencias y actores clave en la propagación de desinformación en un período determinado.
- **Verificación de hechos**: En situaciones donde surge una noticia o afirmación específica, la herramienta puede ser utilizada para rastrear su origen y propagación en Telegram, ayudando en el proceso de *fact-checking*.

4.2.3. Limitaciones y desafíos

- **Privacidad y ética**: Dada la naturaleza encriptada y privada de Telegram, es esencial garantizar que la herramienta respete la privacidad de los usuarios y opere dentro de los límites éticos y legales.

- **Gran volumen de datos**: Telegram es una plataforma masiva con una gran cantidad de mensajes enviados diariamente. Esto presenta desafíos en términos de procesamiento y análisis de datos.
- **Diversidad de contenido**: Los mensajes en Telegram pueden variar enormemente en términos de idioma, contexto y formato. Esto puede presentar desafíos en términos de análisis y clasificación de contenido.

4.3. METODOLOGÍA DE IDENTIFICACIÓN DE REQUISITOS

La identificación adecuada de requisitos es un proceso que va más allá de simplemente listar características deseadas para un software. Requiere un enfoque sistemático y colaborativo para garantizar que el software desarrollado cumpla con las necesidades y expectativas de sus usuarios y otras partes interesadas.

4.3.1. Participación de expertos

El proyecto IBERIFIER[4], del que es investigador el autor del presente trabajo, ha sido una fuente invaluable de conocimientos y experiencia en el campo de la desinformación. A través de este proyecto, se ha tenido acceso a expertos en la materia, cuyas perspectivas y conocimientos han sido esenciales para identificar los requisitos clave de la herramienta.

Estos expertos, que incluyen académicos, periodistas y profesionales en el campo de la desinformación, han proporcionado *insights* sobre las características y funcionalidades que serían más valiosas en una herramienta de monitorización de desinformación en Telegram.

4.3.2. Entrevistas

Las entrevistas con expertos y otros actores clave han sido una herramienta esencial en el proceso de identificación de requisitos. Estas entrevistas, estructuradas pero flexibles, han permitido profundizar en las necesidades, preocupaciones y expectativas de los usuarios potenciales de la herramienta.

[4] Más información del proyecto IBERIFIER y sus resultados en https://iberifier.eu/

A través de estas conversaciones, se ha podido obtener una comprensión detallada de los desafíos que enfrentan los profesionales al tratar de combatir la desinformación en Telegram y cómo una herramienta podría ayudar a abordar estos desafíos.

Concretamente, se realizaron entrevistas semiestructuradas a responsables de distintas entidades de verificación, así como a diversos periodistas que se dedican a la verificación dentro de estas entidades. Estas entrevistas permitieron profundizar en temas específicos, entender las visiones estratégicas de cada entidad y obtener opiniones expertas sobre las áreas de mejora en el sector.

Las entrevistas semiestructuradas son un método de investigación cualitativa que combina elementos de las entrevistas estructuradas y no estructuradas. Permiten al investigador abordar preguntas específicas y, al mismo tiempo, ofrecen suficiente flexibilidad para que el entrevistado proporcione respuestas detalladas y comparta información adicional que pueda surgir durante la conversación. Asimismo, son útiles para lograr mayor profundidad en la información y establecer un diálogo más directo y personal, lo que puede conducir a una mayor apertura y sinceridad por parte del entrevistado. También son flexibles como para permitir cierta adaptabilidad contextual, dependiendo del perfil del entrevistado, los investigadores pueden adaptar la entrevista para maximizar la relevancia y pertinencia del intercambio.

Con este fin, se llevaron a cabo entrevistas a representantes clave de distintas entidades de verificación, entre ellos Verificat, Maldita, Lusa, EFE Verifica, Polígrafo, Newtral y VerificaRTVE. Cada entrevista se diseñó considerando el perfil y responsabilidades del entrevistado, asegurando así la obtención de datos relevantes y detallados de cada entidad. Estas entrevistas no solo permitieron comprender las operaciones individuales de cada entidad, sino también identificar tendencias y desafíos comunes en el mundo de la verificación.

4.3.3. Observación participante

La observación participante ha ofrecido una oportunidad única para comprender el problema de la desinformación desde una perspectiva interna. Al estar inmerso en las actividades y discusiones del proyecto, se ha podido identificar requisitos y consideraciones que podrían haber sido pasados por alto en un enfoque más tradicional. Esta metodología ha permitido no

solo identificar requisitos técnicos para la herramienta, sino también comprender el contexto más amplio en el que se utilizará, incluidos los desafíos éticos, legales y sociales asociados con la monitorización de plataformas como Telegram.

Esta técnica cualitativa nos permitió sumergirnos en el ambiente de trabajo real de los *fact-checkers*. A través de esta, se ha logrado observar y comprender las prácticas diarias, los retos y necesidades que enfrentan, así como las herramientas que utilizan. La observación participante se realizó siguiendo un esquema previamente establecido para garantizar que todos los aspectos relevantes fueran cubiertos.

La observación participante es una técnica de investigación cualitativa que permite a los investigadores involucrarse y sumergirse directamente en el entorno de la comunidad o grupo que están estudiando. Esta metodología se ha destacado por su capacidad para proporcionar una comprensión profunda de las prácticas cotidianas, interacciones y significados dentro de un contexto particular. La elección de la observación participante para este estudio tiene varias razones fundamentales, principalmente, la comprensión contextual, la interacción directa, la flexibilidad, y el hecho de servir como complemento a otras metodologías, como la cuantitativa.

Durante la última semana de octubre de 2022, el autor se desplazó a Madrid para llevar a cabo la observación participante, con la visita a las redacciones de EFE Verifica, VerificaRTVE, Newtral y Maldita, que proporcionaron una oportunidad inigualable para entender las particularidades y desafíos de cada entidad, enriqueciendo significativamente los hallazgos del estudio. El autor querría aprovechar aquí para agradecer a los responsables y participantes su disponibilidad y atención para la realización de esta investigación.

4.4. REQUISITOS IDENTIFICADOS

Una vez realizada la investigación preliminar y habiendo consultado a los expertos, se identificaron una serie de requisitos clave que la herramienta de monitorización de desinformación en Telegram debería cumplir. Estos requisitos se dividen en funcionales, no funcionales y de integración.

4.4.1. Requisitos funcionales

Los requisitos funcionales describen las funciones y características específicas que la herramienta debe ofrecer:

- **Extracción de mensajes**: La herramienta debe ser capaz de extraer mensajes de canales y grupos específicos de Telegram de manera eficiente y en tiempo real.
- **Análisis de contenido**: Debe analizar el contenido de los mensajes para identificar patrones, temas y otros indicadores de desinformación.
- **Clasificación automática**: Basándose en algoritmos de IA y técnicas de NLP, la herramienta debe ser capaz de detectar mensajes potencialmente dañinos o virales.
- **Interfaz de usuario**: Debe ofrecer una interfaz intuitiva que permita a los usuarios visualizar, filtrar y etiquetar mensajes, así como configurar alertas y generar informes.
- **Entrenamiento del modelo**: La herramienta debe permitir el reentrenamiento y ajuste del modelo de IA basado en las etiquetas proporcionadas por los usuarios.

4.4.2. Requisitos no funcionales

Los requisitos no funcionales se refieren a las cualidades y características generales que la herramienta debe poseer:

- **Rendimiento**: La herramienta debe ser capaz de procesar grandes volúmenes de datos en tiempo real sin retrasos significativos.
- **Seguridad**: Debe garantizar la privacidad y seguridad de los datos, especialmente dado el carácter sensible de la información en Telegram.
- **Usabilidad**: La interfaz de usuario debe ser intuitiva y fácil de usar, incluso para aquellos sin experiencia técnica.
- **Escalabilidad**: La herramienta debe ser capaz de manejar un aumento en la cantidad de datos o usuarios sin comprometer su rendimiento.
- **Adaptabilidad**: Debe ser capaz de adaptarse a cambios en el entorno, como actualizaciones de Telegram o cambios en los patrones de desinformación.

4.4.3. Requisitos de integración

Estos requisitos se refieren a cómo la herramienta se integrará con otros sistemas o plataformas:

- **Compatibilidad con otras herramientas**: La herramienta debe ser capaz de integrarse con otras plataformas o herramientas de monitorización de desinformación, si es necesario.
- **Exportación de datos**: Debe ofrecer opciones para exportar datos en formatos comunes como CSV o JSON para su análisis en otras plataformas.
- APIs y *webhooks*: Debería ofrecer APIs o webhooks para permitir la integración con otros sistemas o para automatizar ciertas tareas.

4.5. CONCLUSIONES

La identificación de requisitos es una etapa crítica en el desarrollo de cualquier software, y en el caso de una herramienta destinada a combatir la desinformación en una plataforma tan amplia y dinámica como Telegram, esta etapa adquiere una importancia aún mayor. A través de un proceso meticuloso y colaborativo, hemos logrado definir un conjunto claro y detallado de requisitos que guiarán el desarrollo subsiguiente de la herramienta.

Algunos puntos clave a destacar incluyen:

1. **La complejidad del problema**: La desinformación en Telegram no es simplemente un problema de información errónea; es un fenómeno multifacético que involucra aspectos técnicos, sociales y psicológicos. Esta complejidad se refleja en los requisitos identificados, que abarcan desde la extracción y análisis de datos hasta la interacción con el usuario y la integración con otros sistemas.
2. **La importancia de la colaboración**: La participación activa de expertos en el campo de la desinformación ha sido esencial para identificar requisitos que son tanto técnicamente factibles como relevantes para el contexto real de uso. El proyecto IBERIFIER ha sido una fuente invaluable de insights y experiencia en este sentido.
3. **Flexibilidad y adaptabilidad**: Dada la naturaleza cambiante de la desinformación y las plataformas en las que se propaga, es esencial que la herramienta desarrollada sea

flexible y adaptable. Los requisitos identificados reflejan esta necesidad, con énfasis en la escalabilidad, adaptabilidad y capacidad de integración.

4. **Centrado en el usuario**: A lo largo del proceso de identificación de requisitos, ha quedado claro que la herramienta debe ser diseñada con el usuario en mente. Esto se refleja en requisitos relacionados con la usabilidad, la interfaz de usuario y la interacción con el sistema.

En resumen, este capítulo ha establecido una base sólida para el diseño y desarrollo de la herramienta de monitorización de desinformación en Telegram. Con un entendimiento claro de las necesidades y expectativas de los usuarios, así como de los desafíos técnicos y contextuales, estamos bien posicionados para avanzar en las siguientes etapas del proyecto.

5. Descripción de la herramienta software desarrollada

Ante el desafío que suponen plataformas de mensajería como Telegram para la desinformación, surge la necesidad de herramientas que puedan identificar, monitorizar y, en última instancia, combatir la propagación de información falsa o engañosa.

La herramienta software desarrollada en el marco de este proyecto aborda precisamente este desafío. Diseñada específicamente para Telegram, esta herramienta utiliza técnicas avanzadas de Inteligencia Artificial para monitorizar en tiempo real los mensajes y grupos de la plataforma, identificando y clasificando aquellos que puedan contener desinformación. El propósito principal es proporcionar a los usuarios, investigadores y profesionales una herramienta robusta y eficiente que les permita tener una visión más clara del panorama de la desinformación en Telegram, facilitando así la toma de medidas adecuadas para contrarrestarla.

A lo largo de este capítulo, se detallará el proceso de desarrollo de esta herramienta, desde su concepción inicial hasta su implementación final, pasando por las decisiones técnicas y metodológicas que han guiado su creación.

5.1. DESARROLLO

5.1.1. Diseño y prototipado

Una vez definidos los requisitos y objetivos del proyecto, el siguiente paso crucial fue diseñar la arquitectura de la herramienta y desarrollar un prototipo inicial. Esta fase es esencial para visualizar cómo funcionaría la herramienta en un entorno real y para identificar posibles desafíos o limitaciones antes de embarcarse en el desarrollo completo.

El diseño de la herramienta se basó en una arquitectura modular, donde cada módulo tiene una función específica y puede operar de manera independiente, pero al mismo tiempo, se integra armoniosamente con otros módulos. Esta arquitectura modular no solo facilita el desarrollo y las pruebas, sino que también permite futuras expansiones o modificaciones con mínimo impacto en el sistema global.

Con la arquitectura definida, se procedió a desarrollar un prototipo inicial de la herramienta. Este prototipo, aunque no tenía todas las funcionalidades de la versión final, sirvió como una versión alfa, que permitió validar conceptos, probar la viabilidad de las técnicas de IA seleccionadas y obtener un feedback temprano. El prototipo se centró principalmente en la extracción de datos de Telegram y en la implementación básica de técnicas de procesamiento de lenguaje natural para analizar el contenido de los mensajes.

Este proceso de prototipado también fue esencial para identificar y abordar posibles desafíos técnicos, como la gestión de grandes volúmenes de datos en tiempo real o la optimización de algoritmos para garantizar tiempos de respuesta rápidos.

El desarrollo del prototipo se llevó a cabo en varias etapas, reflejando la complejidad y la naturaleza multifacética de la herramienta:

- **Construcción del *scraper*:** El primer paso fue desarrollar un *scraper* para extraer datos de Telegram. Utilizando la biblioteca Telethon, se creó un *scraper* eficiente capaz de rastrear y extraer mensajes de diferentes canales y grupos de Telegram en tiempo real. Esta fase fue crucial para garantizar que la herramienta tuviera acceso a un flujo constante y actualizado de datos para su posterior análisis.

- **Desarrollo de la interfaz:** Una vez que se estableció la extracción de datos, se comenzó a construir la interfaz de usuario. Inicialmente, se diseñó una interfaz simple que permitía a los usuarios visualizar los mensajes extraídos. Con el tiempo, se añadieron más funcionalidades, como filtros, opciones de búsqueda y visualizaciones, para mejorar la experiencia del usuario y proporcionar una visión más detallada de los datos.

- **Construcción del modelo de IA:** Con el scraper en funcionamiento y una interfaz en desarrollo, el siguiente paso fue construir el modelo de Inteligencia Artificial que analizaría los mensajes. Se investigaron y probaron varias técnicas de procesamiento de lenguaje natural y machine learning para identificar y clasificar la desinformación. Finalmente, se seleccionó un modelo basado en su precisión y eficiencia, y se integró en la herramienta para analizar los mensajes en tiempo real.

A lo largo de este proceso de prototipado, se mantuvo un enfoque iterativo. Cada componente desarrollado se evaluaba y refinaba continuamente, garantizando que la herramienta evolucionara de manera coherente y efectiva.

5.1.2. Desarrollo iterativo

El desarrollo iterativo es una de las piedras angulares de la metodología ágil, permitiendo al equipo abordar el proyecto en fases manejables y garantizando que cada característica se desarrolle y refine a fondo antes de pasar a la siguiente. Esta fase fue crucial para transformar el prototipo inicial en una herramienta software robusta y completamente funcional.

Basándonos en la metodología Scrum, el desarrollo se dividió en *sprints* muy cortos, de varios días de duración. Cada sprint tenía objetivos claramente definidos, ya fuera desarrollar una nueva funcionalidad, mejorar una existente o solucionar problemas identificados en *sprints* anteriores. Como el trabajo se realizó de manera autónoma e independiente por el autor del trabajo, no se hizo necesaria la coordinación con un equipo, pero sí una reflexión constante y evaluada de las necesidades del proyecto y la herramienta. De esta manera, la revisión y el feedback se producían de manera dialéctica y continua, si bien en alguna ocasión se mostró la herramienta y sus capacidades a diversos compañeros de trabajo para obtener algún tipo de feedback extra y externo al autor del trabajo.

A medida que se desarrollaban y refinaban las características, se implementaba un proceso de integración continua para garantizar que todos los componentes de la herramienta funcionaran juntos de manera cohesiva. Esto implicaba pruebas regulares y la automatización de ciertos procesos para detectar y solucionar problemas rápidamente.

Paralelamente al desarrollo, se mantuvo una documentación detallada de todas las características, decisiones técnicas y cambios realizados. Esto garantiza que cualquier desarrollador o *stakeholder* pueda comprender fácilmente el funcionamiento y la estructura de la herramienta.

A través de este enfoque iterativo y estructurado, se garantizó que la herramienta se desarrollara de manera sistemática, con un alto nivel de calidad y atención al detalle en cada fase.

5.1.3. Integración y pruebas

Una vez que la herramienta sea integrada, probada y optimizada, el siguiente paso crucial será prepararla para su lanzamiento y uso en entornos reales. Esta fase no solo implicará el despliegue de la herramienta, sino también la implementación de mecanismos para monitorizar su rendimiento y recopilar feedback de los usuarios.

Dada la arquitectura modular de la herramienta, cada módulo o componente será desarrollado para operar de manera independiente, pero con interfaces claramente definidas para interactuar con otros módulos. En esta fase, se llevará a cabo la integración de estos módulos, asegurando que interactúen sin problemas y que los datos fluyan correctamente a través de la herramienta.

Una vez integrados todos los componentes, se realizarán pruebas exhaustivas para asegurar que la herramienta funcione de manera cohesiva. Estas pruebas se centrarán en identificar posibles puntos de fallo en las interacciones entre módulos y garantizar que la herramienta produzca resultados consistentes y precisos.

Además de las pruebas de integración, se llevarán a cabo pruebas de funcionalidad para validar cada característica de la herramienta. Esto incluirá pruebas de la interfaz de usuario, del proceso de extracción de datos, del modelo de IA y de cualquier otra funcionalidad incorporada.

Para garantizar que la herramienta no solo cumpla con los criterios técnicos, sino que también sea útil y eficaz para los usuarios finales, se realizará una serie de pruebas con usuarios reales. Estos usuarios proporcionarán feedback valioso sobre la usabilidad, funcionalidad y eficacia de la herramienta en la detección de desinformación.

Basándose en los resultados de las pruebas y el feedback recopilado, se llevarán a cabo optimizaciones y refinamientos adicionales para mejorar el rendimiento, la precisión y la experiencia del usuario.

Esta fase de integración y pruebas será esencial para transformar los diversos componentes desarrollados en una herramienta software robusta, confiable y lista para enfrentar el desafío de la desinformación en Telegram.

5.1.4. Lanzamiento y monitorización

Una vez que la herramienta esté completamente integrada y haya pasado por todas las pruebas necesarias, el siguiente paso planificado será su lanzamiento y despliegue en un entorno de producción. Esta fase no solo contemplará el despliegue efectivo de la herramienta, sino también la implementación de mecanismos para monitorizar su rendimiento y recopilar feedback de los usuarios.

La herramienta será configurada y desplegada en un entorno de producción adecuado, asegurando que todos los recursos necesarios estén disponibles y que pueda manejar la carga de usuarios y datos en tiempo real. Se tomarán medidas para garantizar la seguridad, la escalabilidad y la disponibilidad constante de la herramienta.

Se establecerán herramientas y procesos para monitorizar el rendimiento de la herramienta en tiempo real. Esto incluirá la monitorización de métricas técnicas, como tiempos de respuesta, uso de recursos y errores, así como métricas de uso, como número de usuarios, frecuencia de uso y patrones de interacción.

Se implementarán mecanismos para recopilar feedback de los usuarios, ya sea a través de encuestas, formularios de feedback o entrevistas. Este feedback será esencial para identificar áreas de mejora y adaptar la herramienta según las necesidades y comentarios de los usuarios.

Basándose en la monitorización y el feedback recopilado, se planificarán y llevarán a cabo actualizaciones y mejoras periódicas de la herramienta. Esto garantizará que la herramienta se mantenga actualizada y continúe satisfaciendo las necesidades cambiantes en el campo de la desinformación en Telegram.

Se establecerán canales de comunicación para informar a los usuarios sobre actualizaciones, cambios y nuevas características. Además, se proporcionará soporte para ayudar a los usuarios con cualquier problema o consulta relacionada con la herramienta.

El lanzamiento y la monitorización continuada serán esenciales para asegurar que la herramienta no solo se introduzca con éxito en el mercado, sino que también evolucione y se adapte en respuesta a los desafíos y necesidades cambiantes en el campo de la desinformación en Telegram.

5.2. ARQUITECTURA DE LA HERRAMIENTA

La arquitectura de una herramienta software define cómo están organizados sus componentes, cómo interactúan entre sí y cómo se gestionan los datos. Para la herramienta desarrollada, se optó por una arquitectura modular y escalable que permitiera adaptarse a las necesidades cambiantes y garantizar un rendimiento óptimo.

5.2.1. Componentes principales

La herramienta ha sido diseñada con una estructura modular, lo que facilita su mantenimiento, escalabilidad y adaptabilidad a futuras necesidades. A continuación, se detallan los componentes principales que conforman la arquitectura de la herramienta:

1. Scraper de Telegram: este módulo actúa como el punto de entrada de datos para la herramienta. Utilizando la biblioteca Telethon, está diseñado para rastrear y extraer mensajes de diferentes canales y grupos de Telegram en tiempo real. La eficiencia de este módulo es esencial, ya que garantiza que la herramienta tenga acceso a un flujo constante y actualizado de datos para su posterior análisis. Además, se han implementado medidas para manejar posibles limitaciones o restricciones de la API de Telegram.
2. Motor de Procesamiento de Lenguaje Natural (NLP): una vez que los mensajes son extraídos, es esencial procesar el contenido textual para prepararlo para el análisis. Este módulo se encarga de tareas como la tokenización, eliminación de palabras vacías, detección de entidades y otras funciones de preprocesamiento. La calidad y precisión de este módulo son cruciales, ya que prepara los datos para el análisis posterior por parte del modelo de IA.
3. Modelo de Inteligencia Artificial: este es el núcleo analítico de la herramienta. Utiliza técnicas avanzadas de machine learning y procesamiento de lenguaje natural para analizar los mensajes y detectar posibles desinformaciones. Se ha puesto especial énfasis en garantizar que el modelo sea preciso, pero también eficiente, para poder analizar grandes volúmenes de datos en tiempo real.
4. Base de Datos: para almacenar y gestionar los mensajes extraídos y los resultados del análisis, se ha implementado una base de datos robusta y eficiente. Esta base de datos

no solo almacena la información, sino que también está diseñada para realizar consultas rápidas, lo que es esencial para la funcionalidad de búsqueda y filtrado de la herramienta.

5. Interfaz de usuario: la interfaz actúa como el punto de interacción entre la herramienta y los usuarios. Está diseñada para ser intuitiva y fácil de usar, permitiendo a los usuarios visualizar los resultados, configurar parámetros y realizar búsquedas específicas. Además, se ha puesto especial énfasis en la visualización de datos, proporcionando gráficos y visualizaciones que ayuden a los usuarios a comprender mejor los resultados.

Estos componentes trabajan en conjunto para garantizar que la herramienta sea capaz de detectar y clasificar la desinformación de manera eficiente y precisa en Telegram.

5.2.2. Flujo de datos

El flujo de datos dentro de la herramienta es un proceso secuencial y coordinado que garantiza que cada mensaje extraído de Telegram sea procesado, analizado y almacenado de manera eficiente. A continuación, se describe en detalle cómo se gestiona y procesa la información a través de los diferentes componentes de la herramienta:

1. Extracción de datos con el scraper:
    a. El proceso comienza con el scraper de Telegram, que se conecta a la API de Telegram utilizando las credenciales proporcionadas en el archivo “credentials.txt”.
    b. Se rastrean y extraen mensajes de los canales y grupos proporcionados en el archivo “telegram_channels.csv”, en tiempo real.
    c. Los mensajes extraídos se almacenan temporalmente en una cola o buffer para su procesamiento posterior.
2. Preprocesamiento con el Motor NLP:
    a. Los mensajes en la cola son procesados por el motor de NLP.
    b. Se lleva a cabo la tokenización, donde el contenido textual se descompone en unidades más pequeñas, como palabras o frases.
    c. Las palabras vacías, que son palabras comunes que no aportan significado relevante (como "y", "o", "el"), se eliminan para reducir el ruido en el análisis.

   d. Se detectan entidades, como nombres de personas, lugares o fechas, que pueden ser relevantes para el análisis posterior.
   e. El contenido procesado se pasa al siguiente módulo para su análisis.
3. Análisis con el Modelo de IA:
   a. El modelo de Inteligencia Artificial recibe los mensajes procesados y lleva a cabo el análisis.
   b. Utilizando técnicas avanzadas de machine learning, el modelo evalúa cada mensaje para determinar si contiene desinformación o no.
   c. Se asigna una puntuación o clasificación a cada mensaje, indicando la probabilidad de que contenga desinformación.
4. Almacenamiento en la Base de Datos:
   a. Los mensajes, junto con los resultados del análisis y cualquier metadato relevante, se almacenan en la base de datos.
   b. La estructura de la base de datos está optimizada para consultas rápidas, permitiendo recuperar y visualizar mensajes específicos según las necesidades del usuario.
5. Visualización en la Interfaz de Usuario:
   a. Los resultados del análisis se presentan al usuario a través de la interfaz.
   b. Los usuarios pueden filtrar, buscar y explorar los mensajes, visualizando aquellos que han sido identificados como posibles desinformaciones.
   c. Se proporcionan herramientas y visualizaciones adicionales para ayudar a los usuarios a comprender mejor los resultados y tomar decisiones informadas.

Este flujo de datos garantiza que cada mensaje sea procesado de manera eficiente y que los resultados estén disponibles para los usuarios en tiempo real, permitiendo una detección y respuesta rápidas a la desinformación en Telegram.

5.2.3. Interfaz de usuario

La herramienta desarrollada presenta una interfaz de usuario diseñada para monitorizar mensajes en Telegram y etiquetarlos según su relevancia en relación con la desinformación. La interfaz se centra en la presentación de mensajes extraídos de Telegram. Cada mensaje se muestra en una tarjeta individual que contiene:

- **Puntuación de rendimiento (Overperforming Score)**: Esta puntuación indica cuánto está "sobreexponiéndose" un mensaje en comparación con otros mensajes similares del mismo canal. Se muestra en la parte superior de cada tarjeta.
- **Contenido del mensaje**: Justo debajo de la puntuación, se muestra el contenido del mensaje extraído de Telegram.
- **Botones de etiquetado**: Cada tarjeta tiene dos botones para etiquetar el mensaje como "Relevante" o "No relevante". Estos botones permiten al usuario indicar si un mensaje es potencialmente desinformador o no (o, al menos, relevante para ser tenido en cuenta para el modelo de IA a entrenar).
- **Botón "Ir al mensaje"**: Este botón enlaza directamente al mensaje original en Telegram, permitiendo al usuario ver el contexto completo del mensaje.

En la parte inferior de la interfaz, hay dos botones adicionales:

- **Cargar más** ("Load more"): Permite al usuario cargar más mensajes para su revisión (en ciclos de 24 mensajes, siendo la primera carga de 48).
- **Exportar relevantes**: Este botón permite al usuario exportar todos los mensajes que han sido etiquetados como "relevantes".

La interfaz interactúa con el *backend* de la aplicación para cargar mensajes, etiquetarlos y exportarlos. Cuando un usuario etiqueta un mensaje como "Relevante" o "No relevante", la interfaz envía una solicitud al *backend* para actualizar la etiqueta de ese mensaje en particular. Además, el botón "Cargar más" solicita al *backend* que proporcione más mensajes para revisión.

Para futuras iteraciones de la herramienta ya están contempladas las siguientes mejoras:

- Capacidades de **filtrado y búsqueda** para que los usuarios puedan centrarse en mensajes específicos o temas de interés.
- **Visualización de tendencias**: una sección que muestre las tendencias actuales en los mensajes podría ayudar a los usuarios a identificar rápidamente patrones o temas emergentes, mediante un "topic extractor", por ejemplo.
- **Integración con otras plataformas**: aunque la herramienta se centra en Telegram, se pretende integrarla con otras plataformas de mensajería o redes sociales para ofrecer

una monitorización más amplia. Asimismo, también sería útil conectar la herramienta dentro del flujo de trabajo de la organización que la usa. Por ejemplo, si en la entidad se trabaja internamente mediante una herramienta de comunicación organizacional como Slack, podría integrarse una acción para copiar el mensaje a un canal propio donde empezar a trabajar en su desmentido.

- **Feedback en tiempo real**: Proporcionar retroalimentación en tiempo real sobre el impacto de un mensaje (por ejemplo, cuántas veces ha sido compartido o visto) podría ser valioso para evaluar su alcance.

Estas mejoras no solo enriquecerían la experiencia del usuario, sino que también aumentarían la eficacia de la herramienta en la lucha contra la desinformación en Telegram. En cualquier caso, puede verse cómo la interfaz de usuario ha sido diseñada con un enfoque en la usabilidad y la eficiencia, garantizando que los usuarios puedan interactuar con la herramienta de manera efectiva y obtener *insights* valiosos sobre potenciales mensajes desinformadores y tendencias en Telegram.

5.3. FUNCIONAMIENTO DE LA HERRAMIENTA

5.3.1. Extracción de Datos de Telegram

El archivo `scraper.py` es el encargado de extraer mensajes de diferentes canales y grupos de Telegram. Este es su contenido y funcionamiento:

1. Importación de bibliotecas: Se importan las bibliotecas necesarias para el funcionamiento del script, incluyendo `asyncio`, `csv`, `os`, `telethon`, entre otras.
2. Configuración inicial:
    a. Se establece el directorio de trabajo al directorio del script.
    b. Se solicita al usuario el número de días y el límite máximo de mensajes a extraer por canal.
3. Carga de credenciales: Se cargan las credenciales de la API de Telegram desde el archivo `credentials.txt`.
4. Inicialización del cliente de Telegram: se inicializa el cliente de Telegram utilizando las credenciales cargadas.

5. Carga de canales: se cargan los nombres de los canales o grupos de Telegram desde el archivo `telegram_channels.csv`.
6. Funciones de extracción:
    a. Se definen varias funciones para extraer detalles de los canales, detalles de los mensajes, detalles de los medios y participantes de los canales.
    b. Estas funciones convierten la información extraída en formatos estructurados, como diccionarios, para su posterior procesamiento.
7. Función principal (`main`):
    a. Se itera sobre cada canal cargado.
    b. Se extraen los mensajes del canal utilizando el cliente de Telegram.
    c. Se agrupan los mensajes por fecha y se calcula el promedio diario de vistas, así como una métrica denominada "Score", que calcula una puntuación basada en su rendimiento, como se explica a continuación.
    d. Se procesa cada mensaje, se extraen sus detalles y se agregan a una lista de datos.
    e. Se verifica si el mensaje ya existe en el archivo CSV existente para evitar duplicados.
    f. Se extraen los participantes del canal y se agregan a una lista de participantes.
    g. Se manejan posibles errores, como canales no válidos o inaccesibles.
8. Guardado de datos:
    a. Se carga el archivo CSV existente `telegram_messages.csv` y se actualizan los datos con la nueva información extraída.
    b. Se guarda la información actualizada en el archivo CSV.
    c. Se realiza un proceso similar para los participantes (sólo para los grupos, no para los canales) y se guarda en `telegram_participants.csv`.
9. Ejecución: finalmente, se asegura de que el cliente de Telegram esté iniciado y se ejecuta la función principal.

Uno de los aspectos clave de este archivo es el cálculo del "Score" para cada mensaje. Esta métrica se utiliza para evaluar la relevancia o popularidad de un mensaje en relación con otros mensajes del mismo canal. Se calcula utilizando la cantidad de vistas del mensaje en relación con el promedio diario de vistas de todos los mensajes del canal para ese día.

La fórmula utilizada para calcular el "Score" es:

$$Score = \frac{Vistas\ del\ mensaje}{Vistas\ promedio\ diarias\ del\ canal}$$

Si el "Score" resulta ser un valor entre 0 y 1, se multiplica por -1 para convertirlo en un valor negativo. Esto se hace para diferenciar los mensajes que tienen menos vistas que el promedio diario de aquellos que tienen más vistas que el promedio. El propósito de calcular este "Score" es identificar mensajes que son particularmente populares o relevantes en un canal, ya que un mensaje con un "Score" alto indica que ha recibido muchas más vistas que el promedio de mensajes en ese canal para ese día. Por otro lado, un "Score" negativo indica que el mensaje ha recibido menos vistas que el promedio. Este "Score" puede ser útil para identificar tendencias, mensajes virales o contenido que está ganando tracción en un canal o grupo de Telegram.

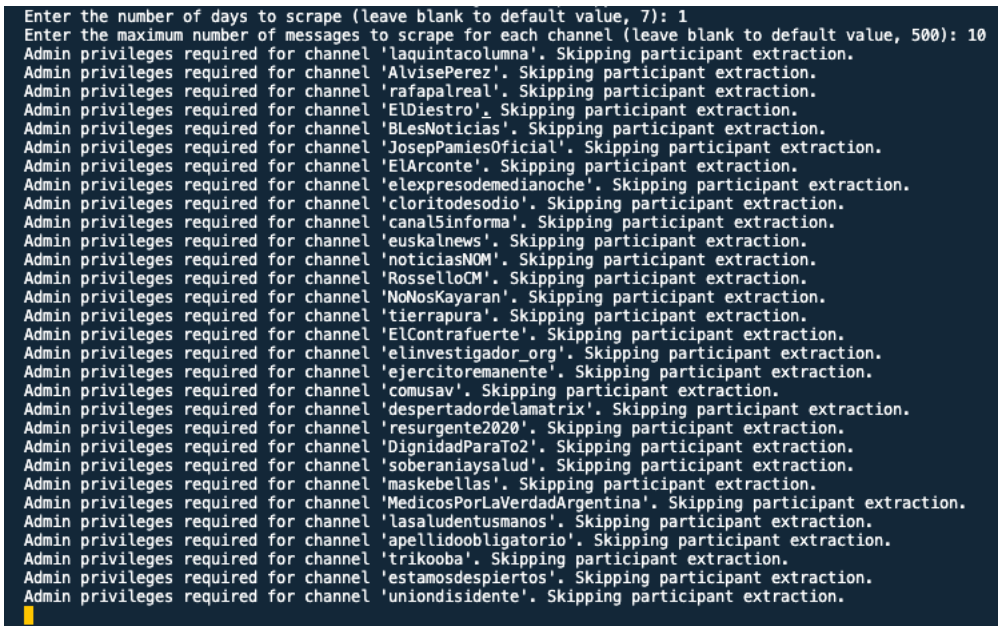

Figura 9.- Captura de pantalla del scraper funcionando, canal a canal. Fuente: elaboración propia.

Así pues, el script `scraper.py` es esencialmente un rastreador que extrae y procesa mensajes y participantes de canales y grupos de Telegram especificados y luego almacena esta información en archivos CSV para su posterior análisis o uso.

5.3.2. Procesamiento y Análisis de Datos

5.3.2.1. Backend

La función de 'app.py' de este archivo es definir la lógica del servidor web utilizando el framework Flask. Se encarga de manejar las solicitudes HTTP, interactuar con el archivo CSV que contiene los mensajes de Telegram y renderizar la página web. Tiene la siguiente estructura:

1. Importaciones: se importan las bibliotecas necesarias, como Flask y pandas.
2. Configuración Inicial: se establece un límite para los mensajes y se carga el archivo CSV `telegram_messages.csv` en un DataFrame.
3. Rutas y Funciones:
    a. `@app.route('/')`: Renderiza la página principal, mostrando los mensajes de Telegram ordenados por su puntuación.
    b. `@app.route('/load_more/<int:offset>', methods=['GET'])`: Permite cargar más mensajes en la página web.
    c. `@app.route('/label', methods=['POST'])`: Etiqueta un mensaje específico como relevante o no relevante.
    d. `@app.route('/export_relevants', methods=['GET'])`: Exporta los mensajes etiquetados como relevantes a un nuevo archivo CSV.
4. Ejecución: el servidor Flask se inicia y escucha en el puerto 5001.

5.3.2.2. Frontend

El archivo index.html (ubicado en la carpeta `templates`): define la estructura y el diseño de la página web que muestra los mensajes de Telegram y permite a los usuarios etiquetarlos, con la siguiente estructura:

1. Diseño: se definen estilos CSS para la presentación de los mensajes, botones y otros elementos de la interfaz.
2. Contenido: se muestra un título y se itera sobre los mensajes para mostrarlos en tarjetas individuales. Cada tarjeta muestra la puntuación del mensaje y tiene botones para etiquetar el mensaje como relevante o no relevante.

3. Interacción: se utiliza JavaScript para manejar eventos de clic en los botones, como cargar más mensajes, etiquetar un mensaje y exportar mensajes relevantes.

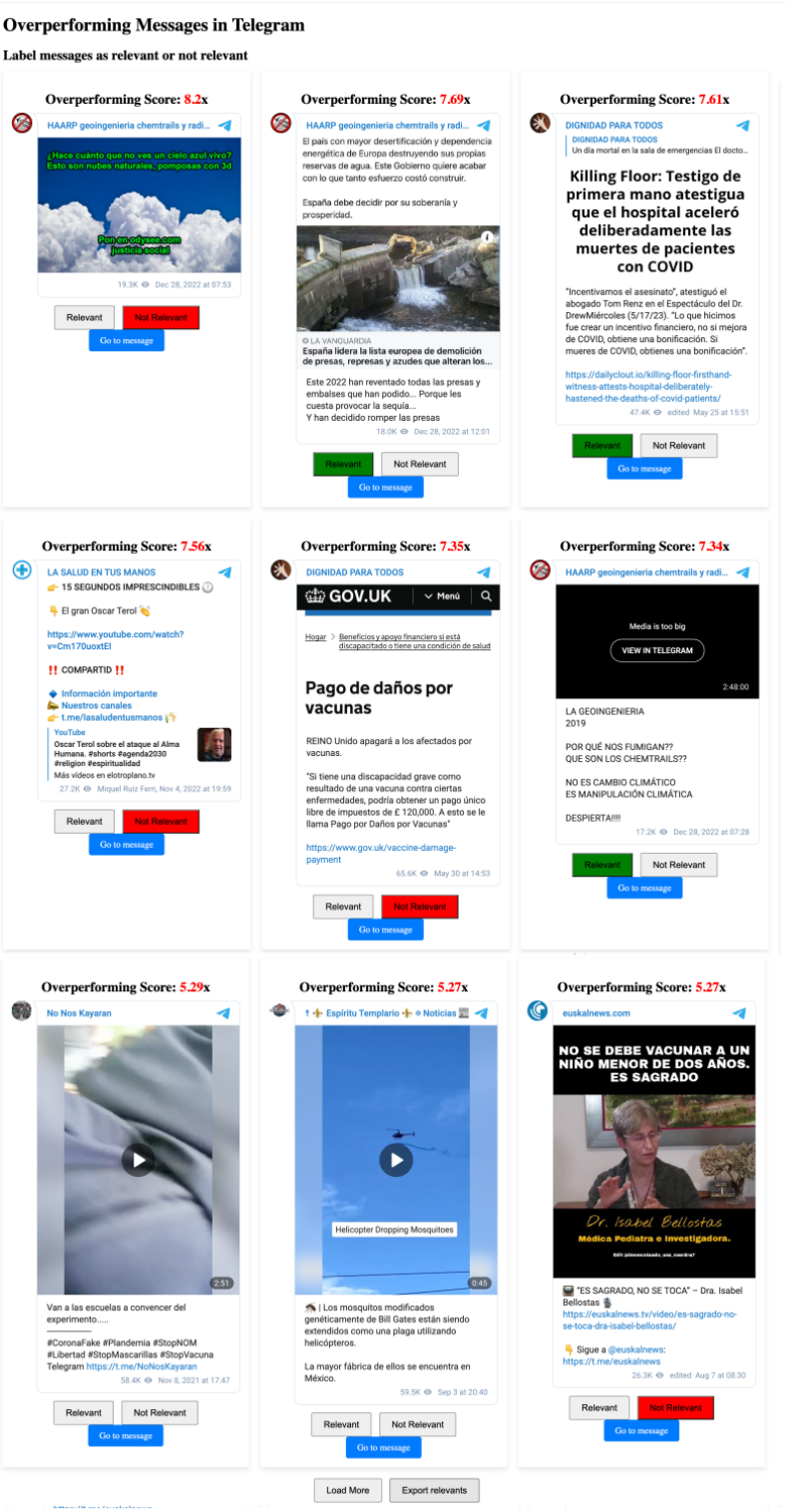

Figura 10.- Captura de pantalla del frontend de la aplicación. Nótense, en cada tarjeta de mensaje, el "overperforming score" y los botones de "relevancia", ya activados. También los botones para la carga de más mensajes y para la exportación de los mensajes relevantes. Fuente: elaboración propia.

Cuando un usuario visita la página web, Flask (a través de `app.py`) renderiza `index.html` y le pasa los mensajes de Telegram como datos. `index.html` muestra estos mensajes y permite a los usuarios interactuar con ellos. Cuando un usuario realiza una acción, como etiquetar un mensaje, se realiza una solicitud HTTP al servidor Flask (definido en `app.py`), que procesa la solicitud y actualiza el archivo CSV según corresponda. El JavaScript en `index.html` también interactúa con `app.py` para cargar más mensajes o exportar mensajes relevantes.

En resumen, `app.py` actúa como el backend que maneja la lógica y la interacción con los datos, mientras que `index.html` es el frontend que proporciona la interfaz de usuario y permite la interacción con los mensajes de Telegram. Ambos archivos trabajan juntos para proporcionar una aplicación web completa para monitorizar y etiquetar mensajes de Telegram.

5.3.2.3. Interacción entre los componentes

Como se ha indicado, `scraper.py` es el responsable de extraer mensajes de canales y grupos de Telegram. Una vez extraídos, estos mensajes se almacenan en un archivo llamado `telegram_messages.csv`. Cada mensaje tiene varios detalles asociados, como el ID del mensaje, el texto del mensaje, la fecha en que se envió, las vistas, entre otros. También se calcula un "Score" para cada mensaje, como mencionamos anteriormente.

Al iniciar `app.py`, se carga el archivo `telegram_messages.csv` en un DataFrame utilizando la biblioteca `pandas`. Este DataFrame contiene todos los mensajes extraídos previamente. `app.py` utiliza este DataFrame para filtrar, procesar y enviar mensajes a la interfaz de usuario (frontend) cuando se solicita.

Cuando un usuario visita la página principal de la aplicación web, `app.py` renderiza `index.html` y le pasa los mensajes de Telegram como datos. `index.html` muestra estos mensajes en tarjetas individuales. Cada tarjeta muestra detalles del mensaje, como el texto del mensaje, la fecha en que se envió, las vistas y el "Score". Los usuarios pueden interactuar con estos mensajes a través de la interfaz web. Por ejemplo, pueden etiquetar un mensaje como relevante o no relevante. Cuando realizan una acción, se envía una solicitud HTTP al servidor Flask (definido en `app.py`), que procesa la solicitud y actualiza el archivo `telegram_messages.csv` según corresponda.

El proceso comienza con `scraper.py` extrayendo mensajes y almacenándolos en `telegram_messages.csv`. Luego, `app.py` carga estos mensajes y los sirve a los usuarios a través de `index.html`. Los usuarios interactúan con los mensajes y realizan acciones, como etiquetarlos. Estas acciones se procesan en `app.py`, que actualiza el archivo `telegram_messages.csv` con la nueva información (por ejemplo, etiquetas). La próxima vez que se ejecute `scraper.py`, se extraerán más mensajes y se actualizará el archivo `telegram_messages.csv`, y el ciclo continúa.

En resumen, `scraper.py` se encarga de la extracción de datos, `app.py` maneja la lógica del servidor y la interacción con los datos, e `index.html` proporciona la interfaz de usuario. Juntos, estos componentes permiten a los usuarios monitorizar y etiquetar mensajes de Telegram de manera efectiva.

5.3.3. Detección y clasificación de mensajes relevantes

A través de estos procesos, el usuario puede ir etiquetando los mensajes, presentados de manera ordenada según su "Score", como "relevantes" o "no relevantes". Esto creará una etiqueta, que será preprocesada mediante técnicas de NLP y luego entrena el modelo de IA para detectar futuros mensajes como relevantes o no. Opcionalmente, el usuario también puede exportar los mensajes del dataset marcados como relevantes. Vamos a verlo de manera más detallada:

1. Cuando un usuario visita la página principal de la aplicación web, `app.py` renderiza `index.html` y le pasa los mensajes de Telegram como datos. Los mensajes se ordenan por su "Score" y se muestran los 48 mensajes principales.
2. Etiquetado de mensajes (`/label`): esta ruta permite a los usuarios etiquetar un mensaje como relevante o no relevante. Cuando un usuario etiqueta un mensaje, se envía una solicitud POST a esta ruta con el ID del mensaje y la etiqueta. Luego, `app.py` actualiza el DataFrame con la nueva etiqueta y guarda los cambios en `telegram_messages.csv`.
3. Exportar mensajes relevantes (`/export_relevants`): esta ruta permite a los usuarios exportar todos los mensajes que han sido etiquetados como relevantes. Los mensajes relevantes se filtran y se guardan en un nuevo archivo CSV llamado `telegram_messages_relevant.csv`.

4. Visualización de mensajes: `index.html` muestra los mensajes de Telegram en tarjetas individuales. Cada tarjeta muestra detalles del mensaje, como el texto del mensaje, la fecha en que se envió, las vistas y el "Score". El usuario ve como ya marcados aquellos mensajes previamente etiquetados. Además, hay un botón para ir al mensaje directamente (para verlo en su contexto, si es necesario).
5. Carga de más mensajes: Hay un botón "Load More" que permite a los usuarios cargar más mensajes. Cuando se hace clic en este botón, se envía una solicitud GET a `app.py` para cargar más mensajes, que luego se agregan a la interfaz de usuario.

El archivo `model.py` es el responsable de procesar y analizar los mensajes de Telegram para entrenar un modelo de clasificación. Aunque `app.py` e `index.html` no interactúan directamente con `model.py`, la etiquetación de mensajes por parte de los usuarios en la interfaz web proporciona datos etiquetados que pueden ser utilizados para entrenar o reentrenar el modelo en `model.py`. El archivo `model.py` es esencialmente un script que procesa y analiza los mensajes de Telegram para entrenar un modelo de clasificación. Este es su funcionamiento:

1. Importaciones y configuración inicial:
    a. Se importan las bibliotecas necesarias, como `pandas`, `numpy`, `nltk`, y `sklearn`.
    b. Se descargan recursos adicionales de `nltk` como 'punkt', 'stopwords' y 'wordnet'.
    c. Se establece el directorio de trabajo al directorio del script.
    d. Se carga el archivo `telegram_messages.csv` en un DataFrame.
2. Procesamiento de texto:
    a. Se define una función `clean_text` para limpiar el texto de los mensajes. Esta función elimina URLs, menciones, caracteres especiales, números y convierte el texto a minúsculas.
    b. Se aplica la función de limpieza a la columna 'Message Text' del DataFrame.
    c. Se tokeniza el texto limpio y se eliminan las palabras vacías (stopwords).
    d. Se realiza la lematización de los tokens.
    e. Vectorización:

        i. Se inicializa un vectorizador TF-IDF para convertir el texto limpio en vectores numéricos.
        ii. Se ajusta y transforma el texto limpio con el vectorizador.
3. Preparación de datos:
    a. Se divide el DataFrame en datos etiquetados y no etiquetados.
    b. Se convierten los datos etiquetados en características y objetivos.
    c. Se dividen los datos etiquetados en conjuntos de entrenamiento y validación.
4. Entrenamiento del modelo:
    a. Se inicializa un modelo de regresión logística.
    b. Se entrena el modelo con el conjunto de entrenamiento.
5. Predicción y actualización:
    a. Se utilizan el modelo entrenado para predecir etiquetas para los datos no etiquetados.
    b. Se actualiza el DataFrame no etiquetado con las etiquetas predichas.
    c. Se combina el DataFrame etiquetado y no etiquetado y se guarda en un nuevo archivo CSV.
6. Evaluación del modelo:
    a. Se generan predicciones para el conjunto de validación.
    b. Se calculan métricas como precisión, exhaustividad, F1 y exactitud.
    c. Se genera y visualiza una matriz de confusión.
    d. Se calculan y visualizan las curvas ROC y de precisión-exhaustividad.

Por tanto, `model.py` toma los mensajes de Telegram almacenados en `telegram_messages.csv`, los procesa y utiliza los mensajes etiquetados para entrenar un modelo de clasificación. Luego, utiliza este modelo para predecir etiquetas para los mensajes no etiquetados y evalúa el rendimiento del modelo utilizando varios métodos y métricas.

Es decir, visto de manera global, `scraper.py` se encarga de la extracción de datos, `app.py` maneja la lógica del servidor y la interacción con los datos, `index.html` proporciona la interfaz de usuario y `model.py` procesa y analiza los mensajes para entrenar un modelo de clasificación. Juntos, estos componentes permiten a los usuarios monitorizar y etiquetar mensajes de Telegram y utilizar esos datos etiquetados para entrenar un modelo de IA.

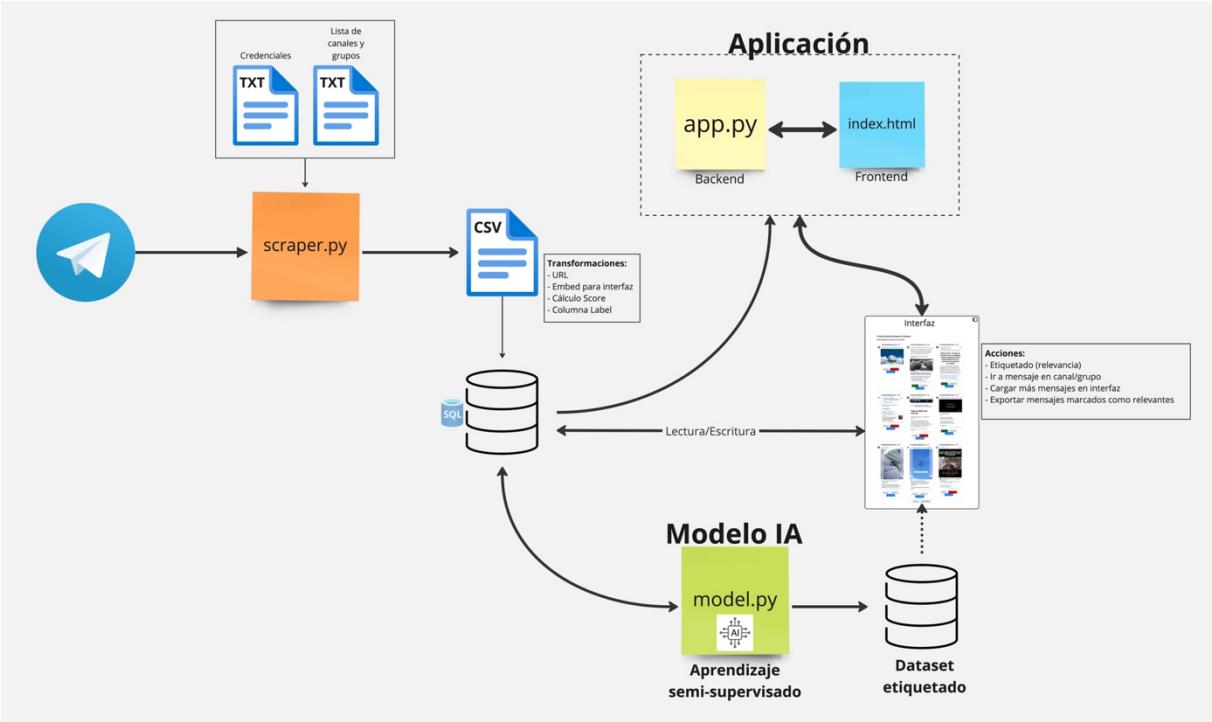

Figura 11.- Diagrama resumen de la herramienta desarrollada, con la interacción entre sus componentes.

Fuente: elaboración propia. Disponible en web.

6. Evaluación

La evaluación es una etapa esencial en el ciclo de vida de cualquier proyecto de desarrollo de software. Permite determinar si la solución propuesta cumple con las expectativas y necesidades de los usuarios, y si aborda adecuadamente el problema para el cual fue diseñada. En el contexto de la lucha contra la desinformación, la evaluación adquiere una importancia aún mayor, ya que las herramientas desarrolladas deben ser no solo funcionales y fáciles de usar, sino también efectivas en la identificación y mitigación de la desinformación.

6.1. EVALUACIÓN INICIAL

La evaluación de una herramienta como la desarrollada en este TFM es esencial para garantizar su eficacia y utilidad. Una evaluación de este tipo abarca múltiples dimensiones que garantizan su eficacia y pertinencia. La **usabilidad** se refiere a la facilidad con la que los usuarios pueden interactuar con la herramienta, considerando aspectos como la intuitividad de la interfaz, la claridad de las instrucciones y la experiencia general del usuario. Una herramienta con alta usabilidad es aquella que los usuarios pueden utilizar sin dificultades y que les proporciona una experiencia satisfactoria. Por otro lado, la **funcionalidad** se centra en la capacidad de la herramienta para realizar las tareas para las que fue diseñada. Esto implica que la herramienta debe ser capaz de identificar y clasificar correctamente la desinformación, adaptándose a diferentes contextos y tipos de mensajes.

El **rendimiento** se relaciona con la eficiencia de la herramienta, es decir, cómo se comporta bajo diferentes cargas de trabajo, su velocidad de respuesta y su capacidad para manejar grandes volúmenes de datos sin degradar su rendimiento. Y la **integración** examina cómo la herramienta se conecta o interactúa con otros sistemas o plataformas. En el contexto de esta herramienta, se refiere a cómo se integra con Telegram o con otras herramientas de análisis de medios sociales.

La **seguridad y privacidad** son aspectos cruciales, especialmente cuando se manejan datos sensibles. Esto implica garantizar que los datos de los usuarios estén protegidos y que la herramienta no sea vulnerable a ataques externos. Además, se debe asegurar que la herramienta cumpla con las regulaciones y leyes de protección de datos.

Hay que tener en cuenta también determinados **aspectos éticos**, que abordan cuestiones relacionadas con la moralidad y la integridad de la herramienta. Por ejemplo, cómo garantizar que la herramienta no perpetúe o introduzca sesgos, o cómo asegurar que la monitorización y análisis de mensajes respete los derechos y libertades de los usuarios. Por último, con la **contribución al campo** nos referimos al valor añadido que la herramienta aporta al ámbito de la desinformación y la inteligencia artificial. Esto puede incluir innovaciones técnicas, metodológicas o teóricas que la herramienta introduce y que pueden ser de utilidad para otros investigadores o profesionales del campo. Vamos a detenernos en cada uno de estos elementos.

6.1.1. Usabilidad

La herramienta, teniendo en cuenta que es un prototipo y que se encuentra en desarrollo, es fácil de usar, partiendo también de la base de que el usuario de esta tiene que conocer el funcionamiento de un lenguaje multiusos como Python. Una vez que la aplicación pueda desplegarse mediante Docker o similar, es posible montar una interfaz en la que distintos usuarios puedan acceder a una misma base de datos para etiquetar datos. Mientras tanto, se han evaluado distintos parámetros de la herramienta en su estado actual, sin tener en cuenta futuras iteraciones.

La velocidad de extracción y descarga de datos desde Telegram depende, obviamente, del equipo y la velocidad de conexión de internet, pero, en el equipo probado, para una búsqueda de 10 mensajes por canal (38 canales en la lista actual), el extractor tarda unos 23 segundos. (casi 17 mensajes por segundo). Para la búsqueda por defecto, 7 días, 500 mensajes por canal, el extractor tarda unos 116 segundos (casi 164 mensajes por segundo). El tiempo de respuesta del *script*, la velocidad de extracción, por tanto, contando que en el tiempo de ejecución del *script* también se generan los archivos xlsx y csv, y que actualiza el archivo con nueva información, manteniendo los mensajes ya existentes en la base de datos, es bastante alta.

La interfaz de usuario se ha creado, modificado y corregido para que sea intuitiva y fácil de usar. Aunque los tiempos de respuesta varían en función del volumen de la base de datos (para lo que se contempla la transformación del archivo CSV en una base de datos SQLite), se considera que entran dentro de lo aceptable para una experiencia de usuario confortable y usable.

También se considera que se ha proporcionado suficiente documentación sobre la herramienta en el repositorio de GitHub, si bien se han incluido medios de contacto con el autor para recabar feedback de los usuarios si éstos notan que falta algún aspecto o es posible alguna mejora.

6.1.2. Funcionalidad

La herramienta cuenta con un nivel de precisión aceptable. La información disponible de los mensajes extraídos de Telegram es extensa y abre muchas puertas a análisis complejos que tengan en cuenta no sólo el texto del mensaje sino también otras muchas variables. La interacción entre los componentes funciona bien y el modelo está listo para su entrenamiento. La cobertura es amplia, y las iteraciones planeadas, en las que se proyecta incluir buscadores y extractores de *topics* para la detección de tendencias, pueden llevar mucho más allá las capacidades de la herramienta. Por otro lado, ya que la herramienta extrae información directamente de los mensajes, se considera que tiene una alta adaptabilidad a los cambios que se puedan producir. También está contemplado que a la hora de trabajar con grupos se extraigan las interacciones entre diferentes miembros, lo que permitiría realizar análisis de redes sobre esta información, trazando así los flujos entre usuarios. Asimismo, sería deseable incluir la capacidad de trazar el "contagio" entre canales, con el reenvío de mensajes, URLs y vídeos entre unos y otros. Algo que puede hacerse ahora, pero de manera muy limitada. En definitiva, la herramienta es funcional, pero en un nivel de mínimos. Hay mucho margen aún para la mejora de esta.

6.1.3. Rendimiento

La herramienta está preparada para la escalabilidad del modelo, pues se ha comprobado que, aunque con tiempos algo más dilatados, la herramienta se comporta bien cuando se enfrenta a grandes volúmenes de datos. La estabilidad de la herramienta también es destacable, pues no se han detectado errores o fallos en su funcionamiento, al irse depurando los mismos antes de su lanzamiento en el repositorio. No obstante, se estará atento en el proceso de evaluación con usuarios a cualquier error que pueda surgir, para repararlo en las sucesivas iteraciones.

Para la evaluación del modelo de IA, que depende de un volumen considerable de etiquetas se ha creado un modelo alternativo, llamado 'model random.py' que completa las etiquetas

de los mensajes de manera aleatoria, aunque teniendo en cuenta la distribución de las etiquetas ya existentes. Las métricas que proporciona este modelo aleatorio sugieren que el modelo no está funcionando bien, especialmente en términos de precisión, recall y puntuación F1. Habría varias causas que explicarían lo que podría estar ocurriendo:

- Accuracy: La precisión de 0,7791 indica que alrededor del 77,91% de las predicciones son correctas. Sin embargo, la precisión puede ser engañosa, especialmente en conjuntos de datos desequilibrados (como el que tenemos).
- Precision, recall y f1-score: Que todas estas métricas sean 0 sugiere que el modelo podría estar prediciendo sólo una clase (probablemente la clase mayoritaria) y no identificando la otra clase en absoluto. Se trata de un problema común en conjuntos de datos desequilibrados en los que una clase supera significativamente a la otra.

6.1.4. Integración

La herramienta ha sido diseñada con un enfoque en la escalabilidad, adaptabilidad y capacidad de integración. Esto sugiere que tiene el potencial de integrarse con otras plataformas o herramientas, aunque todavía no se haya implementado ninguna. Dado que la desinformación no se limita a una sola plataforma, la capacidad de la herramienta para integrarse con otros sistemas o aplicaciones es crucial. Esta compatibilidad permitiría a los usuarios ampliar su alcance y abordar la desinformación en múltiples frentes.

La naturaleza cambiante de la desinformación y las plataformas en las que se propaga requiere que cualquier herramienta desarrollada para combatirla sea flexible y adaptable. La herramienta ha sido diseñada con esta necesidad en mente, lo que indica que es fácil añadir nuevas características o mejorar las existentes. Además, la participación activa de expertos en el campo de la desinformación ha sido esencial para identificar requisitos que son tanto técnicamente factibles como relevantes para el contexto real de uso. Esta colaboración con expertos sugiere que la herramienta puede adaptarse y evolucionar según las necesidades cambiantes del campo de la desinformación.

Es decir, que la herramienta no solo tiene el potencial de integrarse con otras soluciones, sino que también está diseñada para adaptarse y evolucionar con el tiempo, lo que la hace extensible y relevante en la lucha contra la desinformación.

6.1.5. Seguridad y privacidad

La privacidad y seguridad de los datos son esenciales, especialmente cuando se trata de plataformas de comunicación como Telegram, donde los usuarios comparten información personal y sensible. Las comunicaciones están cifradas, si bien los canales analizados son públicos y están disponibles vía URL, por lo que, aunque hay ciertas consideraciones éticas a tener en cuenta, no es competencia de esta herramienta la anonimización de datos, pues sólo extrae y muestra en la interfaz los datos descargados. No obstante, se tienen en cuenta las implicaciones para que cualquier herramienta de este tipo pueda garantizar la privacidad de los datos analizados.

La autenticación y autorización son mecanismos cruciales para garantizar que solo los usuarios autorizados tengan acceso a ciertas partes o funciones de una aplicación. Estos se tendrán en cuenta ante un eventual despliegue para el acceso bajo identificación, pudiéndose implementar algún tipo de sistemas robustos de autenticación (como la autenticación de dos factores) y autorización (como roles y permisos de usuario) para garantizar que solo los usuarios con los permisos adecuados puedan acceder y modificar datos o configuraciones críticas.

6.1.6. Aspectos éticos

La transparencia en las herramientas de Inteligencia Artificial, como la desarrollada en este TFM, es esencial para garantizar que los usuarios y las partes interesadas comprendan cómo se toman las decisiones y cómo se llega a ciertas conclusiones. Una herramienta transparente proporciona explicaciones claras de sus procesos, algoritmos y métodos de análisis. En el caso de la herramienta desarrollada, es crucial que los usuarios comprendan cómo se extraen, procesan y analizan los datos de Telegram para identificar la desinformación. Esta comprensión permite a los usuarios confiar en los resultados y tomar decisiones informadas basadas en los análisis proporcionados por la herramienta.

El sesgo en las herramientas de IA puede surgir debido a datos de entrenamiento sesgados o algoritmos que no se han ajustado adecuadamente para tratar con ciertos tipos de información. Por ello, se tiene en cuenta para futuras iteraciones que la herramienta debe ser

revisada y ajustada regularmente para garantizar que los algoritmos y métodos de análisis sean justos y objetivos.

6.1.7. Contribución al campo

La innovación de la herramienta desarrollada en este TFM radica en su enfoque específico hacia la plataforma Telegram, una de las plataformas de mensajería más populares pero menos estudiadas en términos de desinformación. A diferencia de otras herramientas que se centran en redes sociales más convencionales, esta herramienta aborda los desafíos únicos que presenta Telegram, como la naturaleza privada de muchos de sus canales y la falta de herramientas de moderación tradicionales. Además, la herramienta integra técnicas avanzadas de procesamiento del lenguaje natural y aprendizaje automático para identificar y clasificar la desinformación de manera efectiva. También se contempla su extensibilidad para integrarse con otras plataformas en futuras iteraciones, lo que la posiciona como una solución versátil y adaptable.

En cuanto a la relevancia, la desinformación ha emergido como uno de los desafíos más críticos de nuestra era digital, con consecuencias significativas en la política, la salud pública y la cohesión social. Telegram, siendo una plataforma que ha ganado popularidad rápidamente, se ha convertido en un terreno fértil para la propagación de desinformación debido a su estructura y características. La herramienta desarrollada aborda directamente este problema, ofreciendo a los verificadores de hechos, periodistas y organizaciones una solución para monitorear, identificar y combatir la desinformación en Telegram. Su enfoque en esta plataforma específica y su capacidad para adaptarse a las características únicas de Telegram la hacen especialmente relevante en el campo actual de la lucha contra la desinformación.

6.1.8. Conclusiones

Los aspectos evaluados han tratado de proporcionar una visión lo más exhaustiva posible de la herramienta para ayudar a identificar áreas de mejora y fortalezas. Se entienden las limitaciones al haber realizado la evaluación por cuenta propia del autor y por ello, además de llevar a cabo evaluaciones periódicas para garantizar que la herramienta sigue siendo relevante y efectiva a medida que evoluciona el panorama de la desinformación, se realiza a continuación una propuesta más completa para la evaluación de la herramienta

A continuación, presentamos dicha propuesta de evaluación prevista para esta herramienta de monitorización de desinformación en Telegram. En ella, se abordan aspectos relacionados con la usabilidad de la herramienta, así como su aplicabilidad. A través de pruebas prácticas y feedback de expertos, se buscará obtener una evaluación para detectar fortalezas y áreas de mejora de la herramienta.

6.2. PROPUESTA DE EVALUACIÓN

La herramienta desarrollada en este Trabajo Fin de Máster tiene como objetivo principal combatir la desinformación en Telegram. Para garantizar su eficacia y relevancia, es esencial llevar a cabo una evaluación exhaustiva que aborde diversos aspectos:

1. **Evaluación de usabilidad**: se llevarán a cabo pruebas específicas para determinar la facilidad de uso de la herramienta. Esto incluirá pruebas de interfaz, tiempo de respuesta, y la intuitividad general de la herramienta. Se buscará feedback específico sobre cualquier dificultad o desafío que los usuarios puedan enfrentar al usar la herramienta.
2. **Evaluación de aplicabilidad**: esta evaluación se centrará en determinar la capacidad de la herramienta para abordar y resolver el problema específico para el cual fue diseñada, es decir, combatir la desinformación en Telegram. Se evaluará si la herramienta es efectiva en la identificación y mitigación de la desinformación en esta plataforma. La metodología propuesta para esta evaluación incluirá pruebas en situaciones reales, análisis de la precisión y exhaustividad de los resultados, y comparaciones con otras herramientas o métodos existentes.
3. **Evaluación con expertos**: la opinión y el feedback de expertos en el campo de la desinformación son esenciales. Los expertos pueden proporcionar insights valiosos basados en su experiencia y conocimiento del campo. Se llevará a cabo una selección rigurosa de expertos propuestos que tengan una amplia experiencia en el campo de la desinformación. Basándose en el feedback y los insights de los expertos, se anticipa que surgirán recomendaciones específicas para mejorar la herramienta. Estas recomendaciones pueden abordar aspectos técnicos, funcionales o incluso estratégicos.

4. **Recomendaciones futuras**: A partir de las evaluaciones y el feedback obtenidos, se propondrán mejoras y adaptaciones. Esto puede incluir la implementación de mejoras en la interfaz, la optimización de algoritmos, o la expansión de la herramienta para abordar nuevas plataformas o tipos de desinformación. Además, se recomendará realizar evaluaciones periódicas en el futuro para garantizar que la herramienta siga siendo efectiva a medida que evolucionan las tácticas y estrategias de desinformación.
5. **Feedback y mejoras continuas**: Basándose en todas las evaluaciones, se establecerá un proceso de mejora continua. Esto garantizará que la herramienta se adapte y evolucione según las necesidades cambiantes del campo de la desinformación y las expectativas de los usuarios.

Esta propuesta de evaluación combina los aspectos clave de distintas evaluaciones, proporcionando una visión holística de cómo se puede evaluar la herramienta en el futuro.

6.3. CONCLUSIONES

La evaluación es un paso fundamental en el desarrollo y refinamiento de cualquier herramienta o software. A través de la evaluación, se pueden identificar áreas de mejora, validar la efectividad de la solución y garantizar que cumpla con las necesidades y expectativas de los usuarios y expertos en el campo.

La propuesta de evaluación presentada en este capítulo tiene como objetivo garantizar que la herramienta de monitorización de desinformación en Telegram no solo sea usable, sino también efectiva en su propósito principal: combatir la desinformación. Al combinar pruebas de usabilidad, evaluación de aplicabilidad y feedback de expertos, se busca obtener una visión holística de la herramienta y su impacto potencial.

Aunque la evaluación aún no se ha llevado a cabo, se espera que los resultados proporcionen insights valiosos sobre la herramienta. Estos insights guiarán futuras iteraciones y mejoras, asegurando que la herramienta siga siendo relevante y efectiva en el cambiante paisaje de la desinformación.

Una vez que se complete la evaluación propuesta, será esencial actuar según los resultados y feedback obtenidos. Esto puede incluir la implementación de mejoras en la interfaz, la

optimización de algoritmos o la expansión de la herramienta para abordar nuevas plataformas o tipos de desinformación. Además, se recomendará realizar evaluaciones periódicas en el futuro para garantizar que la herramienta siga siendo efectiva a medida que evolucionan las tácticas y estrategias de desinformación.

7. Conclusiones y trabajo futuro

7.1. CONCLUSIONES

Este TFM se ha centrado en abordar el creciente problema de la desinformación y la posverdad, especialmente en plataformas como Telegram. En un mundo donde la información fluye rápidamente y en grandes cantidades, es esencial contar con herramientas eficientes que permitan discernir la veracidad de los contenidos. La relevancia de este trabajo radica en su aportación al campo del periodismo de verificación y *fact-checking*, utilizando técnicas avanzadas de inteligencia artificial y aprendizaje automático.

El problema de la desinformación fue planteado desde una perspectiva multidimensional, considerando no solo los contenidos falsos o engañosos, sino también los contextos y plataformas donde estos proliferan. Para abordar este desafío, se propuso una metodología robusta y detallada, que abarcó desde la identificación de requisitos hasta el desarrollo y evaluación de una herramienta software específica.

La solución propuesta, una herramienta de monitorización y verificación, se ha validado a través de diversas fases de desarrollo y evaluación. Su diseño se basó en las necesidades reales de los usuarios, y su eficacia se comprobó mediante pruebas de usabilidad, aplicabilidad y consultas con expertos en el campo.

En cuanto a las contribuciones del trabajo, se pueden destacar varias. Primero, se ha proporcionado una visión detallada del estado del arte en desinformación y técnicas de verificación, con un enfoque especial en Telegram. Segundo, se ha desarrollado una herramienta práctica y eficiente que combina técnicas de procesamiento del lenguaje natural y aprendizaje automático para abordar el problema de la desinformación. Y, finalmente, se ha ofrecido una propuesta de evaluación exhaustiva de la herramienta, que puede confirmar su potencial y relevancia en el ámbito del *fact-checking*.

Relacionando estas contribuciones con los objetivos planteados al inicio del trabajo, se puede afirmar que se han alcanzado satisfactoriamente. La herramienta desarrollada no solo aborda el problema inicialmente identificado, sino que también proporciona una solución innovadora

y práctica que puede ser de gran utilidad para periodistas, investigadores y el público en general.

En resumen, este Trabajo de Fin de Máster ha logrado aportar una solución valiosa al desafío de la desinformación, demostrando la importancia y eficacia de combinar técnicas avanzadas de inteligencia artificial con un enfoque centrado en el usuario.

7.2. LÍNEAS DE TRABAJO FUTURO

El Trabajo de Fin de Máster presentado sienta las bases para diversas líneas de investigación y desarrollo que pueden enriquecer aún más la herramienta y el campo de estudio de la desinformación y la verificación de hechos. A continuación, se detallan algunas de estas perspectivas de futuro:

1. **Explicabilidad de la IA**: una de las principales preocupaciones en el ámbito de la inteligencia artificial es la explicabilidad y transparencia de los modelos. Aunque la herramienta desarrollada ha demostrado ser eficaz, es esencial que los usuarios, especialmente los periodistas y verificadores de hechos, comprendan cómo y por qué la herramienta toma ciertas decisiones. Integrar técnicas de IA explicables puede mejorar la confianza en la herramienta y facilitar su adopción en entornos profesionales.
2. **Mejoras en la interfaz de usuario**: como se mencionó en el punto 5.2.3, hay varias mejoras propuestas para la interfaz de usuario. Estas mejoras no solo mejorarán la experiencia del usuario, sino que también pueden facilitar la interpretación y el análisis de los resultados. La incorporación de visualizaciones interactivas y *dashboards* personalizables puede ser de gran valor.
3. **Ampliación a otras plataformas**: aunque Telegram ha sido el foco principal de este trabajo, la desinformación es un problema omnipresente en muchas otras plataformas. Adaptar y expandir la herramienta para monitorizar y verificar contenidos en otras redes sociales y plataformas de mensajería puede ampliar significativamente su impacto.
4. Integración con otras herramientas de *fact-checking*: la colaboración y la integración con otras herramientas y plataformas de verificación de hechos pueden potenciar la

eficacia de la solución propuesta. Esto permitiría una verificación más rápida y precisa, aprovechando la experiencia y los recursos de múltiples fuentes.

5. **Investigación en desinformación generada por ordenador**: con el auge de las técnicas de generación de contenido, como GPT y otros modelos de lenguaje, es esencial investigar y desarrollar métodos para detectar y combatir la desinformación generada automáticamente.

El trabajo desarrollado en este TFM abre un abanico de posibilidades en el campo de la desinformación y la verificación de hechos. La herramienta propuesta puede ser empleada en diversos campos, desde el periodismo y la comunicación hasta la educación y la formación cívica. Además, su adaptabilidad y enfoque centrado en el usuario la convierten en una solución prometedora para enfrentar los desafíos actuales y futuros de la desinformación en la era digital. Esperamos que sea igual de bien acogida entre los posibles usuarios de la herramienta desarrollada.

Referencias bibliográficas

Aleksandra Urman, Urman, A., Ho, J. C. K., Justin Chun-ting Ho, & Katz, S. (2021). Analyzing protest mobilization on Telegram: The case of 2019 Anti-Extradition Bill movement in Hong Kong. *PLOS ONE*, *16*(10). https://doi.org/10.1371/journal.pone.0256675

Andrino, J. P. C., Borja. (2021, febrero 6). La desinformación explota en Telegram: Cientos de miles de cuentas siguen canales conspirativos en español. *El País*. https://elpais.com/tecnologia/2021-02-06/la-desinformacion-explota-en-telegram-cientos-de-miles-de-cuentas-siguen-canales-conspirativos-en-espanol.html

Arnold, P. (2020). *The challenges of online fact checking*. Technical report, Full Fact.

Azevedo, L., D'aquin, M., Davis, B., & Zarrouk, M. (2021). Lux (linguistic aspects under examination): Discourse analysis for automatic fake news classification. *The Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL-IJCNLP 2021)*, 41-56.

Baudrillard, J. (1998). *La posmodernidad: Vol. 4ª*. Kairós.

Bauman, Z. (1996). Teoría sociológica de la posmodernidad. *Espiral*, *Journal Article*. http://unir.summon.serialssolutions.com/link/0/eLvHCXMwQxzbCCsPEMUB6tQ7IjIBWHJa6BqD71iIcrIM8zZzczcKZGKAXdVZlJqSmFOZrFeamJqbWqGXWwET0S8uStYHbSTMzNd3LCoJSEkLSk2BnDnoDDr_1hhYz5lCTg01MAbdcxDkEgkrr0E9GPAyfDMzU11gcwC0rwlqLFJN4ybIwA9tIio4QuJUiIEptViYQRKyj1YBmheLFTSgB0ZrijCoh6TmFx1em6hQDN59cngzqDRTSElVyEksyC_OBd92lpmSmMI3hy2-yvKW-9IpOcb_tuRyXAQAYeJk3A

Bauman, Z. (2001). *La posmodernidad y sus descontentos* (Vol. 11). Akal.

Baumgartner, J., Zannettou, S., Squire, M., & Blackburn, J. (2020). The Pushshift Telegram Dataset. *Proceedings of the International AAAI Conference on Web and Social Media*, *14*, 840-847. https://doi.org/10.1609/icwsm.v14i1.7348

Bondielli, A., & Marcelloni, F. (2019). A survey on fake news and rumour detection techniques. *Information Sciences*, *497*. https://doi.org/10.1016/j.ins.2019.05.035

Bonet-Jover, A., Piad-Morffis, A., Saquete, E., Martínez-Barco, P., & García-Cumbreras, M. Á. (2021). Exploiting discourse structure of traditional digital media to enhance automatic fake news detection. *Expert systems with applications*, *169*, 114340.

Bovet, A., & Grindrod, P. (2022). Organization and evolution of the UK far-right network on Telegram. *Applied Network Science*, *7*(1), 1-27.

Cantón-Correa, Fco. J. (2022). *Tecnologías de IA contra la desinformación*. https://doi.org/10.5281/ZENODO.7596358

Cantón-Correa, F.-J., & VerificaRTVE. (2021, julio 4). De Telegram a la web: Así se organiza una campaña para desinformarte sobre grafeno y vacunas. *RTVE.es*. https://www.rtve.es/noticias/20210704/analizamos-bulo-grafeno-vacuna-informe/2119721.shtml

Castelo, S., Almeida, T., Elghafari, A., Santos, A., Pham, K., Nakamura, E., & Freire, J. (2019). A topic-agnostic approach for identifying fake news pages. *Companion proceedings of the 2019 World Wide Web conference*, 975-980.

Chen, B. X., & Roose, K. (2021, febrero 3). Are Private Messaging Apps the Next Misinformation Hot Spot? *The New York Times*. https://www.nytimes.com/2021/02/03/technology/personaltech/telegram-signal-misinformation.html

Cornella, A. (2000). Cómo sobrevivir a la infoxicación. *Infonomia. com*, *8*.

Dargahi Nobari, A., Reshadatmand, N., & Neshati, M. (2017). Analysis of Telegram, An Instant Messaging Service. *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 2035-2038. https://doi.org/10.1145/3132847.3133132

Dargahi Nobari, A., Sarraf, M. H. K. M., Neshati, M., & Erfanian Daneshvar, F. (2021). Characteristics of viral messages on Telegram; The world's largest hybrid public and private messenger. *Expert Systems with Applications*, *168*, 114303. https://doi.org/10.1016/j.eswa.2020.114303

Della Vedova, M. L., Tacchini, E., Moret, S., Ballarin, G., DiPierro, M., & De Alfaro, L. (2018).

Automatic Online Fake News Detection Combining Content and Social Signals. *2018 22nd Conference of Open Innovations Association (FRUCT)*, 272-279. https://doi.org/10.23919/FRUCT.2018.8468301

Echevarría, B. (2016). Más 'fact-checking'contra la posverdad. *Cuadernos de periodistas*, *33*, 9-16.

Garrido, M. D., Farpón, C. R., & Cano-Orón, L. (2021). La desinformación en las redes de mensajería instantánea. Estudio de las fake news en los canales relacionados con la ultraderecha española en Telegram. *Miguel Hernández Communication Journal*, *12*, 467489-467489.

Giménez, J. (2022, enero 9). Desinformación en Telegram: Cómo se propaga y por qué los desinformantes eligen esta red de mensajería - Chequeado. *Chequeado*. https://chequeado.com/el-explicador/desinformacion-en-telegram-como-se-propaga-y-por-que-los-desinformantes-eligen-esta-red-de-mensajeria/

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. The MIT Press.

Graves, L. (2018). *Understanding the promise and limits of automated fact-checking*.

Graves, L., Nyhan, B., & Reifler, J. (2016). Understanding innovations in journalistic practice: A field experiment examining motivations for fact-checking. *Journal of communication*, *66*(1), 102-138.

Guo, B., Ding, Y., Yao, L., Liang, Y., & Yu, Z. (2020). The future of false information detection on social media: New perspectives and trends. *ACM Computing Surveys*, *53*(4). https://doi.org/10.1145/3393880

Guo, Z., Schlichtkrull, M., & Vlachos, A. (2022). A survey on automated fact-checking. *Transactions of the Association for Computational Linguistics*, *10*, 178-206.

Kakutani, M. (2018). *The death of truth* (First edition). Tim Duggan Books.

La Barbera, D., Roitero, K., & Mizzaro, S. (2022). A hybrid human-in-the-loop framework for fact checking. *Proceedings of the Sixth Workshop on Natural Language for Artificial*

Intelligence (NL4AI 2022) co-located with 21th International Conference of the Italian Association for Artificial Intelligence (AI IA 2022)*.

Lewandowsky, S., Ecker, U. K. H., & Cook, J. (2017). Beyond Misinformation: Understanding and Coping with the "Post-Truth" Era. *Journal of Applied Research in Memory and Cognition*, *6*(4), 353-369. Scopus. https://doi.org/10.1016/j.jarmac.2017.07.008

López, J. C. A. D., Molina-Solana, M., & Gómez-Romero, J. (2019). Towards easy-to-implement misinformation automatic detection for online social media. *TTO*.

Lyotard, J. F. (2008). *La posmodernidad (explicada a los niños): Vol. 9ª reimpr*. Gedisa.

Madrigal, M. (2021a, agosto 14). Médicos por la verdad: Los vínculos de una estructura para el engaño. *Newtral*. https://www.newtral.es/medicos-por-la-verdad-desinformacion-web-marcelino-madrigal/20210814/

Madrigal, M. (2021b, agosto 22). ANÁLISIS | Telegram, segunda parte: Qué contenidos se difunden por los canales de desinformación. *Newtral*. https://www.newtral.es/telegram-desinformacion-bulos-analisis-ciencia-covid/20210822/

Maldita. (2022, diciembre 19). El clan del negacionismo en Telegram. *Maldita.es — Periodismo para que no te la cuelen*. https://maldita.es/malditodato/20221219/clan-negacionismo-telegram/

Marcus, G. (2022, diciembre 19). *AI Platforms like ChatGPT Are Easy to Use but Also Potentially Dangerous*. Scientific American. https://www.scientificamerican.com/article/ai-platforms-like-chatgpt-are-easy-to-use-but-also-potentially-dangerous/

Mitchell, E., Lee, Y., Khazatsky, A., Manning, C. D., & Finn, C. (2023). *DetectGPT: Zero-Shot Machine-Generated Text Detection using Probability Curvature*. https://doi.org/10.48550/ARXIV.2301.11305

Molina-Solana, M., Amador Diaz Lopez, J., & Gomez, J. (2018). Deep learning for fake news classification. *I workshop in deep learning, 2018 conference Spanish association of artificial intelligence*, 1197-1201.

Nakamura, K., Levy, S., & Wang, W. Y. (2020). Fakeddit: A New Multimodal Benchmark Dataset for Fine-grained Fake News Detection. *Proceedings of the Twelfth Language Resources and Evaluation Conference*, 6149-6157. https://aclanthology.org/2020.lrec-1.755

Nakov, P., Corney, D., Hasanain, M., Alam, F., Elsayed, T., Barrón-Cedeño, A., Papotti, P., Shaar, S., & Martino, G. D. S. (2021). Automated fact-checking for assisting human fact-checkers. *arXiv preprint arXiv:2103.07769*.

Newtral. (2022, diciembre 21). ClaimCheck, la herramienta desarrollada por Newtral y ABC Australia para combatir las mentiras repetidas. *Newtral*. https://www.newtral.es/claimcheck-herramienta-mentiras-repetidas/20221221/

Pasi, G., De Grandis, M., & Viviani, M. (2020). Decision Making over Multiple Criteria to Assess News Credibility in Microblogging Sites. *2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 1-8. https://doi.org/10.1109/FUZZ48607.2020.9177751

Peeters, S., & Hagen, S. (2022). The 4CAT Capture and Analysis Toolkit. *Computational Communication Research*, *4*(2), Article 2.

Ruffo, G., Semeraro, A., Giachanou, A., & Rosso, P. (2023). Studying fake news spreading, polarisation dynamics, and manipulation by bots: A tale of networks and language. *Computer science review*, *47*, 100531.

Sádaba-Chalezquer, C., & Salaverría-Aliaga, R. (2022). *I Estudio sobre la desinformación en España*.

Sánchez González, M., Gonzales, H. M. S., & Gonzalo, S. M. (2022). Inteligencia artificial en verificadores hispanos de la red IFCN: Proyectos innovadores y percepción de expertos y profesionales. *Estudios sobre el Mensaje Periodístico*, *28*(4), Article 4. https://doi.org/10.5209/esmp.82735

Schulze, H., Hohner, J., Greipl, S., Girgnhuber, M., Desta, I., & Rieger, D. (2022). Far-right conspiracy groups on fringe platforms: A longitudinal analysis of radicalization dynamics on Telegram. *Convergence: The International Journal of Research into New Media Technologies*, *28*(4), 1103-1126.

Shabani, S., Charlesworth, Z., Sokhn, M., & Schuldt, H. (2021). SAMS: Human-in-the-loop approach to combat the sharing of digital misinformation. *CEUR Workshop Proc.*

Shu, K., Sliva, A., Wang, S., Tang, J., & Liu, H. (2017). Fake news detection on social media: A data mining perspective. *ACM SIGKDD Explorations Newsletter*, *19*(1), 22-36.

Shu, K., Wang, S., & Liu, H. (2019). Beyond News Contents: The Role of Social Context for Fake News Detection. *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, 312-320. https://doi.org/10.1145/3289600.3290994

Solaiman, I., Brundage, M., Clark, J., Askell, A., Herbert-Voss, A., Wu, J., Radford, A., Krueger, G., Kim, J. W., Kreps, S., McCain, M., Newhouse, A., Blazakis, J., McGuffie, K., & Wang, J. (2019). *Release Strategies and the Social Impacts of Language Models*. https://doi.org/10.48550/ARXIV.1908.09203

Su, C. C., Chan, M., & Paik, S. (2022). Telegram and the anti-ELAB movement in Hong Kong: Reshaping networked social movements through symbolic participation and spontaneous interaction. *Chinese Journal of Communication*, *15*(3), 431-448. https://doi.org/10.1080/17544750.2022.2092167

Toffler, A. (1980). *La tercera ola* (A. Martín, Trad.). Plaza & Janés. http://dialnet.unirioja.es/servlet/libro?codigo=224431

Ufarte-Ruiz, M.-J., Peralta García, L., & Murcia-Verdú, F.-J. (2018). *Fact checking: Un nuevo desafío del periodismo*.

Urman, A., & Katz, S. (2022). What they do in the shadows: Examining the far-right networks on Telegram. *Information, Communication & Society*, *25*(7), 904-923. https://doi.org/10.1080/1369118X.2020.1803946

Viviani, M., & Pasi, G. (2017). Credibility in social media: Opinions, news, and health information—A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *7*(5). https://doi.org/10.1002/widm.1209

Vosoughi, S., Roy, D., & Aral, S. (2018). The spread of true and false news online. *Science*, *359*(6380), 1146-1151. https://doi.org/10.1126/science.aap9559

Wang, W. Y. (2017). “Liar, Liar Pants on Fire”: A New Benchmark Dataset for Fake News Detection. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 422-426. https://doi.org/10.18653/v1/P17-2067

Wardle, C., & Derakhshan, H. (2017). Information disorder: Toward an interdisciplinary framework for research and policy making. *Council of Europe report*, *27*.

Yang, J., Vega-Oliveros, D., Seibt, T., & Rocha, A. (2021). Scalable fact-checking with human-in-the-loop. *2021 IEEE International Workshop on Information Forensics and Security (WIFS)*, 1-6.

Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., & Sun, M. (2020). Graph neural networks: A review of methods and applications. *AI Open*, *1*, 57-81. https://doi.org/10.1016/j.aiopen.2021.01.001

Anexo A. Código

El código está disponible públicamente en un repositorio de GitHub, pero se incluye aquí el de los principales ficheros.

scraper.py

```
## Telegram Scraper
# Import libraries
import asyncio
import csv
import os
from telethon import TelegramClient
from telethon.errors import ChannelInvalidError, ChatAdminRequiredError
from datetime import datetime, timedelta, timezone
import pandas as pd

# Set the working directory to the script's directory
os.chdir(os.path.dirname(os.path.abspath(__file__)))

# Prompt the user for the number of days:
try:
    days_to_scrape = int(input("Enter the number of days to scrape (leave blank to
default value, 7): "))
except ValueError:
    print("Invalid input. Using default of 7 days.")
    days_to_scrape = 7

# Prompt the user for the message limit
try:
    MAX_MESSAGES_PER_CHANNEL = int(input("Enter the maximum number of messages to
scrape for each channel (leave blank to default value, 500): "))
except ValueError:
    print("Invalid input. Using default of 500 messages.")
    MAX_MESSAGES_PER_CHANNEL = 500

# Calculate the time for 24 hours ago with timezone information
time_days_ago = datetime.now(timezone.utc) - timedelta(days=days_to_scrape)

# Loading credentials from credentials.txt
def load_credentials(filename):
    credentials = {}
    with open(filename, 'r') as file:
        for line in file:
            key, value = line.strip().split('=')
            credentials[key] = value
    return credentials
creds = load_credentials('credentials.txt')
```

```
API_ID = creds['API_ID']
API_HASH = creds['API_HASH']

# Initialize the client
client = TelegramClient('anon', API_ID, API_HASH)

# Load channels from the CSV file
def load_channels_from_csv(filename):
    channels = []
    with open(filename, 'r', encoding='utf-8') as csvfile:
        reader = csv.reader(csvfile)
        for row in reader:
            if row:  # Check if row is not empty
                channels.append(row[0])
    return channels

CHANNELS = load_channels_from_csv('telegram_channels.csv')

def extract_channel_details(channel):
    return {
        'Channel ID': channel.id,
        'Username': channel.username,
        'Title': channel.title,
        'Description': getattr(channel, 'description', 'N/A'),
        'Photo': channel.photo,
        'Creation Date': getattr(channel, 'date', 'N/A'),
        'Verified': getattr(channel, 'verified', False),
        'Restricted': getattr(channel, 'restricted', False),
        'Members Count': getattr(channel, 'participants_count', 0)
    }

def extract_message_details(message):
    # Construct the URL
    url = f"https://t.me/s/{message.sender_id}/{message.id}"

    # Construct the Embed
    embed = f'<script async src="https://telegram.org/js/telegram-widget.js?22"
data-telegram-post="{message.sender_id}/{message.id}" data-width="100%"></script>'

    return {
        'Message ID': message.id,
        'Message Text': message.text,
        'Date Sent': message.date,
        'Views': message.views,
        'Forwarded From': getattr(message.forward, 'sender_id', None) if
message.forward else None,
        'Reply To': message.reply_to_msg_id,
        'Mentions': message.mentioned,
        'Media': message.media,
        'Entities': message.entities,
        'Edit Date': message.edit_date,
```

```
        'Sender ID': message.sender_id,
        'URL': url,
        'Embed': embed
    }

def extract_media_details(media):
    return {
        'Media Type': type(media).__name__,
        'Media Size': getattr(media, 'size', None),
        'Media Caption': getattr(media, 'caption', None)
    }

async def extract_participants(client, channel):
    try:
        participants = await client.get_participants(channel)
        return {p.id: p.participant for p in participants}
    except ChatAdminRequiredError:
        print(f"Admin privileges required for channel '{channel}'. Skipping
participant extraction.")
        return {}

def load_existing_data(filename):
    try:
        return pd.read_csv(filename)
    except (FileNotFoundError, pd.errors.EmptyDataError):
        return pd.DataFrame()

existing_messages = load_existing_data('telegram_messages.csv')
existing_participants = load_existing_data('telegram_participants.csv')

def load_existing_message_ids(filename):
    try:
        existing_data = pd.read_csv(filename)
        # Create a set of tuples (Username, Message ID) for fast lookup
        existing_ids = set(zip(existing_data['Username'], existing_data['Message
ID']))
        return existing_ids
    except (FileNotFoundError, pd.errors.EmptyDataError):
        return set()

EXISTING_MESSAGE_IDS = load_existing_message_ids('telegram_messages.csv')

async def main():
    all_data = []
    participants_list = []

    for channel in CHANNELS:
        try:
            channel_details = await client.get_entity(channel)
            messages = await client.get_messages(channel,
limit=MAX_MESSAGES_PER_CHANNEL)
```

```
            # Group messages by their date
            grouped_messages = {}
            for message in messages:
                date_str = message.date.strftime('%Y-%m-%d')
                if date_str not in grouped_messages:
                    grouped_messages[date_str] = []
                grouped_messages[date_str].append(message)

            # Calculate daily average views
            daily_average_views = {}
            for date_str, daily_messages in grouped_messages.items():
                views_list = [message.views for message in daily_messages if
message.views is not None]
                daily_average_views[date_str] = sum(views_list) / len(views_list)
if views_list else 0

            for message in messages:
                data = {}
                data.update(extract_channel_details(channel_details))
                data.update(extract_message_details(message))
                # If this message ID for the current channel already exists, skip
it
                if (channel_details.username, message.id) in EXISTING_MESSAGE_IDS:
                    continue

                # Update the URL and Embed columns using the channel's username
                channel_username = data['Username']
                data['URL'] = f"https://t.me/s/{channel_username}/{message.id}"
                data['Embed'] = f'<script async
src="https://telegram.org/js/telegram-widget.js?22" data-telegram-
post="{channel_username}/{message.id}" data-width="100%"></script>'

                date_str = message.date.strftime('%Y-%m-%d')
                data['Average Views'] = daily_average_views.get(date_str, 0)
                data['Average Difference'] = message.views - data['Average Views']
if message.views is not None else None
                if data['Average Views'] == 0:
                    data['Score'] = 0
                else:
                    data['Score'] = message.views / data['Average Views'] if
message.views is not None else None

                    # Convert Score values between 0 and 1 to negative
                    if data['Score'] is not None and 0 < data['Score'] < 1:
                        data['Score'] *= -1

                if message.media:
                    data.update(extract_media_details(message.media))

                all_data.append(data)
```

```
                last_date = messages[-1].date.replace(tzinfo=timezone.utc)

            # Extract participants (might be limited for large channels)
            participants_data = await extract_participants(client, channel)
            for user_id, role in participants_data.items():
                participants_list.append({
                    'Channel': channel,
                    'User ID': user_id,
                    'Role': role
                })

        except ChannelInvalidError:
            print(f"Channel '{channel}' is invalid or not accessible. Skipping to
the next channel.")

    # Convert the new data to a DataFrame
    new_data_df = pd.DataFrame(all_data)

    # Set the 'Message ID' and 'Username' columns as the index for both DataFrames
    if 'Message ID' in existing_messages.columns and 'Username' in
existing_messages.columns:
        existing_messages.set_index(['Message ID', 'Username'], inplace=True)

    if 'Message ID' in new_data_df.columns and 'Username' in new_data_df.columns:
        new_data_df.set_index(['Message ID', 'Username'], inplace=True)

    # Update the existing data with the new data
    existing_messages.update(new_data_df)

    # Reset the index of the existing data
    existing_messages.reset_index(inplace=True)
    new_data_df.reset_index(inplace=True)

    df = pd.concat([existing_messages, new_data_df], ignore_index=True)
    df.drop_duplicates(subset=['Message ID', 'Username'], inplace=True,
keep='first')

    if 'Label' not in df.columns:
        df['Label'] = ''
    df['Label'].fillna('', inplace=True)

    participants_df = pd.concat([existing_participants,
pd.DataFrame(participants_list)], ignore_index=True)
    participants_df.drop_duplicates(subset=['User ID', 'Channel'], inplace=True,
keep='last')

    # Check if the CSV file exists
    file_exists = os.path.isfile('telegram_messages.csv')

    # Convert specific columns and save to CSV
```

```
    # Convertir columnas específicas a entero
    columnas_a_convertir = ['Message ID', 'Sender ID', 'Reply To', 'Forwarded
From', 'Views', 'Members Count']
    for columna in columnas_a_convertir:
        if columna in df.columns and df[columna].dtype not in ['datetime64[ns]',
'timedelta64[ns]']:
            df[columna] = df[columna].fillna(0).astype(int)

    df.to_csv('telegram_messages.csv', index=False, encoding='utf-8')
    participants_df.to_csv('telegram_participants.csv', index=False, encoding='utf-
8')

    # Do the same for participants CSV
    file_exists_participants = os.path.isfile('telegram_participants.csv')
    if file_exists_participants:
        participants_df.to_csv('telegram_participants.csv', index=False,
encoding='utf-8', mode='a', header=False)
    else:
        participants_df.to_csv('telegram_participants.csv', index=False,
encoding='utf-8')

    # Date conversion and save to Excel
    datetime_columns = df.select_dtypes(include=['datetime64[ns, UTC]']).columns

    # Eliminar la información de zona horaria de todas las columnas de fecha y hora
    for col in datetime_columns:
        df[col] = df[col].dt.tz_localize(None)

    if 'Creation Date' in df.columns and df['Creation Date'].dtype ==
'datetime64[ns]':
        df['Creation Date'] = df['Creation Date'].dt.tz_localize(None)
    if 'Date Sent' in df.columns and df['Date Sent'].dtype == 'datetime64[ns]':
        df['Date Sent'] = df['Date Sent'].dt.tz_localize(None)
    if 'Edit Date' in df.columns and df['Edit Date'].dtype == 'datetime64[ns]':
        df['Edit Date'] = df['Edit Date'].dt.tz_localize(None)
    with pd.ExcelWriter('telegram_data.xlsx', engine='openpyxl') as writer:
        df.to_excel(writer, sheet_name='Messages', index=False)
        participants_df.to_excel(writer, sheet_name='Participants', index=False)

# Ensure the client is started before running the main coroutine
with client:
    client.loop.run_until_complete(main())
```

app.py

```
from flask import Flask, render_template, request, jsonify
import pandas as pd

MESSAGES_LIMIT = 48
global df
df = pd.read_csv('telegram_messages.csv')

app = Flask(__name__)

@app.route('/')
def index():
    global df
    if df.empty:
        return render_template('index.html', messages=[])
    sorted_df = df.sort_values(by='Score', ascending=False)
    displayed_df = sorted_df.head(48)
    messages = displayed_df[['Embed', 'Score', 'Message ID', 'URL',
'Label']].to_dict(orient='records')
    return render_template('index.html', messages=messages)

@app.route('/load_more/<int:offset>', methods=['GET'])
def load_more(offset=0):
    global df
    sorted_df = df.sort_values(by='Score', ascending=False)
    displayed_df = sorted_df.iloc[offset:offset+24]
    messages = displayed_df[['Embed', 'Score', 'Message
ID']].to_dict(orient='records')
    return render_template('index.html', messages=messages)

@app.route('/label', methods=['POST'])
def label_message():
    global df  # Declare df as global
    data = request.json
    print(f"Received data: {data}")  # Debugging line

    try:
        message_id = int(request.json['message_id'])
        label = int(request.json['label'])

    except ValueError as e:
        print(f"Error: {e}")
        return jsonify(success=False, error=str(e))

    # Update the DataFrame
    df.loc[df['Message ID'] == message_id, 'Label'] = label
    print(df[df['Message ID'] == message_id])

    # Save the updated DataFrame back to the CSV file
```

```
    df.to_csv('telegram_messages.csv', index=False, encoding='utf-8')

    return jsonify(success=True)

@app.route('/export_relevants', methods=['GET'])
def export_relevants():
    global df
    try:
        # Filter the messages labeled as relevant
        relevant_df = df[df['Label'] == 1]
        # Save to a new CSV
        relevant_df.to_csv('telegram_messages_relevant.csv', index=False,
encoding='utf-8')
        return jsonify(success=True)
    except Exception as e:
        print(f"Error: {e}")
        return jsonify(success=False, error=str(e))

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5001)
```

index.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Telegram Messages Monitorization</title>
    <style>
        .message-container {
            display: flex;
            flex-wrap: wrap;
            justify-content: flex-start;  /* Align items at the start */
        }
        .message-card {
            flex: 0 0 calc(30% - 20px); /* Reduced from 33.33% to 30% */
            margin: 10px;
            box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
            padding: 16px;
        }
        .message-card::after {
            content: "";
            display: table;
            clear: both;
        }
        .button-container {
            text-align: center;  /* Center the buttons horizontally */
            margin-top: 10px;    /* Add some margin at the top for spacing */
        }
        .button-container-general {
            text-align: center;  /* Center the buttons horizontally */
            margin-top: 10px;    /* Add some margin at the top for spacing */
        }
        button {
            padding: 10px 20px;  /* Increase padding for larger buttons */
            font-size: 16px;     /* Increase font size */
            margin: 0 5px;       /* Add margin to space the buttons apart */
            display: inline-block; /* Display buttons inline */
        }

        .score-title {
            font-size: 1.5em;  /* Adjust the size as needed */
            text-align: center;
            margin-bottom: 10px;  /* Add some space between the title and the embed
*/
        }
        .score-value {
            color: red;
        }
        .goto-message-btn {
            display: inline-block;
            text-align: center;  /* Center the buttons horizontally */
```

```
            padding: 10px 20px;
            font-size: 16px;
            margin: 0 5px;
            text-decoration: none;
            background-color: #007bff;
            color: white;
            border: none;
            cursor: pointer;
            border-radius: 4px;
            transition: background-color 0.3s;
        }
        .goto-message-btn:hover {
            background-color: #0056b3;
        }
    </style>
</head>
<body>
    <h1>Overperforming Messages in Telegram</h1>
    <h2>Label messages as relevant or not relevant</h2>
    <div id="messageContainer" class="message-container">
        {% for message in messages %}
        <div class="message-card">
            <!-- Display Overperforming Score as the card title -->
            <h2 class="score-title">Overperforming Score: <span class="score-
value">{{ message['Score']|round(2) }}</span>x</h2>

            <!-- Display the message using the Embed column -->
            {{ message['Embed']|safe }}

            <!-- Buttons -->
            <div class="button-container">
                {% if message['Label'] == 1 %}
                    <button class="relevant-btn" data-message-id="{{
message['Message ID'] }}" style="background-color: green">Relevant</button>
                    <button class="not-relevant-btn" data-message-id="{{
message['Message ID'] }}">Not Relevant</button>
                {% elif message['Label'] == 0 %}
                    <button class="relevant-btn" data-message-id="{{
message['Message ID'] }}">Relevant</button>
                    <button class="not-relevant-btn" data-message-id="{{
message['Message ID'] }}" style="background-color: red">Not Relevant</button>
                {% else %}
                    <button class="relevant-btn" data-message-id="{{
message['Message ID'] }}">Relevant</button>
                    <button class="not-relevant-btn" data-message-id="{{
message['Message ID'] }}">Not Relevant</button>
                {% endif %}
                <a href="{{ message['URL'] }}" target="_blank" class="goto-message-
btn">Go to message</a>
            </div>
        </div>
```

```
        {% endfor %}
    </div>
        <!-- Separate container for the two general buttons -->
    <div class="button-container-general">
        <button id="loadMoreBtn">Load More</button>
        <button id="exportRelevantsBtn">Export relevants</button>
    </div>

    <script>
        document.addEventListener("DOMContentLoaded", function() {
            function labelMessage(messageId, label) {
                fetch('/label', {
                    method: 'POST',
                    headers: {
                        'Content-Type': 'application/json'
                    },
                    body: JSON.stringify({
                        message_id: messageId,
                        label: label
                    })
                })
                .then(response => response.json())
                .then(data => {
                    if (data.success) {
                        console.log(`Labeled ${messageId} as ${label}`);
                        const relevantBtn =
document.querySelector(`button.relevant-btn[data-message-id="${messageId}"]`);
                        const notRelevantBtn = document.querySelector(`button.not-
relevant-btn[data-message-id="${messageId}"]`);
                        const payload = {
                            message_id: messageId,
                            label: label
                        };
                        console.log("Sending payload:", payload);
                        fetch('/label', {
                            method: 'POST',
                            headers: {
                                'Content-Type': 'application/json'
                            },
                            body: JSON.stringify(payload)
                        })

                        if (label === 1) {
                            relevantBtn.style.backgroundColor = "green";
                            notRelevantBtn.style.backgroundColor = "";
                        } else {
                            relevantBtn.style.backgroundColor = "";
                            notRelevantBtn.style.backgroundColor = "red";
                        }
                    } else {
                        console.error('Error labeling the message.');
```

```
                    }
                });
                console.log("Labeling messageId:", messageId, "with value:",
labelValue);
            }

            let offset = 48;

            document.getElementById("loadMoreBtn").addEventListener("click",
function() {
                fetch(`/load_more/${offset}`)
                    .then(response => response.text())
                    .then(data => {
                        document.getElementById("messageContainer").innerHTML +=
data;
                        const script = document.createElement('script');
                        script.src = "https://telegram.org/js/telegram-
widget.js?22";
                        document.body.appendChild(script);
                        offset += 24;
                    });
            });

            document.body.addEventListener('click', function(event) {
                let messageId;
                if (event.target.matches('.relevant-btn')) {
                    messageId = event.target.dataset.messageId;
                    labelMessage(messageId, 1);

                    // Change the button colors
                    event.target.style.backgroundColor = "green"; // set the
relevant button to green
                    event.target.nextElementSibling.style.backgroundColor = ""; //
reset the not-relevant button to its original color
                } else if (event.target.matches('.not-relevant-btn')) {
                    messageId = event.target.dataset.messageId;
                    labelMessage(messageId, 0);

                    // Change the button colors
                    event.target.style.backgroundColor = "red"; // set the not-
relevant button to red
                    event.target.previousElementSibling.style.backgroundColor = "";
// reset the relevant button to its original color
                }
            });

            document.getElementById("exportRelevantsBtn").addEventListener("click",
function() {
                fetch('/export_relevants')
                    .then(response => response.json())
                    .then(data => {
```

```
                        if (data.success) {
                            alert("Relevant messages exported successfully!");
                        } else {
                            alert("There was an error exporting the messages.");
                        }
                    });
            });
        });

    </script>
</body>
</html>
```

model.py

```
import pandas as pd
import numpy as np
import re
import nltk
import os
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_curve, roc_auc_score, auc, confusion_matrix,
precision_recall_curve, average_precision_score, accuracy_score, precision_score,
recall_score, f1_score
from sklearn.linear_model import LogisticRegression
import matplotlib.pyplot as plt
import seaborn as sns

# Set the working directory to the script's directory
os.chdir(os.path.dirname(os.path.abspath(__file__)))

# Load the data
df = pd.read_csv('telegram_messages.csv')

# Text cleaning function
def clean_text(text):
    # Remove URLs
    text = re.sub(r'http\S+|www\S+|https\S+', '', text, flags=re.MULTILINE)
    # Remove mentions and special characters
    text = re.sub(r'\@\w+|\#\w+', '', text)
    text = re.sub(r'\W', ' ', text)
    # Convert to lowercase
    text = text.lower()
    # Remove numbers
    text = re.sub(r'\d+', '', text)
    # Remove extra spaces
    text = re.sub(r'\s+', ' ', text).strip()
    return text

# Apply the cleaning function to the 'Message Text' column
df['Cleaned_Text'] = df['Message Text'].apply(lambda x: clean_text(str(x)))

# Tokenization
df['Tokens'] = df['Cleaned_Text'].apply(lambda x: nltk.word_tokenize(x))

# Remove stopwords
stop_words = set(stopwords.words('english'))
```

```
df['Tokens'] = df['Tokens'].apply(lambda x: [word for word in x if word not in
stop_words])

# Lemmatization
lemmatizer = WordNetLemmatizer()
df['Tokens'] = df['Tokens'].apply(lambda x: [lemmatizer.lemmatize(word) for word in
x])

# Display the processed data
print(df[['Message Text', 'Cleaned_Text', 'Tokens']].head())

# Initialize the TF-IDF vectorizer
vectorizer = TfidfVectorizer(max_features=5000)  # Limiting to 5000 most frequent
words for demonstration

# Fit and transform the cleaned text
tfidf_matrix = vectorizer.fit_transform(df['Cleaned_Text'])

# Split the data into labeled and unlabeled subsets
labeled_df = df.dropna(subset=['Label'])
unlabeled_df = df[df['Label'].isna()]

# Convert the labeled data into features and target
X_labeled = vectorizer.transform(labeled_df['Cleaned_Text'])
y_labeled = labeled_df['Label']

# Split labeled data into training and validation sets
X_train, X_val, y_train, y_val = train_test_split(X_labeled, y_labeled,
test_size=0.2, random_state=42)

# Initialize and train the model
logreg = LogisticRegression(max_iter=1000)
logreg.fit(X_train, y_train)

# Predict labels for the unlabeled data
X_unlabeled = vectorizer.transform(unlabeled_df['Cleaned_Text'])
predicted_labels = logreg.predict(X_unlabeled)

# Update the unlabeled data with the predicted labels
unlabeled_df['Label_predicted'] = predicted_labels

# Combine the labeled and unlabeled data
df_updated = pd.concat([labeled_df, unlabeled_df])

# Save the updated DataFrame
df_updated.to_csv('telegram_messages_updated.csv', index=False)

# Generate predictions for the validation set
y_pred = logreg.predict(X_val)

# Calculate metrics
```

```
accuracy = accuracy_score(y_val, y_pred)
precision = precision_score(y_val, y_pred)
recall = recall_score(y_val, y_pred)
f1 = f1_score(y_val, y_pred)

print(f"Accuracy: {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"F1 Score: {f1:.4f}")

# Generate confusion matrix
cm = confusion_matrix(y_val, y_pred)

# Plot confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='g', cmap='Blues', cbar=False)
plt.xlabel('Predicted labels')
plt.ylabel('True labels')
plt.title('Confusion Matrix')
plt.show()

# Calculate ROC curve
fpr, tpr, thresholds = roc_curve(y_val, logreg.predict_proba(X_val)[:,1])

# Plot ROC curve
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label=f'AUC = {roc_auc_score(y_val, y_pred):.2f}')
plt.plot([0, 1], [0, 1], 'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')
plt.show()

# Calculate precision-recall curve
precision, recall, thresholds = precision_recall_curve(y_val,
logreg.predict_proba(X_val)[:,1])

# Plot precision-recall curve
plt.figure(figsize=(8, 6))
plt.plot(recall, precision, label=f'AP = {average_precision_score(y_val,
y_pred):.2f}')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Recall Curve')
plt.legend(loc='upper right')
plt.show()
```

Anexo B. Artículo de investigación

Al final de la memoria y como un anexo obligatorio deberá incluirse un artículo de investigación que resuma el trabajo realizado y los principales resultados obtenidos. Este artículo deberá seguir la plantilla proporcionada a continuación y tendrá una extensión de entre 6 y 8 páginas. El artículo se podrá desarrollar en español o en inglés (para ello utilizar la plantilla adecuada).

Una propuesta de herramienta para la lucha contra la desinformación en Telegram

Fco. Javier Cantón Correa

Universidad Internacional de la Rioja, Logroño (España)

13/9/2023

Resumen

Este artículo proviene de un Trabajo de Fin de Máster (TFM), perteneciente al Máster de Inteligencia Artificial de la Universidad Internacional de La Rioja (UNIR), y aborda el desafío de la desinformación en plataformas digitales, con un enfoque particular en Telegram. El propósito principal de esta investigación es desarrollar una herramienta basada en técnicas avanzadas de procesamiento del lenguaje natural y aprendizaje automático para combatir la desinformación. La metodología adoptada combina un análisis exhaustivo del estado del arte, la identificación de requisitos y el desarrollo iterativo de la herramienta. Como resultado, se ha creado una solución eficaz que permite monitorizar y verificar contenidos en Telegram, contribuyendo significativamente al campo del *fact-checking*. Las conclusiones destacan la relevancia y potencial de la herramienta en la lucha contra la desinformación, sugiriendo líneas de trabajo futuro para mejorar y expandir su alcance.

Palabras Clave

aprendizaje semi-supervisado, desinformación, fact-checking, Inteligencia Artificial, Telegram

I. Introducción

El avance tecnológico y la proliferación de plataformas digitales han traído consigo un aumento en la circulación de información. Sin embargo, este flujo masivo de datos ha llevado a un problema creciente: la desinformación. El presente artículo resume el Trabajo de Fin de Máster (TFM) realizado en el marco del Máster de Inteligencia Artificial de la Universidad Internacional de La Rioja (UNIR), centrado en abordar la desinformación en Telegram, una de las plataformas digitales más populares.

El propósito principal de esta investigación ha sido el desarrollo y validación de una herramienta que, mediante técnicas avanzadas de procesamiento del lenguaje natural y aprendizaje automático, permita monitorizar y verificar contenidos en Telegram. A través de un análisis exhaustivo del estado del arte, se identificaron los principales desafíos y se establecieron objetivos claros para la herramienta. La metodología adoptada combinó la identificación de requisitos, el desarrollo iterativo y la evaluación de la herramienta.

Como resultado, se ha creado una solución eficaz que no solo aborda el problema de la desinformación en Telegram, sino que también contribuye significativamente al campo del fact-checking. Las conclusiones del trabajo destacan la relevancia y potencial de la herramienta, sugiriendo diversas líneas de trabajo futuro para mejorar y expandir su alcance en el combate contra la desinformación en la era digital.

Este artículo proporciona una visión detallada de la investigación realizada, desde el planteamiento inicial del problema hasta las conclusiones y recomendaciones finales. A través de su lectura, se obtendrá una comprensión clara de lo que se pretendía, las conclusiones alcanzadas y el procedimiento seguido en el desarrollo del proyecto.

II. Estado del Arte

El fenómeno de la desinformación, aunque no es nuevo, ha adoptado nuevas formas en la era tecnológica actual, caracterizada por el surgimiento de la posverdad. Fenómenos como las 'fake news' se enmarcan en lo que expertos denominan 'desórdenes de la información', diferenciando entre 'disinformation', 'misinformation' y 'malinformation' [1]. La preocupación social en torno a la desinformación en España es notable, con un 96% de la población considerándolo un problema y un 91% viéndolo como un peligro para la democracia.

El interés de la comunidad académica en investigar Telegram es reciente, posiblemente debido a su reciente popularidad. Una búsqueda simple en Google Scholar o Semantic Scholar combinando “Telegram” con “disinformation” produce solo unos 15,000 resultados. En contraste, los términos “desinformación” y “disinformation” por sí solos generan más de 100,000 y 220,000 resultados, respectivamente. Utilizando la herramienta Research Rabbit, se ha obtenido una visión general de los principales estudios en este campo en años recientes. El análisis muestra una escasez de artículos significativos y una estructuración de los existentes en torno a áreas específicas de interés.

Las principales temáticas en los trabajos académicos recientes abordan varios temas. Primero, análisis técnicos de la plataforma, enfocándose en características específicas y en el desarrollo de herramientas para capturar y analizar datos de Telegram. Segundo, muchos estudios se centran en el uso de Telegram para activismo y protesta política, especialmente en movimientos como el anti-ELAB en Hong-Kong y grupos de extrema derecha y teorías conspirativas. Finalmente, fuera del ámbito académico, hay numerosos reportajes periodísticos que analizan el papel de Telegram en la propagación de desinformación, examinando cómo se difunden las noticias falsas en la plataforma.

Del primer grupo, destacamos varias herramientas: por un lado, el Pushshift Telegram Dataset [2] es una amplia recopilación de datos de Telegram, abarcando más de 27.800 canales y 317 millones de mensajes de 2,2 millones de usuarios. Creado con el propósito de facilitar investigaciones sobre fenómenos sociales en línea, este conjunto de datos fue recopilado mediante técnicas de web scraping de canales públicos, garantizando la privacidad al excluir conversaciones privadas e información personal identificable. Paralelamente, el 4CAT Capture and Analysis Toolkit [3] es una herramienta de código abierto que permite capturar y analizar datos de diversas plataformas en línea, incluyendo Telegram. Diseñada para ser modular, transparente y trazable, 4CAT facilita la investigación sin requerir habilidades avanzadas de programación, aunque su eficacia puede verse afectada por cambios en las APIs de las plataformas y presenta desafíos éticos relacionados con la privacidad y el consentimiento.

Este trabajo forma parte de ese primer grupo de trabajos centrados en el desarrollo de herramientas que permitan capturar y analizar la aplicación, pero pretende no perder el foco de que se persigue un fin mayor: el análisis de la desinformación dentro de la red para ahondar en la lucha contra las noticias falsas.

III. Objetivos y Metodología

El propósito principal de este Trabajo Fin de Máster es abordar el creciente problema de la desinformación en la plataforma de mensajería Telegram. Este desafío, cada vez más prevalente en la era digital, requiere soluciones innovadoras y efectivas para garantizar la integridad de la información y proteger a los usuarios de contenidos engañosos o falsos. El objetivo general del proyecto es desarrollar una herramienta software que permita monitorizar, analizar y detectar la desinformación en Telegram. Esta herramienta busca no solo identificar contenidos falsos o engañosos, sino también proporcionar a los usuarios y verificadores de información herramientas robustas para combatir la propagación de la desinformación.

Los objetivos específicos del proyecto incluyen la identificación de patrones comunes en la desinformación, la creación de algoritmos de detección basados en técnicas de Inteligencia Artificial y Procesamiento del Lenguaje Natural, y la integración de estas soluciones en una plataforma amigable y fácil de usar.

La metodología adoptada para este proyecto se basa en un enfoque sistemático y estructurado. Se inicia con una fase de introducción y justificación, seguida de la definición de las fases del proyecto. Estas fases incluyen la identificación de requisitos, el diseño y desarrollo de la herramienta, y su posterior evaluación y optimización. A lo largo del proyecto, se han utilizado diversas herramientas y técnicas, incluyendo soluciones de código abierto y herramientas específicas para el análisis y procesamiento de datos en Telegram.

IV. Contribución

La desinformación en plataformas digitales, especialmente en aplicaciones de mensajería como Telegram, representa un desafío significativo en la era actual de la información. La contribución principal de este trabajo es la creación y desarrollo de una herramienta software diseñada específicamente para abordar este problema en Telegram. La herramienta desarrollada en este Trabajo Fin de Máster se basa en una aplicación de Telegram que utiliza técnicas avanzadas de procesamiento de lenguaje natural y aprendizaje automático [4]. Esta herramienta, basada en una arquitectura modular, ofrece una solución integral para monitorizar, analizar y detectar la desinformación en tiempo real. A continuación, se detalla la estructura y la interacción entre los diferentes componentes, programados en Python y HTML:

1. Scraper de Telegram (scraper.py): extrae mensajes de diferentes canales de Telegram, usando la API de Telegram para obtener mensajes y detalles de los canales. Los mensajes se agrupan por fecha y se calcula una media diaria de vistas, guardando los datos extraídos en archivos CSV y Excel.

2. Aplicación principal (app.py): utiliza el framework Flask para crear una interfaz web en HTML que muestra los mensajes de Telegram. Estos mensajes se cargan desde un archivo CSV y se ordenan según una puntuación determinada llamada Score, que calcula el rendimiento de los mensajes de los canales monitorizados comparados con la media diaria. La interfaz permite clasificar los mensajes como relevantes o no para el desarrollo del modelo de IA.

3. Modelo de clasificación (model.py): emplea un modelo de regresión logística para clasificar los mensajes de Telegram. El texto de los mensajes se limpia, tokeniza, lematiza y se convierte en vectores TF-IDF. El modelo se entrena con un conjunto etiquetado a través de la interfaz y se utiliza para predecir etiquetas para mensajes no etiquetados. Se calculan métricas de rendimiento como precisión, exhaustividad y F1 y se visualizan mediante matrices de confusión y curvas ROC.

La interacción entre estos componentes es esencial para el funcionamiento de la herramienta. El scraper (**scraper.py**) alimenta la base de datos con nuevos mensajes de Telegram. La aplicación principal (**app.py**) actúa como la interfaz de usuario, mostrando los mensajes y permitiendo la interacción del usuario. Por último, el modelo (**model.py**) proporciona las capacidades de clasificación, determinando la relevancia de cada mensaje.

La herramienta integra técnicas avanzadas de Inteligencia Artificial y Procesamiento del Lenguaje Natural para analizar y clasificar el contenido extraído. Esto permite no solo identificar mensajes potencialmente falsos o engañosos, sino también entender los patrones y tendencias subyacentes en la propagación de la desinformación. Su estructura modular permite que la herramienta sea escalable y fácilmente extensible. Por ejemplo, se podrían añadir más modelos de clasificación o integrar otras fuentes de datos además de Telegram. La contribución principal de este trabajo es la combinación de técnicas de procesamiento de lenguaje natural y aprendizaje automático para crear una herramienta útil y práctica que pueda ser utilizada en la monitorización de medios sociales para la lucha contra la desinformación.

La contribución de esta herramienta al campo de la detección de desinformación es significativa. No solo proporciona una solución técnica a un problema social crítico, sino que también ofrece una plataforma para la investigación futura, permitiendo a otros investigadores y profesionales del campo construir y mejorar sobre esta base. En resumen, la herramienta desarrollada representa un paso adelante en la lucha contra la desinformación en Telegram, ofreciendo una solución robusta, escalable y basada en la investigación para abordar este desafío en constante evolución.

La contribución de esta herramienta al campo de la detección de desinformación es significativa. No solo proporciona una solución técnica a un problema social crítico, sino que también ofrece una plataforma para la investigación futura, permitiendo a otros investigadores y profesionales del campo construir y mejorar sobre esta base. En resumen, la herramienta desarrollada representa un paso adelante en la lucha contra la desinformación en Telegram, ofreciendo una solución robusta, escalable y basada en la investigación para abordar este desafío en constante evolución.

V. Evaluación y Resultados

La evaluación es un componente vital en el desarrollo de cualquier proyecto de software, y en el contexto de la lucha contra la desinformación, su importancia se magnifica. La herramienta, aún en su fase prototipo, ha sido diseñada para ser intuitiva y eficiente. La velocidad de extracción de datos y la capacidad de respuesta son indicativos de su rendimiento. Sin embargo, como cualquier herramienta en desarrollo, es susceptible de mejoras y adaptaciones. La seguridad y la privacidad son de suma importancia, especialmente en una plataforma donde los usuarios comparten información personal. Además, se contemplan aspectos éticos, como garantizar que la herramienta no introduzca sesgos y respete los derechos de los usuarios. La contribución al campo de la desinformación es evidente, con un enfoque innovador en Telegram, una plataforma menos estudiada en este contexto.

Para garantizar la eficacia y pertinencia de la herramienta, es esencial una evaluación exhaustiva. Se propone una evaluación con usuarios que combine pruebas de usabilidad, aplicabilidad y *feedback* de expertos en el campo de la desinformación. A través de estas evaluaciones, se busca no solo validar la herramienta sino también identificar áreas de mejora y adaptación, garantizando así que la herramienta siga siendo relevante y efectiva en el cambiante panorama de la desinformación.

Este trabajo se ha enfocado en combatir la desinformación en plataformas como Telegram, utilizando técnicas avanzadas de inteligencia artificial y aprendizaje automático. El estudio abordó la desinformación desde una perspectiva multidimensional, considerando no solo el contenido falso, sino también los contextos en los que se propaga. La herramienta propuesta, diseñada para monitorizar y verificar la información, ha sido validada a través de diversas fases y ha demostrado ser eficaz en la identificación y mitigación de la desinformación. Las contribuciones del trabajo incluyen una visión detallada del estado del arte en desinformación, el desarrollo de una herramienta práctica y una propuesta de evaluación exhaustiva.

En cuanto al trabajo futuro, se identifican varias líneas de investigación y desarrollo. Es esencial mejorar la explicabilidad de la inteligencia artificial para que los usuarios comprendan las decisiones de la herramienta. Se propone mejorar la interfaz de usuario, expandir la herramienta a otras plataformas y colaborar con otras herramientas de fact-checking. Además, se destaca la necesidad de investigar la desinformación generada automáticamente por modelos avanzados de lenguaje. En resumen, este TFM ofrece una solución valiosa al desafío de la desinformación y sienta las bases para futuras investigaciones y desarrollos en el campo.

VI. Discusión

Tras la presentación de los resultados y la descripción detallada de la herramienta desarrollada, es esencial reflexionar y discutir sobre su impacto y las implicaciones en el campo de la desinformación en Telegram. Aunque la herramienta ha demostrado ser prometedora en su capacidad para identificar y clasificar mensajes, hay varios aspectos a considerar.

En primer lugar, la naturaleza dinámica de la desinformación significa que las tácticas y estrategias utilizadas por los propagadores de noticias falsas están en constante evolución. Esto plantea la cuestión de cómo la herramienta se adaptará a estas cambiantes tácticas y si será capaz de mantener su eficacia a lo largo del tiempo.

Además, aunque la herramienta se centra en Telegram, la desinformación es un problema que afecta a múltiples plataformas. Sería interesante discutir cómo las técnicas y métodos desarrollados podrían adaptarse o ampliarse para abordar la desinformación en otras plataformas de medios sociales. También es crucial considerar las implicaciones éticas de la monitorización y análisis de mensajes en plataformas como Telegram. Aunque el objetivo es combatir la desinformación, es esencial garantizar que se respeten la privacidad y los derechos de los usuarios.

Mientras que los resultados presentados son prometedores, la discusión destaca la necesidad de una reflexión continua y una adaptación constante para garantizar que la herramienta siga siendo relevante y efectiva en el combate contra la desinformación en el entorno digital actual.

VII. Conclusiones

Este trabajo aborda un desafío contemporáneo de gran relevancia: la desinformación en la plataforma de mensajería Telegram. A través de la investigación y el desarrollo, se ha creado una herramienta que combina técnicas avanzadas de procesamiento de lenguaje natural y aprendizaje automático para identificar, clasificar y analizar mensajes potencialmente engañosos o falsos. Esta herramienta representa una contribución significativa al campo, ofreciendo una solución práctica y basada en la investigación para combatir la propagación de la desinformación.

Sin embargo, como se discutió anteriormente, la lucha contra la desinformación es un esfuerzo continuo que requiere adaptación y evolución constantes. Aunque la herramienta ha demostrado ser efectiva en su estado actual, las evaluaciones futuras y las iteraciones basadas en el feedback serán cruciales para mantener su relevancia. Además, se han identificado varias líneas de trabajo futuro, incluyendo la adaptación de la herramienta para otras plataformas y la integración de características adicionales para mejorar la explicabilidad de los algoritmos de IA. En conjunto, este trabajo sienta las bases para futuras investigaciones y desarrollos en el campo de la detección de desinformación, con el objetivo final de crear un entorno digital más seguro y confiable para todos los usuarios.

Referencias

[1] Wardle, C., & Derakhshan, H. (2017). Information disorder: Toward an interdisciplinary framework for research and policy making. *Council of Europe report*, *27*.

[2] Baumgartner, J., Zannettou, S., Squire, M., & Blackburn, J. (2020). *The Pushshift Telegram Dataset* (arXiv:2001.08438). arXiv. https://doi.org/10.48550/arXiv.2001.08438

[3] Peeters, S., & Hagen, S. (2022). The 4CAT Capture and Analysis Toolkit. *Computational Communication Research*, *4*(2), Article 2.

[4] Cantón, J. (2023). Telegram-app [Python]. https://github.com/Prosumidor/Telegram-app (Obra original publicada en 2023).