Universidad de Alcalá
Escuela Politécnica Superior

Grado en Ingeniería Informática

Trabajo Fin de Grado

Análisis de la evolución de la opinión pública ante las vacunas de la *covid-19* mediante análisis de sentimiento en Twitter

Autor: David Moreno López

Tutor: Pablo Muñoz Martínez
Cotutora: María Dolores Rodríguez Moreno

2021

UNIVERSIDAD DE ALCALÁ
Escuela Politécnica Superior

Grado en Ingeniería Informática

Trabajo Fin de Grado

Análisis de la evolución de la opinión pública ante las vacunas de la *covid-19* mediante análisis de sentimiento en Twitter

Autor: David Moreno López

Tutor: Pablo Muñoz Martínez
Cotutora: María Dolores Rodríguez Moreno

TRIBUNAL:

Presidente: Julia María Clemente Párraga

Vocal 1º: David Fernández Barrero

Vocal 2º: Pablo Muñoz Martínez

FECHA:

A mi familia y amigos.

Agradecimientos

Quiero dar las gracias a mis padres y amigos que me ayudaron a clasificar *tweets*. También al Intelligent Systems Group de la Universidad de Alcalá por permitirme utilizar uno de sus servidores para la captura y almacenamiento de datos, y en especial a Mario por toda la ayuda prestada y sus siempre acertados comentarios.

Resumen

En este trabajo se presenta un análisis de sentimiento para detectar la polaridad de la opinión pública respecto a la vacunación frente a la *covid-19*. A un conjunto de comentarios publicados en Twitter, en idioma español, recopilados durante un amplio periodo de tiempo, se aplican técnicas de procesamiento de lenguaje natural y algoritmos de aprendizaje supervisado. El trabajo continúa con un análisis de la evolución de la polaridad en el tiempo. Por último, a partir de un conjunto de noticias recopiladas durante el mismo periodo, se estudia su influencia sobre el comportamiento de los usuarios de la red social.

Palabras clave: vacuna, Twitter, análisis de sentimiento, aprendizaje supervisado

Abstract

This study presents a sentiment analysis to detect the polarity of the public's opinion regarding vaccination against covid-19. Natural language processing techniques and supervised machine learning algorithms are applied to a set of tweets, in Spanish, compiled during a broad time frame. This was followed by an analysis of the evolution of the polarity throughout time. Lastly, news gathered during the same time span are studied to determine their influence over the social media's users' behaviour.

Keywords: vaccine, Twitter, sentiment analysis, supervised learning

Resumen extendido

La realización de este TFG se encuadra en el contexto originado por el proceso de vacunación contra la *covid – 19*, que ofrece la oportunidad de conocimiento y manejo de herramientas de procesado de lenguaje natural y de aprendizaje automático sobre datos de redes sociales. El objetivo que se persigue es la realización de un análisis de sentimiento en idioma español para detectar la polaridad de la opinión pública respecto a la administración de la vacuna. De este objetivo principal deriva el análisis de la evolución de la polaridad en un periodo prolongado de tiempo y, tomando como referencia un conjunto de noticias recopiladas durante ese mismo periodo, estudiar su influencia sobre el comportamiento de los usuarios de la red social.

Arranca el estudio con una descripción, buscando el estado del arte, de las principales disciplinas y herramientas necesarias para la ejecución del TFG. Se realiza un recorrido por la historia de las redes sociales, destacando su importancia como fuente de datos. Se particulariza en Twitter por su relevancia para comentarios con contenido emocional y las posibilidades de acceso a través de su API.

Esto conduce al ámbito del *big data*, comentando su evolución hasta llegar a su implicación en la gestión de datos de redes sociales y la necesidad de utilizar herramientas ETL. Se destaca el protagonismo del formato JSON y de las bases de datos NoSQL, con MongoDB como referente. También se menciona el otro gran generador de datos que da significado al *big data* en la actualidad: IoT, con sus diferencias de requisitos respecto a las redes sociales.

Y para su manejo, actualmente se utilizan ampliamente las técnicas de aprendizaje automático, en sus variantes tradicionales y las más novedosas, que permiten abordar el análisis de sentimiento, aunque también se indican las técnicas basadas en diccionarios. En este apartado tiene un especial protagonismo el procesamiento de lenguaje natural, ya sea con herramientas tradicionales o con aprendizaje profundo, incorporado más recientemente. Se finaliza este apartado comentando el análisis para el idioma español y el papel que desempeña la SEPLN como impulsora y referente para grupos de investigación.

Todo ello lleva a la elección de Python como el entorno de programación más adecuado para la realización del TFG, que se complementa con Apache NiFi para la captura desde la API de Twitter y almacenamiento en una base de datos MongoDB de *tweets* con comentarios acerca de la vacuna. Se han obtenido datos, en formato JSON, desde mediados de diciembre de 2020 hasta finales de junio de 2021. Suponen 1,3 millones de registros que son limpiados y filtrados,

eliminando duplicados. Se pasa así de los cerca de 150 campos iniciales que tiene un *tweet* a sólo 8, incluyendo uno para sentimiento, y con un resultado de 534.700 registros finales.

El siguiente paso consiste en la clasificación manual de alrededor de 16.800 *tweets,* asignándole un sentimiento positivo, negativo o neutro. Tanto a los tweets clasificados como al resto se les somete a un proceso de limpieza y preparación, utilizando el módulo "NLTK" de Python, con el apoyo de la librería "pandas" para simplificar el manejo de datos. Se unifican formatos; se "tokenizan" los comentarios, es decir que se separan todas las palabras del *tweet* para tratarlas individualmente; se eliminan las "*stop words"*, que son una serie de palabras reservadas que no aportan información a la hora de entrenar al clasificador y es mejor prescindir de ellas; y se procede a su "stemming", que consiste en reducir cada palabra a su raíz.

Entonces entran en juego las funciones del módulo "scikit-learn" de Python, comenzando por la tarea de vectorización para dejar los datos en un estado que permita entrenar al clasificador de la manera más eficiente posible. Se han entrenado varios clasificadores, "SVC" y "LInearSVC" de tipo SVM, y cuatro pertenecientes a la familia de Naïve Bayes: "MultinomialNB", "BernouilliNB", "ComplementNB" y "GaussianNB". El mejor resultado se ha obtenido con el clasificador "SVC" con un 73,56% de precisión, confirmando así la potencia y flexibilidad que proporciona Python para idioma español, con resultados comparables a los encontrados en la literatura.

A partir del análisis de los tweets recopilados y de su evolución en el tiempo se observa una mayor aparición de comentarios con polaridad positiva, con usuarios más activos y con un vocabulario diferente al utilizado para comentarios con polaridad negativa o neutra, que se representan visualmente mediante nubes de palabras. También se ha podido constatar la existencia de una línea base en el número de comentarios diarios para cada polaridad, que se mantiene constante durante todo el periodo, excepto para la polaridad neutra, para la que se observa un descenso en el tiempo, más acusada para los *retweeets* que para los *tweets* únicos.

En paralelo, se han recopilado los hitos y noticias más relevantes que tratan el tema de la vacunación durante el periodo que abarca el estudio, lo que ha permitido detectar la existencia de varios tipos de reacciones ante los hitos mediáticos. Se presentan los efectos "fin de semana", "crecimiento por acumulación", "impacto", "gran impacto", "debate" y "viral", aunque este último no siempre está relacionado con los hitos mediáticos.

El TFG se complementa con varios anexos con información de detalle, un Capítulo de conclusiones y posibilidades de mejora; y una extensa colección de referencias bibliográficas para poder ampliar información de todos los temas relacionados.

Contenido

Figuras

Tablas

Código

Acrónimos y abreviaturas

API Application Programming Interface

BSON Binary JSON

ETL Extract, Transform, Load

IoT Internet of Things

JSON JavaScript Object Notation

ML Machine Learning

MLOps – Machine Learning Operations

NB Naïve Bayes

NLP Natural Language Processing

NLTK Natural Language ToolKit

NoSQL Not only SQL

PLN Procesamiento de lenguaje natural

RRSS Redes sociales

SEPLN Sociedad Española para el Procesamiento del Lenguaje Natural

SQL Structured Query Language

SVC Support Vector Classifier

SVM Support Vector Machines

TASS – Taller de Análisis Semántico en la SEPLN

TFG Trabajo Fin de Grado

UAH Universidad de Alcalá

VPN Virtual Private Network

W3C World Wide Web Consortium

XML Extensible Markup Language

1 Introducción

Aunque seamos seres racionales, a menudo nuestras decisiones están afectadas por sesgos comportamentales. El entorno físico y social influye de manera notable en nuestro comportamiento, incluso más de lo que pensamos o estamos dispuestos a admitir. Esta influencia es especialmente relevante en las decisiones relacionadas con nuestra salud. A pesar de que conocemos los aspectos fundamentales para llevar una vida saludable y del papel predominante de la prevención en el cuidado de nuestra salud a largo plazo, nuestras decisiones a corto plazo a menudo no se ajustan a principios saludables.

El modelo racional, clásico, basado en la maximización de la utilidad esperada, concluye que tomamos decisiones a partir de la información disponible y del conocimiento de qué es lo mejor para nosotros. Cuando la decisión no es óptima, se asume que el modelo se realimenta de la experiencia y que la decisión se acercará más a la óptima la próxima vez.

El modelo comportamental, por otra parte, considera que tomamos decisiones que satisfacen nuestras necesidades (en vez de maximizar la utilidad del resultado) e incorpora aspectos de la psicología para explicar dichas decisiones cuando son sistemáticamente diferentes de las “racionales” o “esperadas”. Tiene en cuenta que no siempre utilizamos toda la información disponible, que sentimos emociones o, simplemente, que caemos en las tentaciones. Y para estas decisiones, a menudo nos apoyamos en nuestro entorno social. Ahí buscamos la información que nos falta, reforzamos nuestras emociones y encontramos excusas para nuestras debilidades [1].

Puesto que nuestro comportamiento está fuertemente influenciado por el entorno social, una forma de cambiar dicho comportamiento pasa por modificar o ajustar nuestra percepción de las normas sociales. Si en nuestro entorno se produce un cambio de opinión, es muy posible que nuestra opinión también se modifique en el mismo sentido. Este cambio de opinión puede estar provocado por una experiencia personal, por un acontecimiento externo, o por una actuación deliberada de terceras personas [2].

Actualmente, nuestro entorno social no se limita a los familiares con los que convivimos, los compañeros con los que asistimos a clase y los amigos con los que compartimos nuestro tiempo de ocio. Primero, con la aparición de internet, y posteriormente con el desarrollo de las redes sociales, las posibilidades de comunicación interpersonal se han multiplicado de forma exponencial. Especialmente entre los jóvenes, la actividad comunicativa ya no puede entenderse sin las redes sociales [3], [Figura 1, Figura 2].

Figura 1. *Digitalización y redes sociales. Fuente: Datareportal*

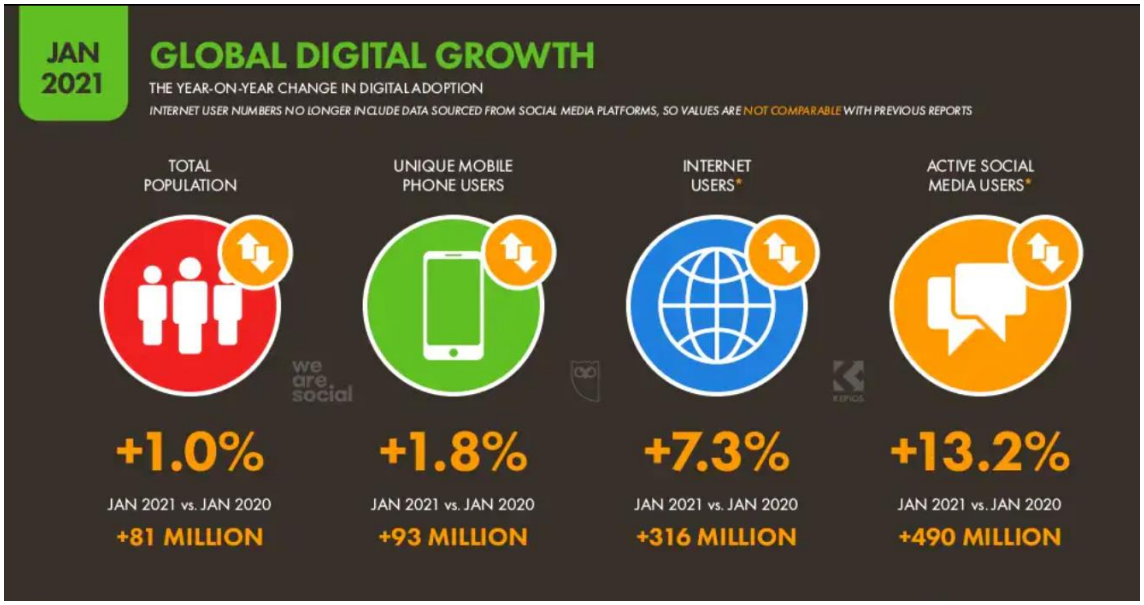

Figura 2. *Crecimiento de las redes sociales en 2021. Fuente: Datareportal*

Mención especial merece Twitter, por ser la fuente de datos del presente Trabajo. Desde su inicio se ha mantenido en los primeros puestos de las redes preferidas por los usuarios y actualmente es la segunda, sólo superada por Facebook, entre las generalistas para la comunicación de opinión y difusión de noticias en español [4], [Figura 3]. Tras su nacimiento ganó popularidad rápidamente debido a sus *tweets* de sólo 140 caracteres y no ha parado de crecer hasta alcanzar cobertura mundial. Lleva años luchando por mantener su papel de red social de referencia, incorporando funcionalidades para mantener a sus usuarios y atraer otros nuevos. Ahora permite incluir fotos o videos, realizar encuestas, ha ampliado el límite a 280 caracteres y ha eliminado dicho límite para mensajes privados.

Figura 3. *Las 5 redes sociales más populares en España. Fuente: Estudio Anual de RRSS 2021 de IAB y Elogia*

El trabajo de recopilación de datos implica tomar decisiones sobre la fuente de datos, conexión a la misma, método de extracción y formato de salida. Una vez obtenidos los datos, su correcta preparación es crucial para poder trabajar sobre ellos, tanto para realizar un buen análisis como para poder visualizarlos de forma adecuada. Además del problema de la existencia de ruido y posibles datos no fiables, también pueden ser de baja calidad. El usuario puede decidir no proporcionarlos o el coste de recopilarlos adecuadamente puede ser elevado (en tiempo, recursos computacionales o dinero). Twitter es un claro ejemplo por la limitación del número máximo de caracteres. En esta fase se suelen utilizar filtros para eliminar los datos de nula o baja calidad y los atributos sin contenido. Cuando esto supone una reducción importante del conjunto de datos, se aplican técnicas para inferir los datos que faltan.

Con los datos ya preparados, su análisis se puede abordar desde diversas perspectivas y con diferentes objetivos. En ocasiones se realizará un análisis centrado en su estructura con fines estadísticos o de estudio de la propia red social, sus peculiaridades y evolución. Otras veces, el estudio irá dirigido al contenido de los mensajes con la intención de buscar un tópico determinado, identificar temas de actualidad o tendencias. Pero hay un interés creciente por estudiar cómo se plasma la opinión de los usuarios en el contenido y demás atributos de los mensajes de la red social y, tratando de llegar un poco más lejos, cómo esa opinión refleja el sentimiento del usuario y qué sentimientos provoca en sus seguidores.

1.1 Motivación

En el momento de abordar este proyecto, la sociedad se encontraba inmersa en la conmoción provocada por la *covid-19* y en pleno proceso de vacunación a la población. La vacunación era el único elemento disponible en ese momento para hacer frente a la pandemia

y, junto a la propia enfermedad, era objeto de constante debate en redes sociales. La opinión a favor o en contra y la decisión de vacunarse o no hacerlo, ha estado influenciada por toda esa actividad en redes sociales e, incluso, puede haber cambiado con el paso del tiempo.

La *covid 19*, que inicialmente parecía ser una enfermedad nueva más, muy similar a otras ya conocidas y que sería fácilmente acotada y controlada, en muy poco tiempo pasó a convertirse en la pandemia con mayor impacto del siglo. Rápidamente saltó de país en país y, en su camino, su repercusión mediática fue aumentando hasta copar la práctica totalidad de medios sociales y ser el tema central de todas las conversaciones. Cualquier noticia que aparecía era objeto de intenso debate en redes sociales. Se hablaba de su origen, propagación, síntomas, posibles tratamientos y sus consecuencias por grupos de edad. No se encontraba un tratamiento realmente eficaz para su cura y parecía que estábamos condenados a la reclusión como única alternativa. Hasta que aparecieron las primeras vacunas.

Y el reflejo mediático de las vacunas fue igual de impactante, con una peculiaridad; había que tomar la decisión de ponerse o no la vacuna, y esta decisión cargó de subjetividad y emotividad los comentarios en redes sociales. Para otras vacunas, como la estacional frente a la gripe, esta decisión siempre ha sido casi totalmente personal. Sin embargo, en el caso de la vacuna frente a la *covid 19* estábamos sobrecargados de información. Casi cada día aparecía una noticia, y esa noticia pasaba, de inmediato, a formar parte del debate social.

Twitter ha sido una fuente inagotable de datos acerca de las vacunas. Además, por el alcance universal del tema, hay suficiente información en español, tanto de noticias como de mensajes en redes sociales. Se dan pues, las condiciones adecuadas para que resulte de interés abordar un trabajo de experimentación con la extracción de datos de redes sociales, preparación de información no estructurada, procesamiento del lenguaje natural, análisis de sentimiento, seguimiento en el tiempo y búsqueda de relaciones con los hitos que se han ido produciendo. Con el añadido de que es posible abordar su estudio para el idioma español y de que hay suficientes herramientas de uso libre para acometer con ellas todas las tareas necesarias durante el proceso.

1.2 Objetivos

El presente Trabajo Fin de Grado tiene como **objetivo principal detectar la polaridad** (positiva, negativa o neutra) de la opinión pública **en idioma español respecto a la vacunación** frente a la *covid-19*. De este objetivo prioritario derivan otros dos: por un lado, a) analizar la **evolución** de esa polaridad en el tiempo, en función del impacto de los hitos (noticias) que se publican en los medios de comunicación; y por otro lado, b) **estudiar, entender y aprender a manejar herramientas** de captura y limpieza de datos de redes sociales, de procesamiento de lenguaje natural y de aprendizaje automático.

Para conseguir tanto el objetivo principal como los objetivos derivados se ha realizado un análisis de sentimiento a partir de datos obtenidos de Twitter durante un periodo de 6 meses, específico para comentarios en español. La captura de datos se ha realizado por medio de un flujo de trabajo creado en Apache NiFi, solicitando *tweets* a la API de Twitter y volcándolos a una base de datos MongoDB en formato JSON. En total, alrededor de 1,13 millones de *tweets*. En paralelo se ha realizado una búsqueda manual de noticias relacionadas con las vacunas, almacenándolas también en una base de datos MongoDB [Figura 4].

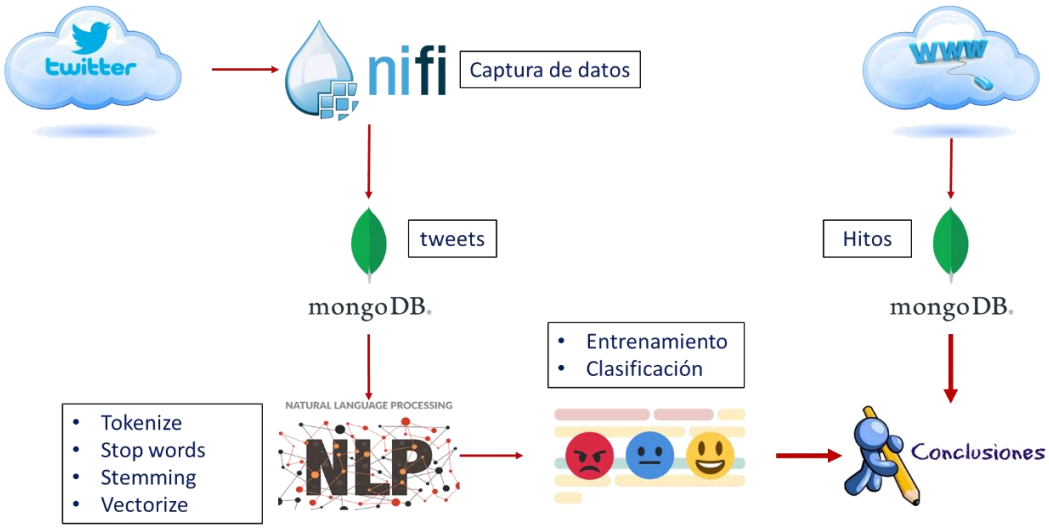

Figura 4. *Diagrama de actividades*

A continuación, se ha procedido a la limpieza y preparación de los datos, en parte de forma manual con MongoDB Compass y la *shell* de Mongo, en parte con Python. En este paso se ha simplificado la base de datos, pasando de los cerca de 150 campos iniciales a 8. También se han eliminado todos los *retweets* duplicados, con lo que la cifra de *tweets* baja a 534.700.

Se ha utilizado la librería NLTK de Python para aplicar procesado de lenguaje natural a los mensajes: "tokenización", que separa cada una de las palabras del tweet para tratarlas individualmente; eliminación de *stop words*, que son una serie de palabras reservadas que no aportan información a la hora de entrenar al clasificador y han de ser eliminada, junto a otras palabras; y "stemmización", que consiste en reducir cada palabra a su raíz. En paralelo, se han clasificado de forma manual alrededor de 17.000 *tweets.* Después se han vectorizado los mensajes clasificados para proceder a la fase de entrenamiento. Se han probado varios algoritmos de aprendizaje supervisado para comparar resultados.

Una vez elegido el mejor algoritmo de entrenamiento y ajustado los parámetros para un entrenamiento óptimo, se ha clasificado el resto de los *tweets,* dando cumplimiento al primero de los objetivos. Se han obtenido diferentes datos estadísticos y se ha analizado la evolución del sentimiento en el tiempo, teniendo en cuenta los momentos en que se han producido hitos relevantes, para así dar por satisfechos los objetivos derivados.

1.3 Líneas de trabajo

Para la realización de este Trabajo Fin de Grado se han seguido tres líneas de actuación:

1 Un proceso inicial, pero también continuado, de búsqueda, estudio y selección de las herramientas y algoritmos más adecuados para cada paso del proyecto. Se ha pretendido en todo momento que hayan probado su eficacia en estudios similares en el pasado y que sean abiertos, flexibles y escalables. El reflejo del esfuerzo dedicado a esta actividad queda plasmado en los Capítulos 1 y 2 de la memoria.

2 El proceso de captura de comentarios de redes sociales, su procesado y la aplicación de técnicas de aprendizaje automático para realizar un análisis de sentimientos.

- Implantación de un flujo de trabajo para acceso a los datos, su captura durante un periodo de tiempo prolongado y su almacenamiento.

- Limpieza, normalización y organización de los datos obtenidos para la aplicación de técnicas de procesado de lenguaje natural.

- Uno de los aspectos más relevantes del proceso ha sido la selección del algoritmo de clasificación y su entrenamiento para predecir el sentimiento de los datos recopilados. El resultado ha sido la clasificación del conjunto de datos, añadiendo a cada comentario un atributo de polaridad que lo caracteriza. Por lo observado durante el proceso de entrenamiento, se ha optado por una métrica multiclase con tres valores de sentimiento: polaridad negativa, neutra o positiva.

3 El proceso de recopilación y clasificación de hitos temporales de relevancia para el trabajo. Generación de una base de datos de hitos a los que se ha añadido una marca temporal y un atributo de sentimiento que refleja su influencia para el proceso de vacunación.

Finalmente, se han analizado las relaciones existentes entre la polaridad de los comentarios y los hitos relevantes que han acontecido durante el periodo de estudio, detectando las variaciones que se han producido tanto en el número de comentarios como en su polaridad. Se presentan los resultados obtenidos y las conclusiones extraídas, así como posibilidades de trabajo futuro. Todo ello queda reflejado a partir del Capítulo 3.

1.4 Estructura de la memoria

En el Capítulo 1 se presenta el contexto en el que se desarrolla el presente Trabajo Fin de Grado, así como la motivación para llevarlo a cabo, los objetivos perseguidos y las líneas de trabajo acometidas.

En el Capítulo 2 se hace un repaso al estado del arte de los temas y áreas de investigación relacionados con el Trabajo y de las herramientas disponibles para acometerlo, incluyendo referencias externas que permiten ampliar información.

En el Capítulo 3 se describen las herramientas empleadas y se explica en detalle todo el proceso que se ha llevado a cabo para lograr el objetivo perseguido. Los resultados obtenidos aparecen en el Capítulo 4, ampliándose en algunos casos en los Anexos. El Capítulo 5 refleja las conclusiones obtenidas y las reflexiones acerca de posibles mejoras y nuevas líneas de trabajo a acometer.

2 Estado del arte

2.1 Redes sociales

Existen diferentes clasificaciones de las redes sociales. Por ejemplo, podemos diferenciar entre redes especializadas y redes generalistas. Al primer grupo pertenecen Flixter (cine), Badoo (relaciones sentimentales), Flickr o Tumblr (diseño y fotografía). Entre las generalistas más populares se encuentran Twitter o Facebook.

También podemos clasificarlas como redes de conexión simétrica o de conexión asimétrica. En el primer caso dos personas deben "aceptarse" para tener relación virtual y vinculación en la red social. En el segundo caso, una de las personas puede seguir las publicaciones de la otra sin aceptación expresa. Como tercer ejemplo, las redes pueden centrar el protagonismo en las personas, y se denominan redes humanas (un claro ejemplo es Whatsapp); o pueden centrarse en el contenido que se divulga y comparte (por ejemplo, Vimeo).

Una de las primeras plataformas que puede considerarse red social fue Geocities (1994), donde se buscaba, principalmente, visibilidad. Le siguieron otras como Classmates (1995), para antiguos alumnos, o Six Degrees (1997), que ya permitía la conexión entre personas y la realización de listas y perfiles. A partir del año 2000 internet cambió sustancialmente, dando lugar a nuevas redes. En 2002 apareció Friendster, para compartir material; en 2003 surgieron MySpace, LinkedIn y Xing. El crecimiento en el número de usuarios comenzó en 2004, con la creación de Facebook, a la que siguieron Youtube y Reddit en 2005, y Twitter y Spotify en 2006.

En 2009, con la aparición de Pinterest, Instagram y Whatsapp, se produce otro gran cambio, hacia redes que fomentan la inmediatez y la personalización. En 2011 cobran popularidad Google+, Twitch y Snapchat. En 2012 surge Tinder, como referente para relaciones personales. De 2014 a 2016 se desarrollan Musically y TikTok, centrados en la publicación de videos musicales breves. En 2020 ha comenzado a extenderse Clubhouse, red especializada en debates de audio [5].

Hace ya tiempo que las redes sociales se utilizan como fuente de información, con una credibilidad creciente acerca de su contenido [6], pero también son fuente de rumores y opiniones no contrastadas que es necesario identificar de forma automatizada [7]. Facebook es desde hace tiempo la red social preferida, aunque paulatinamente va perdiendo peso a favor de Instagram, Twitter, Pinterest y LinkedIn, en ese orden [8]. Sin embargo, Twitter destaca como fuente de información para periodistas, en tres aspectos: descubrimiento de contenido potencialmente interesante, identificación de las fuentes de información más relevantes y

verificación colaborativa del contenido [9]. También se utiliza como canal de referencia en la comunicación durante periodos de crisis [10], ante desastres naturales [11], y en emergencias sanitarias [12].

Twitter dispone de un modelo de entrega de datos sencillo, implementado en una infraestructura muy eficiente [13]. Su API de acceso público permite acceder a los datos publicados en los últimos 7 días, y comercializa accesos "Premium" para los últimos 30 días y "Enterprise" para acceso ilimitado [14]. Se puede acceder a la API con unas pocas líneas de código y desde casi cualquier entorno, como R, Python, Weka, SPSS o Tableau, entre otros muchos [15]. También se han desarrollado aplicaciones que eluden las limitaciones del acceso público [16]. Existen paquetes integrados que incluyen o están centrados en la extracción y análisis de datos de redes sociales, entre los que destacan Netlytic [17], NodeXL [18] o SocioViz [19].

2.2 Big data

Big data es un área de investigación de gran crecimiento y en continua evolución. Se le considera una nueva generación de tecnologías y arquitecturas diseñadas para extraer valor de grandes volúmenes de datos, que varían con frecuencia, posibilitando su captura, identificación y análisis a una velocidad adecuada. Aunque ya se utilizaba el término en 1997, se puede considerar 2001 como un año crucial, al describirse sus tres principales dimensiones (volumen, velocidad y variedad) [20]. Posteriormente se le han añadido otras, destacando las que se refieren a su utilidad (veracidad y valor) [Figura 5].

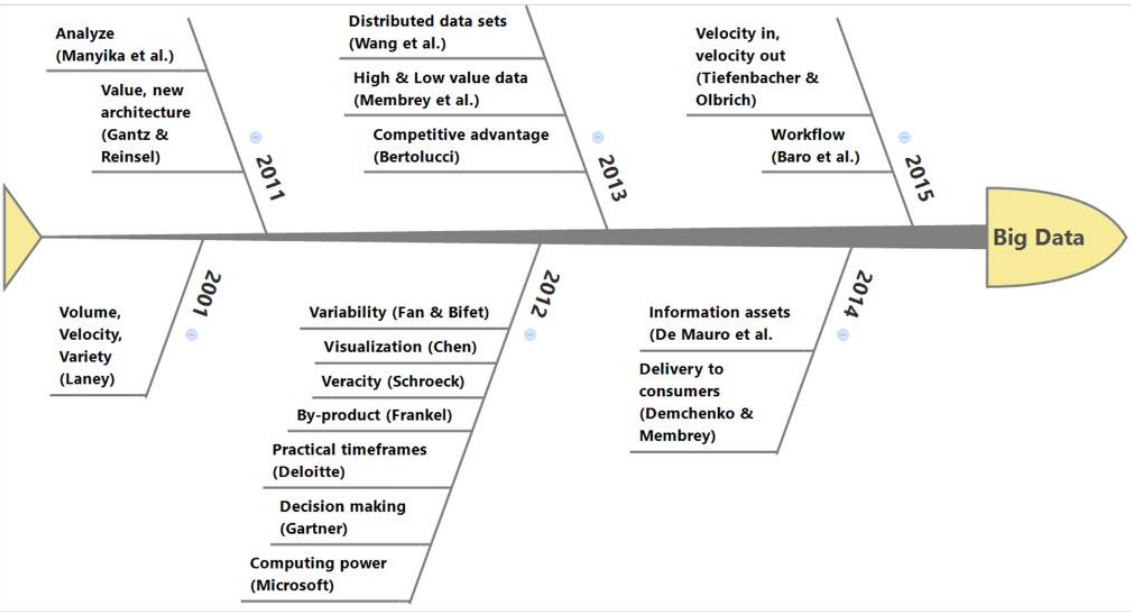

Figura 5. *Evolución de la definición de Big data de 2001 a 2015. Fuente* [100]

A partir de 2012 se produce un nuevo impulso, propiciado por la capacidad tecnológica, y comienzan a desarrollarse otros aspectos, tales como privacidad, seguridad o valor económico.

También se le añade el apellido "*Analytics*", para hacer referencia a los tipos de resultados que se obtienen a partir de su análisis. Se habla así de análisis descriptivo, predictivo y prescriptivo [21]. El análisis descriptivo pretende obtener conocimiento *a posteriori* (responder a la pregunta "¿qué ocurrió?"). El análisis predictivo busca construir modelos a partir de los datos existentes, de forma que se puedan generar datos que no existen o no se conocen (pasados, presentes o futuros). Por último, el análisis prescriptivo va un paso más allá, utilizándose para recomendar una o varias líneas de actuación y mostrar el resultado probable de su aplicación.

Llegados a este punto, resulta sencillo concluir que las redes sociales y *big data* estaban destinados a encontrarse[1]. Las redes sociales cumplen todos los parámetros descritos para ser consideradas una fuente prácticamente inagotable de datos y, cada vez más, los investigadores están descubriendo su utilidad descriptiva, predictiva y prescriptiva. A las dificultades que plantea el análisis de los datos, en el caso de las redes social se unen los retos de su recopilación y preparación para el análisis [22], [Figura 6]. Y los campos de aplicación parecen también no tener fin: negocios, ocio, gestión de crisis, política y, por supuesto, comportamiento y toma de decisiones.

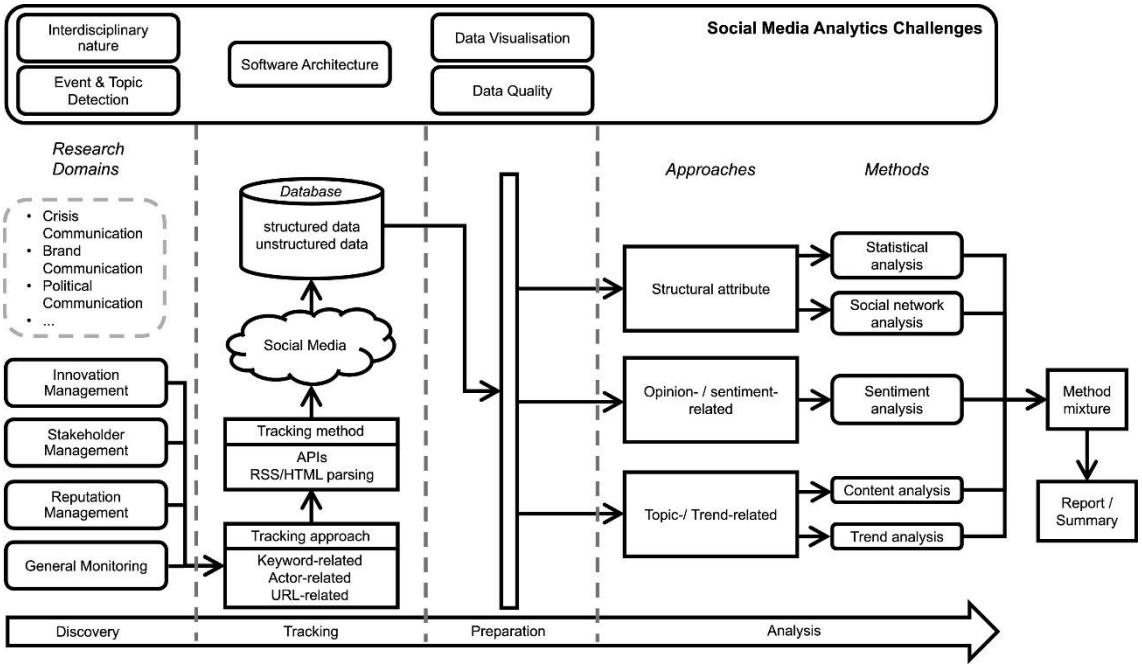

Figura 6. *Estructura del proceso de análisis de datos de redes sociales. Fuente* [22]

Todas las plataformas de redes sociales, y Twitter en particular, generan datos en múltiples formatos, que pueden ser estructurados o no estructurados. Los primeros son datos (como el número de seguidores en Twitter) con una estructura interna claramente definida que permite un procesado automatizado relativamente sencillo. Los datos no estructurados (como el texto del *tweet*) no tienen una estructura interna fácilmente identificable y necesitan un tratamiento específico para que resulten de utilidad.

Con el auge del *big data* han aparecido herramientas de tipo ETL que gestionan el flujo de datos de extremo a extremo y que incluyen funcionalidades para acceso a redes sociales. Las

[1] Y así aparecen términos propios como "*social media big data*" y "*social media analytics*".

principales son Apache NiFi [23], Streamsets [24], Apache Airflow [25], AWS Data Pipeline [26], AWS Glue [27], Talend [28], Informatica PowerCenter [29], Ab Initio [30], Pentaho [31] y Azure Data Factory [32].

La API de Twitter entrega los datos en formato JSON [33], con una estructura relativamente compleja que no es inmediato transformar para su almacenamiento en bases de datos relacionales tradicionales. Por ese motivo, los investigadores prefieren utilizar bases de datos NoSQL, que admiten el formato JSON de forma nativa, como MongoDB [34]. En cualquier caso, la gestión de bases de datos en el entorno empresarial aún está dominada por los sistemas relacionales [35], [Figura 7].

Rank			DBMS	Database Model
Sep 2021	Aug 2021	Sep 2020		
1.	1.	1.	Oracle	Relational, Multi-model
2.	2.	2.	MySQL	Relational, Multi-model
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model
4.	4.	4.	PostgreSQL	Relational, Multi-model
5.	5.	5.	MongoDB	Document, Multi-model
6.	6.	↑ 7.	Redis	Key-value, Multi-model
7.	7.	↓ 6.	IBM Db2	Relational, Multi-model
8.	8.	8.	Elasticsearch	Search engine, Multi-model
9.	9.	9.	SQLite	Relational
10.	↑ 11.	10.	Cassandra	Wide column

***Figura 7**. Sistemas de gestión de bases de datos más populares. Fuente* [35]

JSON apareció como alternativa a XML, estándar basado en texto desarrollado por el W3C para el intercambio de información entre sistemas, independiente del sistema operativo. XML aporta el rigor necesario para la comunicación entre máquinas y resulta adecuado para el uso *web* [36]. JSON resulta mucho más legible para humanos e igual de eficaz que XML para la comunicación entre máquinas. Aunque la utilización de MongoDB está ampliamente extendida, existen otras alternativas para el manejo de bases de datos NoSQL, como Apache Cassandra [37], Amazon DynamoDB [38], CouchDB [39], RethinkDB [40], OrientDB [41], e, incluso, PostgreSQL [42].

2.3 Aprendizaje automático

El elevado crecimiento del *big data* de los últimos años tiene dos protagonistas. Por un lado, la proliferación de dispositivos IoT. Estos dispositivos generan mensajes generalmente cortos, intermitentes y muy estructurados [43]. Sus aplicaciones suelen estar relacionadas con ciudades inteligentes [44] o entornos industriales [45]. Con la llegada de la computación en la nube y el 5G, las plataformas de procesado de los datos están cada vez más cerca de sus fuentes, aprovechando las funcionalidades del *edge computing*, *cloud computing* y *fog computing* [46].

El otro protagonista son las redes sociales que, por el contrario, generan mensajes de tamaño indeterminado, cargados de subjetividad y muy poco estructurados. Para su análisis se recurre a una gran variedad de técnicas, herramientas y plataformas [47]. Previamente es necesaria su limpieza e integración, para lo que se recurre a tecnologías como Spark [48], Hadoop [49] o Mesos [50]. Entre las técnicas más novedosas de procesado de datos se encuentran las que pertenecen al ámbito de la inteligencia computacional, que se centra en el estudio de mecanismos adaptativos para entender el comportamiento de sistemas complejos y cambiantes: redes neuronales , sistemas borrosos, *swarm intelligence*, computación evolutiva y aprendizaje profundo [51].

El incesante y variado flujo de datos que generan las redes sociales no es sencillo de almacenar, procesar y analizar para extraer conocimiento. Requiere mecanismos muy eficientes con un grado de automatización muy elevado, como los que ofrecen las técnicas de aprendizaje automático (*ML – Machine Learning*) [52]. Surgidas en el seno de la inteligencia artificial, proporcionan a los sistemas la capacidad de adaptarse cuando se les proporcionan nuevos datos sin ser programados explícitamente para ello. El aprendizaje automático utiliza los datos para detectar patrones y ajustar los parámetros del programa en consecuencia.

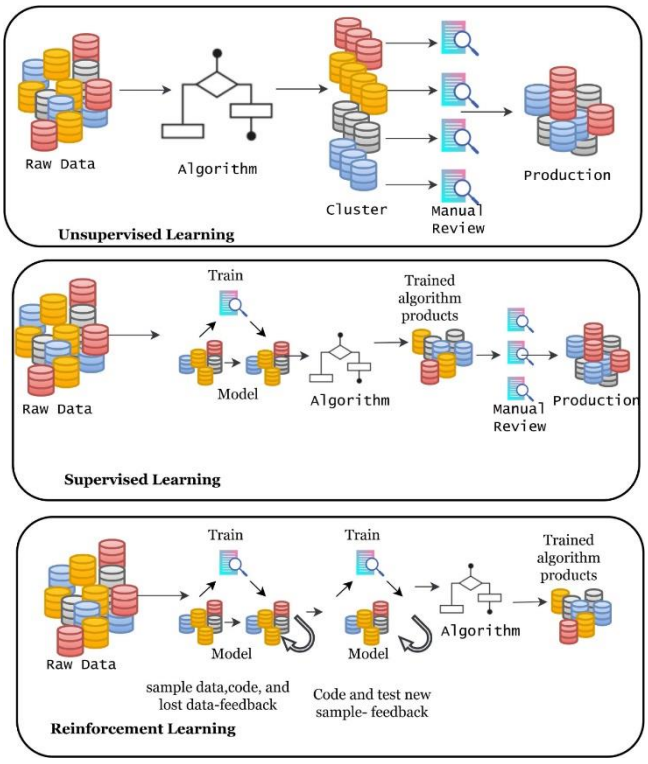

Figura 8. *Aprendizaje supervisado, no supervisado y por refuerzo. Fuente* [52]

La gran variedad de algoritmos de aprendizaje automático se clasifica tradicionalmente en tres grandes tipos: aprendizaje supervisado, no supervisado y por refuerzo [Figura 8 , Figura 9]. El aprendizaje supervisado resulta útil cuando queremos añadir a los datos un nuevo atributo. Partimos de un conjunto de datos que disponen de dicho atributo, con el que “entrenamos” al algoritmo, de forma que sea capaz de asignar dicho atributo a un nuevo conjunto de datos. Un ejemplo clásico es la detección de correo de *spam*. En el aprendizaje no supervisado, el

algoritmo se enfrenta directamente a los datos de entrada, buscando patrones que permitan clasificar o agrupar los datos en clases atendiendo a sus similitudes. Es la técnica que utiliza el recomendador de Netflix. Finalmente, en el aprendizaje por refuerzo el algoritmo aprende a partir de los datos y es premiado o penalizado por las decisiones que toma, en un proceso de ajuste y realimentación. Este es el caso, por ejemplo, de los coches autónomos. En el mundo real, los conjuntos de datos suelen tener características que hacen aconsejable combinar algoritmos de diferentes tipologías. Se habla entonces de aprendizaje semi-supervisado.

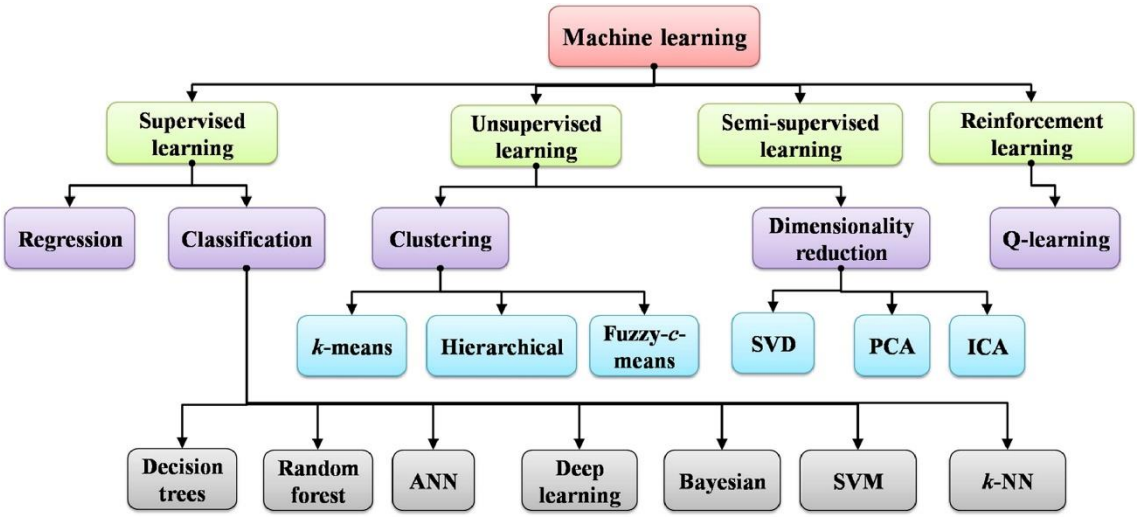

Figura 9. *Algoritmos de aprendizaje automático. Fuente* [52]

Entre las incorporaciones más recientes a esta disciplina podemos citar el aprendizaje basado en reglas de asociación [53], la programación lógica inductiva [54], los diccionarios para representación escasa [55], el aprendizaje por representación o transformación [56], los algoritmos genéticos [57] o el aprendizaje por métricas y similitudes [58]. Las principales áreas de aplicación del aprendizaje automático a partir de los datos de las redes sociales son: análisis de imágenes, relaciones y reputación, detección de eventos, detección de anomalías, inteligencia de negocio, bioinformática, recomendaciones, epidemias, análisis de sentimiento, análisis de comportamiento y detección de delitos.

2.4 Procesamiento de lenguaje natural

El análisis de sentimiento pretende extraer de un texto las emociones, opiniones, actitudes y sensaciones de su autor. Se puede utilizar para clasificar un mensaje como positivo, negativo o neutro acerca de un determinado tema u opinión [59]. El objetivo es poder asignar a un texto (de Twitter, por ejemplo) un atributo que permita extraer conclusiones sobre el autor de dicho texto. En función del nivel de influencia de dicho autor dentro de la red social, también es posible anticipar la reacción de un grupo social más amplio. Aplicando de nuevo técnicas de aprendizaje automático, es posible hacer predicciones sobre los resultados de unas elecciones, la evolución del mercado bursátil, la propagación de enfermedades o el aumento de la alarma social en determinadas zonas, entre otras muchas.

Existen tres grupos de técnicas: las basadas en diccionarios, las que utilizan aprendizaje automático y las que proponen un enfoque híbrido [60]. Las primeras utilizan una colección de términos para buscar ocurrencias en el texto a analizar. Para el idioma inglés existen algunos diccionarios muy elaborados, como SentiWordNet [61], SentiWords [62] o VADER [63]. Para el idioma español los diccionarios existentes no arrojan buenos resultados, por lo que se recurre a la adaptación de alguno de los diccionarios en inglés. Este es el caso de CRISOL [64] o Elhuyar [65]. También es posible crear uno propio a partir del análisis estadístico de un grupo de documentos

Las técnicas que utilizan aprendizaje automático se basan en el procesamiento del lenguaje natural (NLP – Natural Language Processing) y se dividen en dos grupos, las que siguen modelos tradicionales y las que utilizan aprendizaje profundo. Las primeras utilizan algoritmos como Naïve Bayes [66] o SVM [67] y su precisión depende en gran medida de un ajuste correcto de los parámetros empleados. Estas son las técnicas utilizadas en el presente TFG. Los modelos basados en aprendizaje profundo utilizan redes neuronales profundas, recurrentes o convolucionales y suelen obtener mejores resultados [68].

Existen herramientas ya configuradas, accesibles en la nube o mediante API. Además de análisis de sentimiento, permiten diversas clasificaciones de textos, extracción de temas, análisis morfológico y sintáctico, análisis de la estructura de los documentos y resumen automático, entre otras. Permiten personalizar los diccionarios, los modelos de clasificación y los modelos de sentimiento. Destacan MeaningCloud [69], Google API [70], Azure API [71], Watson Natural Language Understanding [72] o Sumy [73], que aunque se trata simplemente de una librería gratuita y un intérprete de comandos, proporciona muchas funcionalidades. Estas herramientas no consiguen mejores resultados que los programas a medida [74] [Figura 10].

Tool	Accuracy	Precision	Recall	Macro F1
MeaningCloud	0.536	0.447	0.449	0.437
Google	0.539	0.355	0.397	0.364
Azure	0.585	0.485	0.479	0.480

Figura 10. *Resultado de herramientas de análisis de sentimiento en la TASS 2019*

Para idioma español existe el grupo de trabajo TASS [75] organizado por la SEPLN. Se trata de una competición que se celebra desde 2012 para impulsar la investigación en análisis de sentimiento en español. Dispone de un conjunto de datos obtenidos de Twitter, con polaridad asignada. La SEPLN sirve de punto de encuentro para la multitud de grupos de investigación que están dedicados al PLN, generalmente en entornos universitarios [76]. También es la encargada de dictar la estrategia en el PLN [77]. Para el idioma español no hay diccionarios muy elaborados y de las técnicas tradicionales SVM suele ser la que mejor resultados proporciona. En la actualidad hay dos tendencias: por un lado, utilizar métodos que combinan SVM con la unión de varios diccionarios y, por otro lado, la utilización de redes neuronales [78], o combinaciones de ambas.

2.5 Python

Python [79], es un lenguaje de programación de uso libre, interpretado, dinámico y multiplataforma, enfocado a la legibilidad de su código. La inmediatez con la que permite comenzar a escribir código hace que se encuentre a la cabeza de los lenguajes más populares, aunque el más utilizado sigue siendo C. También es el lenguaje de programación para aprendizaje automático utilizado con más frecuencia en GitHub [Figura 11]. Surgió en 1989, pero hasta 1994 no se distribuyó la versión 1.0, trabajando ya desde la programación orientada a objetos y el tipado dinámico. En el año 2000 aparece la versión 2.0, que incluía la generación de listas, una de sus características más importantes. En el año 2008 se distribuye la versión 3.0, marcando un punto de inflexión, pues deja de ser compatible hacia atrás.

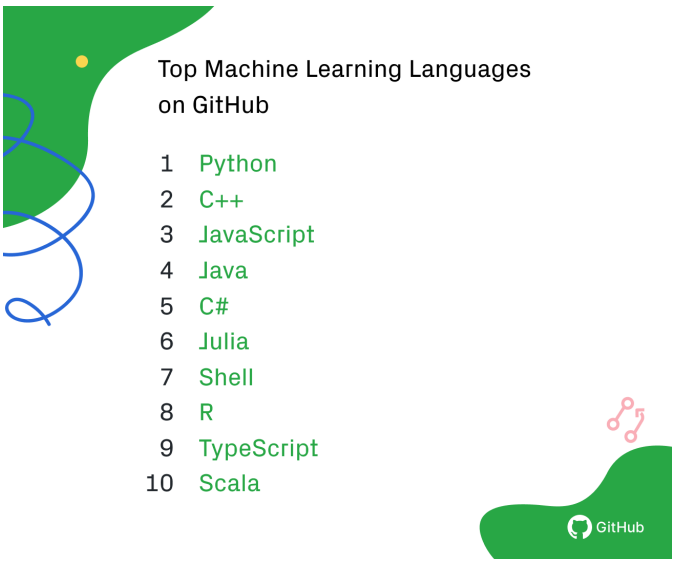

Figura 11. *Los 10 lenguajes de programación preferidos para aprendizaje automático (2019). Fuente GitHub*

Posee cientos de bibliotecas que hacen que pueda utilizarse para cualquier tipo de proyecto, ya sea una aplicación móvil, web, de ciencia de datos o de aprendizaje automático [80]. Entre las que podemos considerar fundamentales se encuentran *NumPy* y *SciPy* para cálculo numérico y *Cython* para la integración con código de terceros. Para preparación de datos la mejor y más utilizada es *pandas*, seguida de lejos por *PyTables*, *h5py* y *Tabel*. Para visualización destacan *Ploty*, *seaborn*, *Matplotlib*, *Bokeh* y *ggplot.*

En el ámbito del aprendizaje automático, la más popular es *scikit-learn*, aunque dispone de otras como *mlxtend* y *Shogun*, con algunos algoritmos adicionales. Para aprendizaje profundo destacan *TensorFlow*, aunque está escrita a bajo nivel y no es inmediato su dominio; sobre ella se ha construido *Keras*, y también destaca *PyTorch*, desarrollada y utilizada por Facebook. A cierta distancia se encuentra *Caffe* (y su más reciente versión, *Caffe2*). Para procesamiento de lenguaje natural, además de NLTK [81], dispone de *gensim* y *spaCy*.

Existen multitud de entornos de desarrollo que facilitan la programación, depuración y ejecución. Entre los principales se encuentran PyCharm [82], PyDev [83], Spyder [84] y el propio de Python, IDLE [85]. También está disponible en algunos editores de texto como Sublime Text [86], Atom [87] o los populares Vim [88] y Emacs [89].

3 Implementación

Este Capítulo se compone de dos partes. Comienza con una descripción de las principales herramientas utilizadas para la ejecución del TFG, para continuar con una descripción detallada de las diferentes fases de su desarrollo.

3.1 Herramientas utilizadas

3.1.1 tweets

La unidad básica de información en Twitter es el objeto denominado *tweet* [90]. Un *tweet* es un conjunto de parejas clave-valor (atributos etiquetados y sus valores asociados) codificados en JSON que, a su vez, puede contener otros objetos anidados para representar otros atributos, como los relativos al autor, menciones, archivos multimedia o enlaces web. Cuando el objeto anidado es otro *tweet*, permite representar *tweets* relacionados entre sí. Esta relación puede ser de tres tipos, y los *tweets* asociados se denominan *Retweet*, *Quote* o *Reply*. Así pues, un *tweet* no tiene una estructura fija, su formato depende del tipo de objeto que se trate.

Un *retweet* es uno de los objetos utilizados con más frecuencia para involucrarse o extender una conversación en Twitter, y consiste en compartir con tus seguidores un *tweet* ya publicado por otro usuario. Este objeto contiene el atributo “retweet_count” que indica el número de veces que se ha vuelto a publicar hasta ese momento y el objeto anidado “retweeted_status”, que contiene el *tweet* original al que se hace referencia, con sus atributos (los *tweets* intermedios no se conservan). La existencia de este atributo es el que permite identificar de forma inequívoca este tipo de objeto, aunque hay otras evidencias, como el comienzo del *tweet* raíz con el texto “RT @username”, donde “username” es el autor del *tweet* original.

Un *quote tweet* es similar a un *retweet*, pero en este caso incluye un nuevo mensaje, además de la referencia al *tweet* original. Este nuevo mensaje puede contener su propio conjunto de atributos, independientes de los que contiene el *tweet* al que hace referencia. Entre sus atributos se encuentran “quote_count”, que indica de forma aproximada el número de veces que se ha hecho mención al *tweet* original hasta ese momento, y el objeto anidado “quoted_status”, que contiene el *tweet* original que se menciona, con sus atributos. De nuevo,

la existencia de este atributo es el que permite identificar de forma inequívoca este tipo de objeto.

Un *reply tweet* es un objeto con la misma estructura que un *quote tweet*, pero su comportamiento y visibilidad en la red social es muy diferente. Cuando, desde la interfaz de usuario de Twitter, generamos un *reply tweet*, este nuevo *tweet* aparecerá en la conversación del usuario que generó el *tweet* original (al que estamos respondiendo). Cuando generamos un *quote tweet*, el nuevo *tweet* aparecerá en nuestra propia conversación, igual que ocurre con un *retweet.*

3.1.2 Apache NiFi

Apache NiFi es una herramienta multiplataforma, distribuida, gratuita y *open source,* diseñada para automatizar y gestionar el flujo de información entre sistemas en tiempo real [91]. Está desarrollada y mantenida por la Apache Software Foundation. Permite realizar operaciones de extracción, transformación y almacenamiento de grandes cantidades de datos con gran flexibilidad, utilizando una interfaz visual muy intuitiva, basada en web. Su elemento básico es el **procesador**, encargado de ejecutar las operaciones de extracción, transformación o carga de los datos; dispone de cerca de 300 procesadores ya implementados, que pueden ser personalizados en gran medida.

Entre sus ventajas se puede destacar que posee una licencia Apache, que está construida en torno al concepto de programación de flujos de datos, la posibilidad de manejar datos binarios, el gran número de componentes disponibles o su interfaz de usuario sencilla y visual. Como inconvenientes se pueden mencionar la escasez de estadísticas disponibles, que no está optimizado para realizar transformaciones de datos complejas y que realiza un consumo de recursos elevado.

En NiFi se comienza definiendo un flujo de trabajo (*flow*) para indicar la forma en que deben gestionarse los datos, que se agrupan en archivos de flujo (*flowfile*) que viajan entre los procesadores. Cada uno de estos archivos contiene un puntero a los datos (NiFi no maneja el propio dato) y una serie de atributos que les caracterizan. Muchas de las operaciones que se realizan en NiFi no llegan a modificar el dato ni necesitan cargarlo en memoria, el cual se encuentra almacenado en lo que se denomina repositorio de contenido. Además de los procesadores, hay otros componentes, como "**conexiones**", para facilitar la interacción entre procesadores y el flujo de archivos, gestionando colas, caducidades y prioridades; "**grupos de procesadores**", para tratar varios procesadores como una unidad lógica independiente, definiendo puertos de entrada y salida; y "**controladores**", que se utilizan para compartir un recurso entre varios procesadores o con conexiones externas.

3.1.3 NoSQL y MongoDB

Las bases de datos NoSQL pueden administrar altos volúmenes de datos no estructurados que cambian con rapidez de forma más eficiente que una base de datos relacional, con filas y tablas. Las tecnologías NoSQL existen desde hace tiempo, pero se están haciendo cada vez más populares, gracias al *big data*, la importancia creciente de los metadatos y la diversidad de

fuentes de datos: “en la nube”, dispositivos móviles, redes sociales, etc. A la inversa, las bases de datos NoSQL también pueden gestionar datos muy estructurados. Los tipos más comunes de bases de datos NoSQL son: tablas *hash* de pares clave-valor, documentos organizados en colecciones, datos basados en columnas y datos de grafos [92]. La API de Twitter devuelve los *tweets* como documentos en formato JSON, que pueden tener incrustados otros *tweets* como objetos anidados.

Será preferible una base de datos NoSQL cuando:

- Tenemos grandes volúmenes de datos, no relacionados, indeterminados o que cambian rápidamente.
- Los datos son independientes del esquema establecido por la aplicación.
- El rendimiento y la disponibilidad son más importantes que una coherencia de datos alta.
- Aplicaciones siempre activas con usuarios distribuidos.

Por el contrario, puede ser aconsejable utilizar bases de datos relacionales cuando:

- Los datos tienen requisitos lógicos y discretos que se pueden identificar con antelación.
- El esquema se debe mantener sincronizado entre la aplicación y la base de datos.
- Se trata de sistemas heredados creados para estructuras relacionales.
- Aplicaciones que requieren consultas complejas.

MongoDB es una base de datos orientada a documentos con una gran escalabilidad y flexibilidad, junto a un modelo de consultas e indexación avanzado. Se puede utilizar en la nube (MongoDB Atlas) o como servidor local, con dos versiones, “Community”, de uso libre y “Enterprise”, integrada en un paquete avanzado de pago. Los documentos son pares Atributo-valor que se almacenan en formato BSON [93], una representación binaria de documentos JSON que permite más tipos de datos. Puede contener otros documentos, *arrays* y *arrays* de documentos, utilizando la “notación punto” para acceder a ellos.

El acceso al servidor puede realizarse a través de una *shell*, por medio de la interfaz gráfica MongoDB Compass, o directamente desde Visual Studio Code. Aunque en el momento de realizar este Trabajo ya existe la versión 5.0 de MongoDB, se ha optado por utilizar la versión 3.6, para facilitar la compatibilidad con el resto de las herramientas [94].

3.1.4 pandas

“pandas” [95] es una librería de Python con las estructuras y funciones fundamentales para manipulación y análisis de datos, posiblemente la más flexible y potente de cualquier lenguaje de programación. Concebida como extensión de “Numpy”, sus principales características son:

- El objeto *dataframe*, estructura bidimensional con indexación jerárquica integrada compuesta por “series” que, a su vez, son *arrays* unidimensionales capaces de almacenar cualquier tipo de datos. Permite su creación a partir de listas, diccionarios y sus combinaciones.
- Herramientas muy sencillas de lectura y escritura de datos en multitud de formatos.

- Orientado a inspección, selección, agregación y transformación flexible de datos. Permite encadenar operaciones.
- Gestión inteligente de datos incompletos o desordenados.
- Integrado con la librería gráfica “Matplolib”.
- Abierto a la incorporación de nuevas funcionalidades o modificación de las existentes.

3.1.5 NLTK

“NLTK” [96] es el módulo de referencia para procesamiento de lenguaje natural en Python, buscando la sencillez, consistencia, modularidad y escalabilidad. Permite interactuar de forma sencilla con más de 50 recursos léxicos y dispone de una amplia librería de procesamiento de texto. En concreto, permite, para cada idioma:

- Procesar cadenas de texto: *Tokenize,* para separar los elementos que componen un texto. Trabajar con *stop words,* permitiendo añadir nuevas palabras al diccionario. *Stemming,* para obtener la palabra raíz de un vocablo.
- Realizar análisis estadístico y aplicar algoritmos de aprendizaje automático.
- Etiquetar y procesar bloques de texto y realizar análisis semántico.

3.1.6 scikit-learn

Construida sobre “Numpy”, “SciPy” y “matplolib”, “scikit-learn” [97] es la librería de Python para aprendizaje automático y análisis de datos. Se caracteriza por su interfaz simple y consistente y por la gran cantidad de herramientas que implementa, tanto de aprendizaje supervisado como no supervisado [Figura 12]. Para el presente TFG se han utilizado:

- “**TfidVectorizer**”, del módulo “sklearn.feature_extraction”, para transformar el texto en una representación numérica, asignando una frecuencia a cada palabra comparando el número de veces que aparece en un documento con el número de documentos en los que aparece.
- “**train_test_split**” del módulo “model_selection”, para el entrenamiento del clasificador. Divide de forma aleatoria el conjunto de datos clasificados manualmente en dos subconjuntos, uno para entrenar el clasificador y otro para aplicar el algoritmo ya entrenado sobre él y probar su eficacia.
- “**metrics**”, para calcular la matriz de confusión y presentar los resultados de la clasificación. Durante la fase de entrenamiento del algoritmo de clasificación, la matriz de confusión consiste en una tabla con tantas filas y columnas como el número de valores del atributo a clasificar, indicando cuántos de los registros del conjunto de prueba se han clasificado correctamente (diagonal principal) y cuántos se han clasificado en el resto de etiquetas.
- **Algoritmos de clasificación “naïve_bayes” y “svm”**. Su descripción puede consultarse en el Anexo 1.

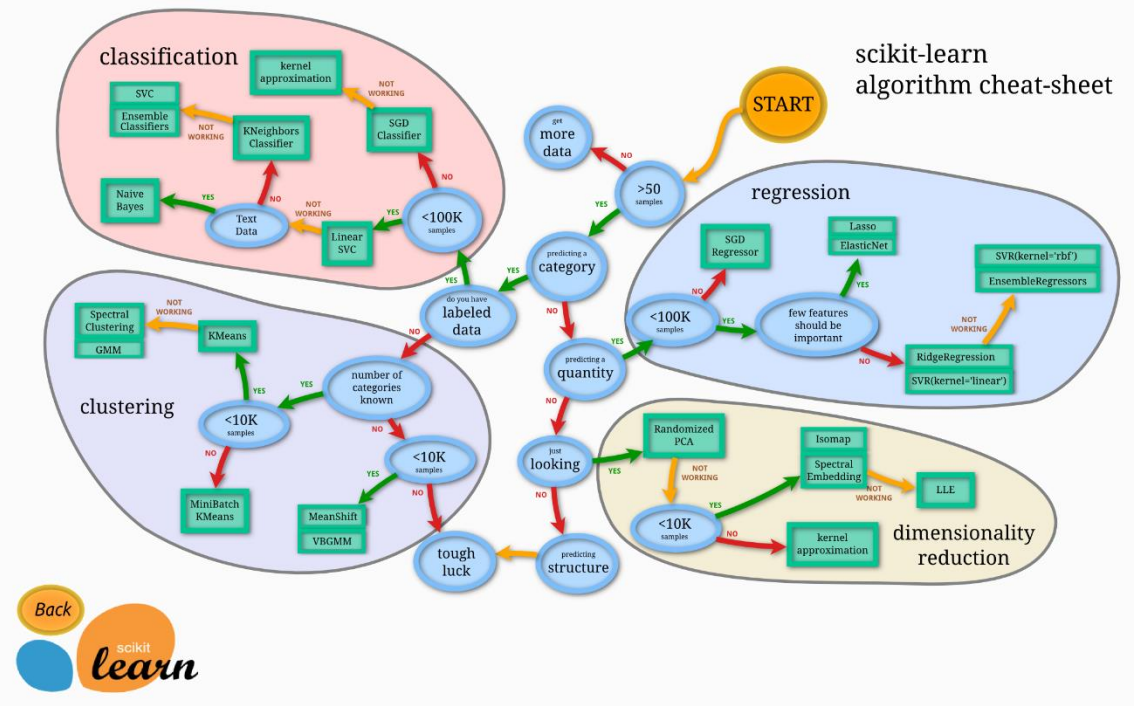

Figura 12. *Algoritmos de aprendizaje automático incluidos en scikit-learn. Fuente scikit-learn.org*

3.2 Procesamiento de datos

3.2.1 Captura y almacenamiento de *tweets*

Partimos de un grupo de trabajo, denominado “Twitter ingestion”, creado con Apache NiFi (versión 1.12.1) en un servidor departamental. Se puede acceder vía web a través de la VPN de la UAH. Contiene un subgrupo de trabajo encargado de extraer los datos (denominado “Twitter feed”), otro que se ocupa de escribir a base de datos (“Write to Twitter table”), y un tercero (“Logging”) que genera un log para cada uno de los otros dos [Figura 13].

Twitter feed consta de un procesador, etiquetado “Get vaccine-related *tweets*”, de tipo “GetTwitter”. Este procesador se conecta a la API de Twitter cada 15 segundos. Tiene configurado el parámetro “Twitter Endpoint” de tipo “Filter Endpoint”, para poder capturar *tweets* cuyo lenguaje sea “es” y contenga las palabras “vacuna” o “Vacuna”. Si la captura ha sido correcta, el *tweet* se coloca en la cola de un puerto de salida, etiquetado “Twitter data”. Dispone de otro puerto de salida, etiquetado “log”, para su conexión con el subgrupo “Logging”.

Desde este puerto se establece una conexión con el procesador etiquetado “Write to Twitter data”, de tipo “PutMongo”. Este procesador, para cada *tweet* en la cola de entrada, realiza una operación de tipo “insert” en una tabla MongoDB creada y configurada previamente en el mismo servidor departamental. Los *tweets* se escriben directamente en la tabla, tal como se capturan, sin realizar sobre ellos ninguna transformación en este paso.

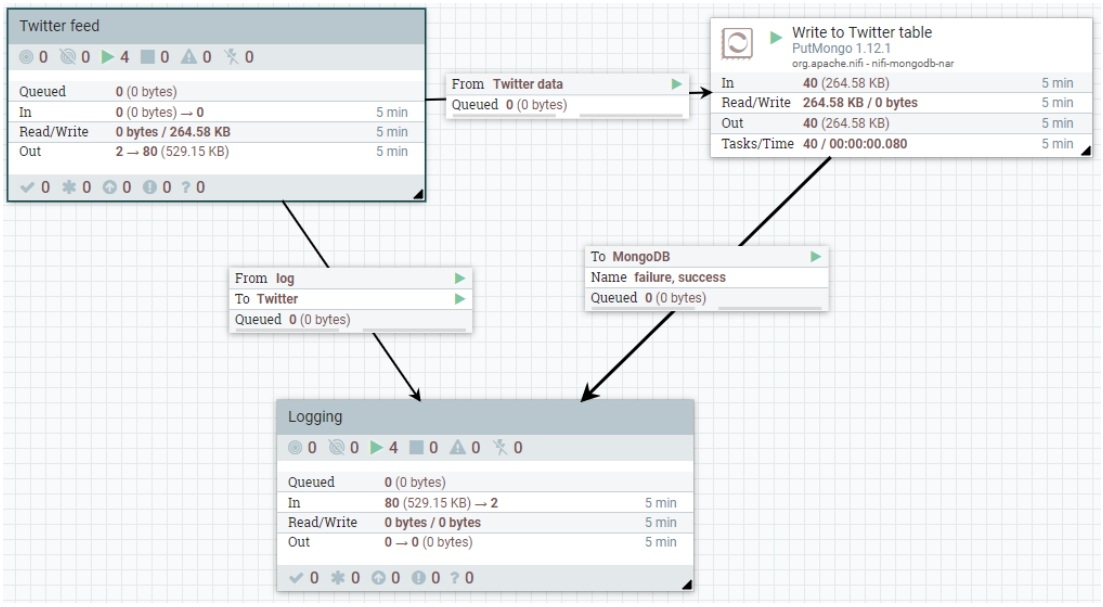

Figura 13. *Nifi: flujo de trabajo de extracción de tweets*

El subgrupo "Logging" consta de dos procesadores, ambos de tipo "LogMessage", etiquetados "Twitter extraction log" y "MongoDB write info", respectivamente. El primero de ellos tiene un puerto de entrada, etiquetado "Twitter", conectado con el puerto de salida "log", de Twitter feed. El segundo de ellos tiene también un puerto de entrada, "MongoDB", conectado directamente al procesador "Write to Twitter data".

El resultado de este proceso es una tabla en MongoDB, con formato JSON, que contiene todos los tweets capturados entre el 16 de diciembre de 2020 y el 30 de junio de 2021 [Tabla 1 , Tabla 2]. Estos tweets mantienen el formato original que proporciona la API. El número de campos (atributos) es variable en función de la tipología de cada tweet y los elementos que contiene (como menciones, imágenes, referencias a páginas web, etc.) y puede llegar a tener cerca de 150 campos.

Tabla 1. *Tweets capturados por meses*

Diciembre 2020 (desde el día 16)	81.834
Enero 2021	160.122
Febrero 2021	107.484
Marzo 2021	204.121
Abril 2021	164.798
Mayo 2021	203.295
Junio 2021	206.609
Total:	**1.128.263**

Tabla 2. *Tweets capturados por tipología*

	Total	**1.128.263**
Simples		212.941
Retweets		852.040
Quote tweets		196.269
Retweets y *quote tweets*		132.987

3.2.2 Limpieza de datos

Una vez almacenados los *tweets* en la base de datos, se han elegido los atributos de cada *tweet* que resultan útiles para su posterior tratamiento y clasificación, eliminando el resto para aligerar el tamaño del conjunto de datos y minimizar los tiempos de procesado. De esta forma, se ha creado una base de datos simplificada con un total de 7 u 8 atributos, según el caso [Figura 14, Figura 15]:

- "**_id**": objeto identificador que asigna automáticamente MongoDB.
- "**Identificador**": para *tweets* simples o *quote tweets* es el campo "id_str"[2] del *tweet* original. Para *retweets* es el campo "retweeted_status.id_str". Es de tipo texto.
- "**NumeroRetweets**": contenido del campo "retweeted_status.retweet_count" si se trata de un *retweet*. En el resto de casos es el contenido del campo "retweet_count". Es de tipo texto.
- "**tweet**": contenido del campo "text" para *tweets* simples y del campo "retweeted_status.text" para *retweets*. No existe para *quote tweets*. Tipo texto.
- "**tweetCitado**": sólo existe en *quote tweets*. Contenido del campo "quoted_status.text". Tipo texto.
- "**tweetNuevo**": sólo existe en *quote tweets*. Contenido del campo "text". Tipo texto.
- **"timestamp"**: contenido del campo "timestamp_ms", convertido a formato de fecha[3].
- "**sentimiento**": nuevo campo, inicialmente vacío, que contendrá el valor "-1", "0" o "1" para expresar sentimiento negativo, neutro o positivo respectivamente.

Se mantienen tres campos distintos para el texto de los *tweets*, en función de su tipología. Esto se hace así para facilitar la labor de clasificación manual. Posteriormente se unificarán en un único campo.

El paso siguiente consiste en eliminar todos los *tweets* duplicados que aparecen en la base de datos. Para ello se ha creado un programa en Python, utilizando las librerías "json" y "pandas", que busca los tweets que tengan el mismo atributo "Identificador", guarda el *tweet* que tenga el mayor número de *retweets* y elimina el resto [Código 1]. Se ha elegido este método para filtrar los *tweets* en vez de otros como guardar el *tweet* más reciente y eliminar los demás ya que de esta forma se almacena el *tweet* en el momento en el que tuvo mayor repercusión y se asegura que quede el reflejo de la opinión de los usuarios en ese momento, puesto que puede que algún usuario cambie de opinión o se arrepienta más adelante y elimine el *retweet*. El

[2] Existe el campo "**id**" con el mismo contenido, pero de tipo entero. No es aconsejable utilizarlo ya que, dependiendo del sistema, puede aparecer redondeado. Por este motivo se ha optado por elegir atributos tipo texto.

[3] El campo "**timestamp_ms**" de un *tweet* contiene el número de milisegundos transcurridos desde el 1 de enero de 1970.

resultado de este proceso es una base de datos, en JSON, con 534.700 *tweets* únicos. Se crea una copia de esta base de datos en formato de Microsoft Excel para que su posterior clasificación resulte más cómoda.


```
_id: ObjectId("608980f066c0e32f83cd846d")
created_at: "Tue Apr 27 18:37:57 +0000 2021"
id: 1387113806740983800
id_str: "1387113806740983808"
text: "RT @CiudadanoDoe: Más vacunación, más casos de #COVID—19. #España #Cov..."
source: "<a href="http://twitter.com/download/android" rel="nofollow">Twitter f..."
truncated: false
in_reply_to_status_id: null
in_reply_to_status_id_str: null
in_reply_to_user_id: null
in_reply_to_user_id_str: null
in_reply_to_screen_name: null
› user: Object
geo: null
coordinates: null
place: null
contributors: null
› retweeted_status: Object
quoted_status_id: 1379114381536026600
quoted_status_id_str: "1379114381536026626"
› quoted_status: Object
› quoted_status_permalink: Object
is_quote_status: true
quote_count: 0
reply_count: 0
retweet_count: 0
favorite_count: 0
› entities: Object
favorited: false
retweeted: false
filter_level: "low"
lang: "es"
timestamp_ms: "1619548677512"
```


Figura 14. *Ejemplo de tweet obtenido de la API de Twitter*


```
_id: ObjectId("612d25054a600d6d601ecef1")
Identificador: "1379215189678116867"
NumeroRetweets: "93"
tweetCitado: "Más vacunación, más casos de #COVID—19. #España #Covid #COVID19 #Vacun..."
tweetNuevo: "🚨 En Israel, Reino Unido, Alemania, Chile, Argentina, Hungría,... "CU..."
timestamp: "2021-04-27 18:37:57.512000"
sentimiento: null
```


Figura 15. *El mismo tweet de la* ***Figura 14*** *una vez simplificado*

```
#Si el tweet analizado es un retweet:
   elif str(df['quoted_status'][i])=='nan' and str(df['retweeted_status'][i])!='nan':
       contadorrt=contadorrt+1

       numeroRT=df['retweeted_status'][i]['retweet_count']
       identificadorRT=str(df['retweeted_status'][i]['id_str'])

       #Si el ID del tweet no aparece en el diccionario:
       if identificadorRT not in idUnicas:
           #Si el retweet está truncado:
           if df['retweeted_status'][i]['truncated']==True:
               textoEnteroRT=df['retweeted_status'][i]['extended_tweet']['full_text']
               idUnicas.setdefault(identificadorRT,[numeroRT,textoEnteroRT,fecha])
           #Si el retweet no está truncado:
           else:
               textoRT=df['retweeted_status'][i]['text']
               idUnicas.setdefault(identificadorRT,[numeroRT,textoRT,fecha])
       #Si el ID del tweet ya está en el diccionario:
       else:
           #Comparo el número de retweets de ambos tweets y actualizo al mayor valor:
           if idUnicas.get(identificadorRT)[0]<numeroRT:
               idUnicas[identificadorRT][0]=numeroRT
```

Código 1. *Eliminación de tweets duplicados, extracción del texto y mayor número de retweets*

Este programa lee el archivo JSON y lo almacena en un *dataframe,* utilizando la librería "pandas". Un *dataframe* es una estructura que permite almacenar datos en forma de tabla. A continuación, lo recorre buscando si se trata de un *tweet* simple, un *retweet*, un *quote tweet* o un *retweet* de un *quote tweet*, ya que tendrá que actuar sobre campos diferentes según el caso.

Si se trata de un *tweet* simple o de un *quote tweet* comprueba si se ha analizado previamente otro *tweet* con el mismo identificador, en caso contrario guarda en un diccionario su identificador como clave y asociado a esa clave guarda también su número de *retweets*, su texto y la fecha en la que se publicó. Si se trata de un *retweet* o un *retweet* de un *quote tweet* comprueba si se ha analizado anteriormente otro *tweet* con el mismo identificador. Si se da el caso, comprueba si el *tweet* que está analizando supera en número de *retweets* al que tiene almacenado y, si lo supera, lo actualiza al nuevo valor. Si no ha analizado anteriormente otro *tweet* con el mismo identificador sigue el mismo proceso que con los *tweets* simples y *quote tweets*.

3.3 Clasificación

3.3.1 Clasificación manual del conjunto de entrenamiento

Para realizar el proceso de clasificación manual se ha requerido la ayuda de varias personas que leyesen los *tweets* y los clasificasen en positivos, negativos o neutros en función de su sentimiento hacia la vacuna de la *covid – 19* [Figura 16 a Figura 18].

```
{
    "_id": {
        "$oid": "612d25094a600d6d601ff05d"
    },
    "Identificador": "1380965422271463429",
    "NumeroRetweets": "13",
    "tweet": ".@o_mediano  : \"La mejor vacuna es la vacuna puesta,\nno hay ninguna duda\" #L6Nvacúnate",
    "timestamp": "2021-04-10 20:50:42.460000",
    "sentimiento": 1
}
```


Figura 16. *Ejemplo de tweet con polaridad positiva*


```
{
    "_id": {
        "$oid": "612d25094a600d6d601fec49"
    },
    "Identificador": "1336355980473470982",
    "NumeroRetweets": "56",
    "tweet": "Vacuna O-BLI-GA-TO-RIA ????\nA mi no me vacuna nadie 😠😠",
    "timestamp": "2021-01-11 02:15:21.037000",
    "sentimiento": -1
}
```


Figura 17. *Ejemplo de tweet con polaridad negativa*


```
{
    "_id": {
        "$oid": "612d25094a600d6d601ff097"
    },
    "Identificador": "1369990597193400328",
    "NumeroRetweets": "0",
    "tweet": "Vacunas COVID 19: aclaraciones\nhttps://t.co/szzDqgcybQ https://t.co/lvMg728f1O",
    "timestamp": "2021-03-11 12:36:26.222000",
    "sentimiento": 0
}
```


Figura 18. *Ejemplo de tweet con polaridad neutra*

El resultado se muestra en la [Tabla 3]. Hay un claro desbalance entre los *tweets* con polaridad positiva y los que tienen polaridad negativa. Además, hay un gran número de *tweets* con polaridad neutra. Eso último se debe, por un lado, a que muchos de ellos no expresan opiniones (por ejemplo, anuncian una página web donde aparece información) y, por otro, a que otros muchos aprovechan para expresar opiniones sobre temas diferentes, habitualmente de carácter político. Aunque se ha realizado una labor de limpieza previa, intentando eliminar *tweets* no relacionados (por ejemplo, los que hacen referencia a “carne vacuna”), aún quedan muchos con polaridad neutra. Ambos aspectos (desbalance y gran número con polaridad neutra) han dificultado el entrenamiento del algoritmo de clasificación.

Tabla 3. *Resultado del proceso de clasificación manual*

***Tweets* clasificados**	**16.809**
Polaridad negativa	2.296
Polaridad neutra	5.750
Polaridad positiva	8.763

3.3.2 Preparación de los datos

Finalizado el proceso de clasificación manual, comienza el entrenamiento y prueba del clasificador. En primer lugar, se unifica el formato de todos los *tweets*. Para facilitar la labor de clasificación manual se habían mantenido tres campos de texto diferentes: "tweet" para *tweets* simples y *retweets* y "tweetCitado" y "tweetNuevo" para *quote tweets*. Concatenamos los dos últimos y el resultado lo volcamos en el campo "tweet", eliminando los dos campos originales. De esta forma, todos los *tweets* tienen el mismo formato, independientemente de su tipología. En este punto se preparan los textos que servirán de entrada para entrenar el algoritmo clasificador. Varias funciones de la librería "nltk" de Python se encargan de separar el texto del *tweet* en palabras, convertir todas las mayúsculas en minúsculas, eliminar palabras reservadas, emoticonos y otros símbolos y reducir cada palabra a su raíz. La función "limpiarTweets" llama a las funciones correspondientes en el orden adecuado [Código 2].

```
def limpiarTweets(cadena,encoding='UTF-8'):
    contador=0
    aux=[]
    for palabra in stemmizador(eliminarStopWords(tokenizador(cadena))):
        if contador!=0:
            aux.append(" ")
            aux.append(palabra)
        else:
            aux.append(palabra)
        contador=contador+1
    return ''.join(aux)
```

Código 2. *Función "limpiarTweets"*

En primer lugar, la función "tokenizador" separa todas las palabras del *tweet* y convierte todas sus letras en minúsculas [Código 3]. La salida de esta función se usa como entrada para la función "eliminarStopwords", que se encarga de borrar de los *tweets* cualquier palabra, símbolo o número que moleste o no aporte información a la hora de entrenar el clasificador [Código 4]:

- Hipervínculos.
- Nombres de usuario y menciones a *retweets*: palabras precedidas por "@" o "rt".
- Símbolo de *hashtag*: se elimina el símbolo "#" pero no la palabra que le sigue, ya que en algunos casos son muy valiosas (como, por ejemplo, en "#yonomevacuno").
- Listado de palabras de la librería "stopwords" de "nltk", con el parámetro "spanish", a la que se han añadido más palabras, signos y abreviaturas.
- Emoticonos.

```
def tokenizador(cadena,encoding='UTF-8'):
    tknzr = TweetTokenizer()
    return tknzr.tokenize(str(cadena).lower())
```

Código 3. *Función "tokenizador"*

```
def eliminarStopWords(cadena,encoding='UTF-8'):
    fraseFiltrada = []
    palabrasReservadas = set(stopwords.words('spanish'))
    palabrasReservadas.update([',','.',':',';','-','≤','≥','#','–','(',')','|','$','~','%',
                               '&','/','@','€','"','\'','‘','’','..','...','>',
                               '<','[',']','¨','{','}','*','+','^','_','Ç','ª','º','`','q',
                               'd','x','v'])
    for i in cadena:
        i=borrarEmoticonos(i)
        if i[0:4]!="http" and i[0:1]!='@' and i[0:2]!="rt" and i[0:1]!='1' and i[0:1]!='2' a
nd i[0:1]!='3' and  i[0:1]!='4' and i[0:1]!='5' and i[0:1]!='6' and i[0:1]!='7' and i[0:1]!=
'8' and i[0:1]!='9' and i[0:1]!='0':
            if i[0:1]=='#':
                i=i[1:]

            if i not in palabrasReservadas and i not in emoji.UNICODE_EMOJI['en']:
                    fraseFiltrada.append(i)

    return fraseFiltrada
```

Código 4. *Función "eliminarStopWords"*

Por último, las palabras del tweet se convierten a su raíz por medio de la función “stemmizador”, que hace uso de la función “SnowballStemmer” de “nltk” para idioma español [Código 5]. Tras este proceso, cada uno de los *tweets* está listo para poder entrenar el algoritmo de clasificación [Figura 19].

```
def stemmizador(cadena,encoding='UTF-8'):
    stemmer = SnowballStemmer('spanish')
    stemmed = []
    for item in cadena:
        stemmed.append(stemmer.stem(item))
    return stemmed
```

Código 5. *Función "stemmizador"*

gent tan trastorn encierr desinform medic bombard medi mensaj terror asimil import si vacun mat unic qui termin vez asi muert import

Figura 19. *Ejemplo de tweet tras el proceso de preparación*

3.3.3 Entrenamiento del algoritmo de clasificación

Con los *tweets* ya clasificados y preparados comienza el entrenamiento del clasificador. Para entrenarlo se emplean tres archivos de Microsoft Excel, con los *tweets* clasificados manualmente con polaridad positiva, neutra y negativa respectivamente. Se leen, se concatenan y se almacenan en un *dataframe*. Después el texto de los *tweets* se vectoriza mediante la función "TfidfVectorizer" de la librería "sklearn". Esta función calcula la importancia de aparición de las palabras en los *tweets* y su frecuencia, con el objetivo de filtrar las palabras menos relevantes. Con los *tweets* vectorizados, sus respectivos sentimientos asignados y la función "train_test_split" de la librería "sklearn", se separan los *tweets* que van a ser utilizados para entrenar el algoritmo de los que van a ser utilizados para testearlo [Código 6].

```
vectorizer = TfidfVectorizer(sublinear_tf=True,max_df=1.0)

vectorizado = vectorizer.fit_transform(dfUnion['texto'].values.astype('U'))

X = vectorizado
y = dfUnion['sentimiento']

#Separa los tweets entre los que van a ser usados para entrenar y los que van a ser usados p
ara testeo
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.119,random_state=67)
#Crea el clasificador
clf = svm.SVC(kernel='rbf', C=10,random_state=3).fit(X_train, y_train)
```

Código 6. *Vectorización del conjunto de datos y entrenamiento del clasificador*

Tras probar diversos algoritmos de clasificación y ajustar sus parámetros, el finalmente empleado es el algoritmo "SVC" de la librería "svm" de "sklearn". Este algoritmo, se ajusta a los datos proporcionados devolviendo un hiperplano que divide los datos. Una vez entrenado y testeado con los 16.809 *tweets* clasificados manualmente, arroja una precisión del 73.56%, y queda preparado para clasificar los *tweets* restantes. Para ello se somete al conjunto de datos sin clasificar al mismo proceso de preparación, utilizando de nuevo las funciones "limpiarTweets" y "TfidfVectorizer".

3.4 Recopilación de hitos temporales

En paralelo con la captura de datos de Twitter se ha realizado una recopilación de noticias y otros hitos relevantes para el proceso de vacunación, abarcando el mismo intervalo temporal, esto es, de mediados de diciembre de 2020 a finales de junio de 2021. Se trata de un proceso manual de búsqueda web y almacenamiento en una base de datos, registrando:

- **Fecha**.
- **Sentimiento**: si se considera que el hito afecta de forma positiva, negativa o neutra a la opinión pública sobre el proceso de vacunación.
- **Resumen**: breve descripción del hito.

- **Texto**: titular del hito en el medio de comunicación.
- **Enlace**: vínculo web al medio donde aparece publicado.

Esta base de datos se ha guardado tanto en formato JSON como en formato de Microsoft Excel [Figura 20].


```
_id: ObjectId("6135c9cc688bb67e3822f333")
Id: "6 "
Fecha: "2020/12/27"
Sentimiento: 1
Resumen: "Comienza la campaña de vacunación en España "
Texto: ""
Enlace: ""
```


Figura 20. *Ejemplo de hito temporal*

4 Resultados

4.1 Análisis de sentimiento

Con el conjunto de *tweets* clasificados de forma manual se han entrenado los siguientes algoritmos de clasificación, utilizando la librería "scikit-learn" de Python:

- SVM
  - SVC
  - LinearSVC
- Naïve Bayes
  - MultinomialNB
  - BernouilliNB
  - ComplementNB
  - GaussianNB

En la [Tabla 4] se muestra un resumen comparativo de resultados. En el Anexo 1 se definen las métricas que se utilizan habitualmente para evaluar el rendimiento de los algoritmos, se describe cada uno de ellos, se presentan las matrices de confusión obtenidas del conjunto de prueba y las puntuaciones alcanzadas en cada caso. Como referencia, en la [Figura 21] se muestra el resultado obtenido por los participantes en el TASS 2020. En este TFG se han alcanzado puntuaciones de un nivel similar, y en todo caso superiores a las que alcanzan las herramientas comerciales presentadas en el Capítulo 2, [Figura 10], lo que corrobora la gran calidad de las librerías "NLTK" y "scikit-learn" de Python.

Tabla 4. *Comparativa del resultado del entrenamiento de los algoritmos utilizados*

Algoritmo	*Accuracy score*	*F1-score*	*Precision*	*Recall*
SVC	0.74	0.75	0.77	0.74
LinearSVC	0.72	0.73	0.75	0.72
MultinomialNB	0.70	0.72	0.77	0.70
BernouilliNB	0.69	0.70	0.71	0.69
ComplementNB	0.69	0.69	0.71	0.69
GaussianNB	0.42	0.40	0.46	0.42

Team	Set	F1	Precision	Recall
ELiRF-UPV	ES	0.671	0.673	0.670
Palomino-Ochoa	ES	0.665	0.665	0.664
UMUTeam	ES	0.503	0.561	0.456

Figura 21. *Resultado del TASS 2020 para polaridad de tres niveles en idioma español*

Los algoritmos de máquinas de vector soporte presentan un rendimiento superior a los de tipo Naïve Bayes y, dentro de los primeros, el algoritmo SVC supera ligeramente al SVC lineal. Entre los de la familia Naïve Bayes todos arrojan un resultado muy similar, excepto el gaussiano, cuyo resultado no es aceptable. "MultinomialNB" obtiene mejor resultado que "ComplementNB", a pesar de que el segundo es una variante del primero para conjuntos de datos no balanceados.

Todos ellos clasifican mucho mejor la polaridad positiva que la neutra o la negativa y las diferencias son más acusadas en precisión (positivos identificados por el clasificador respecto a los realmente etiquetados como positivos por un humano) que en el resto de las métricas. Por ejemplo, para SVC, esa diferencia llega casi a los 40 puntos porcentuales [Tabla 5]. Consideramos que esas diferencias se deben, principalmente, a dos factores; por un lado, por el desbalance en el número de registros entre clases. Hay muchos más *tweets* con polaridad positiva que con polaridad negativa (relación 1:6, aproximadamente), con un número intermedio de neutros (relación 1:3, aproximadamente). Eso hace que los *tweets* positivos o neutros que se acaban clasificando como negativos, aunque sean un porcentaje pequeño de los positivos y neutros respectivamente, suponen un porcentaje elevado respecto a los negativos, perjudicando su puntuación.

Tabla 5. *Puntuaciones obtenidas por polaridad por el clasificador SVC*

Polaridad	***F1-score***	***Precision***	***Recall***
Negativa	0.63	0.52	0.79
Neutra	0.56	0.51	0.62
Positiva	0.82	0.89	0.76

El otro factor se encuentra en la propia naturaleza de los *tweets* neutros. Muchos de ellos incluyen un texto sin ningún tipo de emoción, y les podríamos denominar "neutros reales". Pero otros muchos sí expresan un sentimiento positivo o negativo, aunque ese sentimiento no guarda relación con la vacuna ("falsos neutros"). Esos "falsos neutros" se clasifican como neutros por un humano, pero los algoritmos van a clasificar muchos de ellos como positivos o negativos. Encontrar la forma de eliminar los "falsos neutros" queda fuera del alcance del presente TFG.

Aunque todos los *tweets* capturados hacen referencia a la vacuna o al proceso de vacunación, los argumentos utilizados en los comentarios con polaridad positiva difieren significativamente de los utilizados por usuarios con un sentimiento de rechazo hacia la vacuna.

Los usuarios a favor del proceso de vacunación hablan de salud, de las diferentes vacunas que existen, del calendario para la administración de las diferentes dosis o de los grupos de edad a los que se está administrando la vacuna [Figura 22]. Por otro lado, los comentarios de *tweets* con polaridad negativa mencionan el virus, las muertes, los efectos secundarios o una vacuna en concreto que ha sido el foco de atención por sus posibles efectos secundarios [Figura 23].

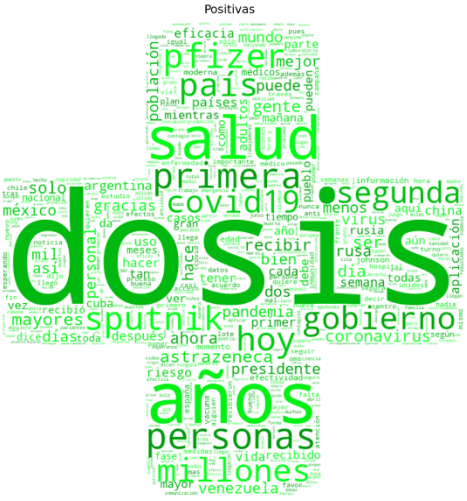

Figura 22. *Nube de palabras para polaridad positiva*

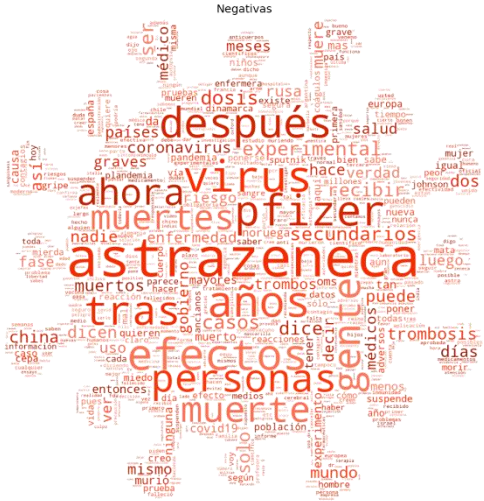

Figura 23. *Nube de palabras para polaridad negativa*

En cuanto a los comentarios que podemos encontrar en los tweets clasificados con polaridad neutra, por un lado se encuentran los que aprovechan cualquier conversación para hablar de otros temas, destacando los de contenido político. Por otro lado, se puede observar una mezcla de las palabras que aparecen en los comentarios con polaridad positiva y con polaridad negativa, con mayor frecuencia de los primeros, pues son los más abundantes [Figura 24].

Figura 24. *Nube de palabras para polaridad neutra*

4.2 Evolución del sentimiento en el tiempo

El mes de mayo de 2021, salvo por el anuncio en sus primeros días de los estudios sobre la vacuna en menores de 12 años, ha sido un mes sin noticias con un fuerte impacto mediático, ni positivo ni negativo. Por ese motivo, se ha utilizado como "mes referencia" o "mes de control" para las observaciones. En el Anexo 2 se recopilan gráficas mensuales con los *tweets* y *retweets* para cada polaridad y también combinando polaridad positiva con negativa.

El tema elegido para este TFG, la vacunación, ha estado presente en los medios de comunicación durante todo el periodo de estudio. Incluso en ausencia de noticias relevantes, a diario se comentaba el avance del proceso de vacunación en cada Comunidad Autónoma, las dosis recibidas de cada una de las vacunas o el grupo de edad al que correspondía vacunarse cada semana. Y eso ha tenido un reflejo también en las redes sociales. Podemos establecer una línea base de *tweets* de cada polaridad registrados durante todo el periodo, debidos al hecho de que ha sido un tema de constante actualidad [Tabla 6]. Esto no es habitual en Twitter, donde el eco de las noticias es muy efímero.

Tabla 6. *Línea base aproximada de la media de tweets diarios*

Polaridad	***tweets únicos***	***retweets***	***Proporción***
Negativa	≈ 200	≈ 2.5K	x 12
Neutra	≈ 800 → ≈ 600	≈ 30K → ≈ 10K	x 35 → x 15
Positiva	≈ 2K	≈ 70K	x 35

Ese nivel de referencia se ha mantenido estable para las polaridades positiva y negativa, tanto para el número de *tweets* únicos registrados como para el número de *retweets*. Durante ese periodo, la realidad social ha evolucionado hacia una proporción cada vez mayor de personas vacunadas y una constatación creciente de ausencia de efectos secundarios y de bajada significativa en la proporción de ingresos hospitalarios y de muertes por la enfermedad. Podría parecer lógico esperar que en la red social se vería un incremento paulatino de mensajes con polaridad positiva, acompañado de un descenso proporcional en la polaridad negativa. Sin embargo, esto no ha sucedido y lo que este estudio refleja es que la opinión de los usuarios se ha mantenido constante durante todo el periodo.

Para la polaridad neutra, aunque el número base de *tweets* únicos se ha mantenido (con ligera tendencia descendente) la línea base del número de *retweets* al final del periodo es una tercera parte de los existentes al principio del periodo. Esto parece confirmar lo que se comentaba anteriormente respecto a los "falsos neutros", *tweets* que aprovechan el tema de conversación para opinar sobre otros temas, habitualmente políticos. Con el tiempo el número de *retweets* de esos comentarios disminuye significativamente.

También es destacable el hecho de que el número base de mensajes con polaridad positiva es 10 veces superior al número base de mensajes con polaridad negativa. Y al calcular la proporción de *retweets* respecto al número de *tweets* únicos también podemos observar una actividad en la red social mucho mayor entre los usuarios que publican mensajes con polaridad positiva respecto al resto, triplicando a la proporción correspondiente a la polaridad negativa.

Todo ello pone de relieve que los usuarios que publican mensajes con polaridad negativa son mucho menos activos que el resto.

Otro efecto observado al analizar la distribución de los *tweets* a lo largo del tiempo, y que se aprecia claramente en el mes de control, es el efecto "fin de semana", exclusivamente para los mensajes con polaridad positiva. El número de *tweets* únicos con polaridad positiva cae drásticamente durante el fin de semana (en ausencia de hitos mediáticos que aumenten la actividad, como se describe más adelante). Esto no ocurre para el resto de polaridades ni tampoco para los *retweets* con polaridad positiva [Figura 25] [Figura 26].

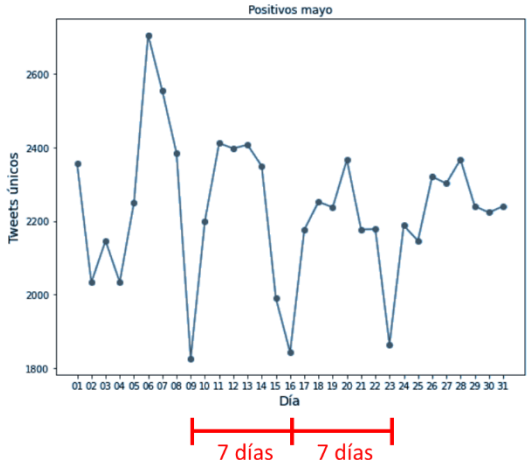

Figura 25. *Efecto "fin de semana" para tweets únicos con polaridad positiva*

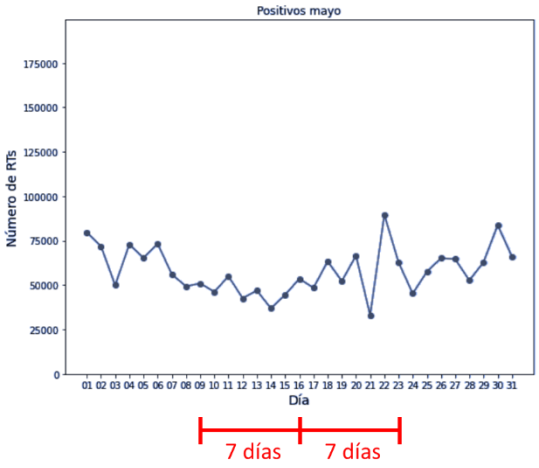

Figura 26. *Retweets con polaridad positiva sin efecto "fin de semana"*

4.3 Influencia de hitos mediáticos

Como se ha indicado anteriormente, en ausencia de noticias relevantes hay un "ruido de fondo" de *tweets* tanto de polaridad positiva como negativa que no varía con el tiempo. Pues bien, cuando las noticias que van apareciendo, aún sin poder ser calificadas como un hito de especial impacto, todas van orientadas hacia la misma polaridad, aunque los *tweets* de esa polaridad se mantienen alrededor de su línea base (con subidas puntuales para volver enseguida a la línea base), los *retweets* van creciendo paulatinamente. De igual forma, los *tweets* y *retweets* de polaridad contraria se mantienen en su línea base. Esto se observa con claridad en el mes de enero de 2021. No hubo noticias positivas con un impacto importante, pero se sucedieron varias en cadena que fueron realimentando las conversaciones en redes sociales: se comienza a administrar la segunda dosis a mayores, posteriormente a sanitarios, aparece un estudio mostrando la eficacia de las vacunas, etc [Figura 27].

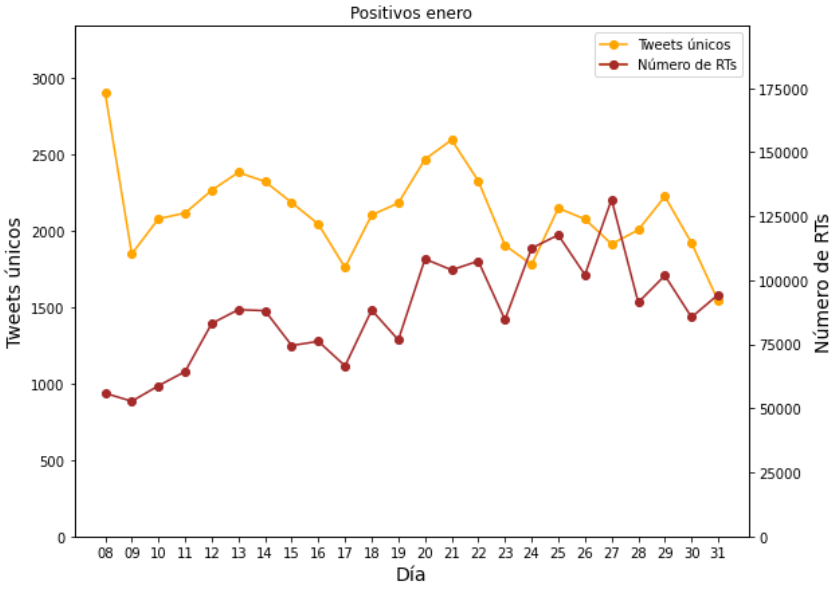

Figura 27. *Noticias encadenadas de una misma polaridad, pero sin hitos con impacto*

Este mismo mes se produjeron dos hitos con polaridad negativa que nos permiten diferenciar dos comportamientos muy diferentes entre los usuarios de la red social: "efecto impacto" y "efecto viral". El día 25 se hizo viral un comentario que hablaba sobre Bill Gates y su posible interés comercial en la administración de la vacuna. Eso provocó un aumento significativo de los comentarios con polaridad negativa, tanto de *tweets* como de *retweets.* Sin embargo, no se observa un descenso en el número de *tweets* con polaridad positiva (que en esos días oscilaban alrededor de su línea base) ni en el número de *retweets* (que continuaban su senda de crecimiento por la acumulación de pequeñas noticias positivas) [Figura 28] [Figura 29].

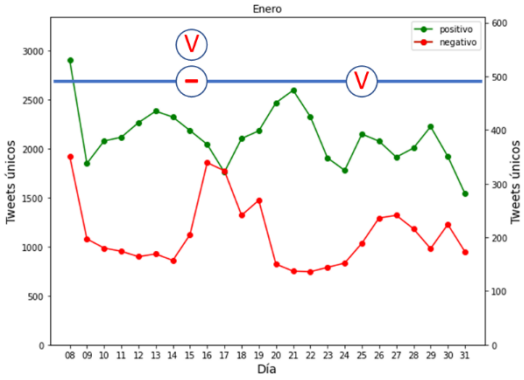

Figura 28. *Efecto del impacto y la viralidad en los tweets únicos*

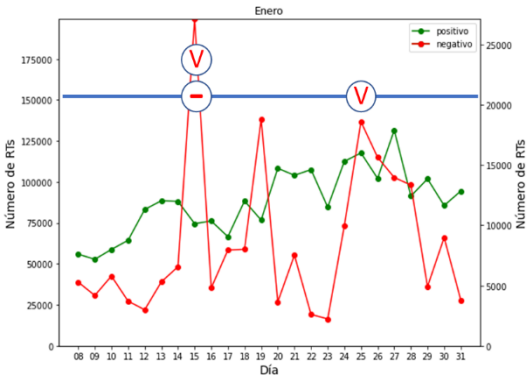

Figura 29. *Efecto del impacto y la viralidad en los retweets*

El día 15 se conoció que había fallecido un número elevado de personas en una residencia de la tercera edad de un país europeo. Esa noticia sí tuvo impacto. Aumentaron de forma importante los *tweets* con polaridad negativa, que llevaban varios días instalados en su línea base, para volver a ella varios días más tarde. Al mismo tiempo, descendieron los *tweets* y los

retweets con polaridad positiva, para volver también a su estado anterior unos días más tarde. Esta noticia, además, se convirtió en viral, con un número muy importante de *retweets* con polaridad negativa, que rebotaron los días siguientes, realimentados con más noticias de muertes, generando otros dos picos de intensidad decreciente. Como se puede observar, los *tweets* y *retweets* positivos se guían por el efecto impacto, reaccionando de forma proporcional al incremento de *tweets* negativos, pero no reaccionan al efecto viral, representada por los *retweets* negativos.

Los hitos mediáticos provocan en determinadas ocasiones un "efecto debate", en cuyo caso se produce un aumento de *tweets* de ambas polaridades. Los hitos que los provocan pueden considerarse neutros pues, o bien no tienen una polaridad clara, o bien se trata de estudios que presentan tanto los aspectos positivos como negativos de la vacunación. Para ilustrarlo podemos recurrir al mes de junio [Figura 30]. El día 2, tras la reunión interterritorial entre el Ministerio de Salud y las Comunidades Autónomas, se inició el debate acerca de qué vacuna debería administrarse para la segunda dosis cuando la primera había sido con Astrazeneca. El día 11 se aviva el debate con la noticia de la vacunación a la Selección Española de Fútbol y con el adelanto de la vacunación ante el avance de la variante delta.

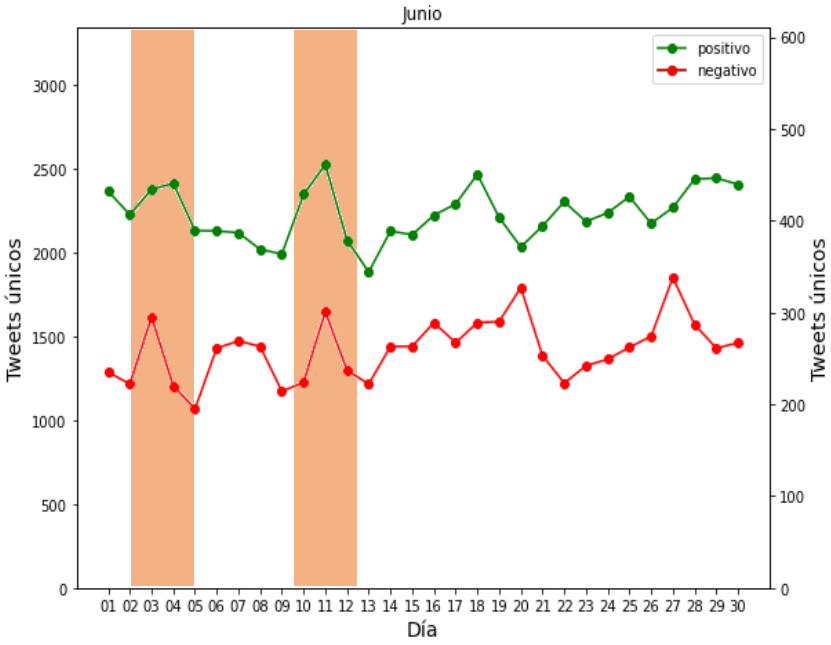

Figura 30. *Ejemplo de hitos mediáticos que generan un efecto debate*

Como puede verse en la figura, en este mes también se han producido hitos mediáticos que han provocado el efecto impacto. Por ejemplo, el día 20 se difundió la noticia de que India estaba sufriendo una segunda ola devastadora y el 25 se conoció un macrobrote de contagios por un viaje de fin de curso. Como en ocasiones anteriores, los *retweets* apenas si acusan el efecto debate y se explican mejor por el efecto viral. De hecho, a partir del día 14 se difundieron mensajes equiparando la vacuna a la eutanasia e informando de muertes tras recibir la vacuna. Quizá como respuesta, a partir del día 19 empezaron a aparecer mensajes a favor de la vacunación. La confrontación pareció consolidarse a partir del día 24 con mensajes que hablaban de que las vacunas contienen un chip, y mensajes de contrapartida ironizando con el tema

[Figura 31]. Estos movimientos en el número de *retweets* no parecen tener reflejo en los *tweets* únicos de esas mismas fechas.

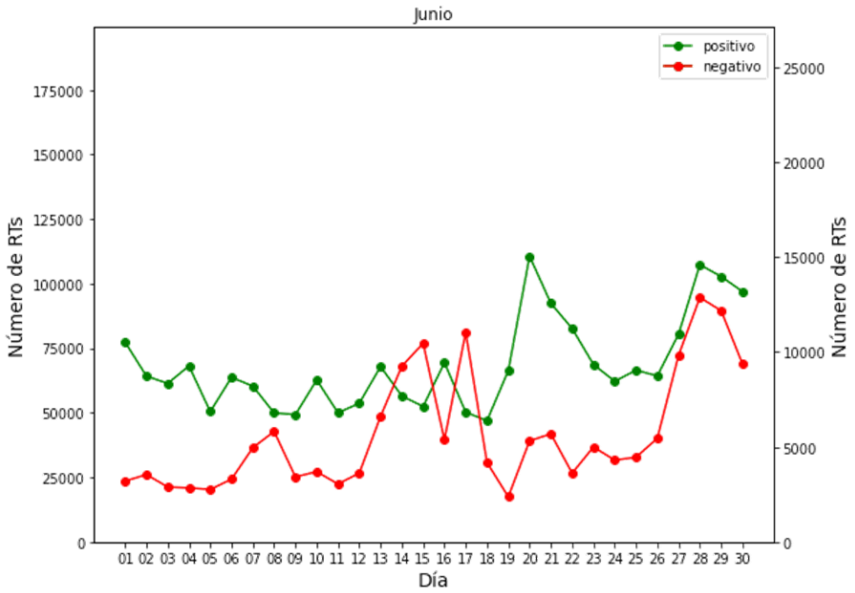

Figura 31. *Retweets y el efecto viral*

Solamente se han observado dos hitos mediáticos que han tenido un impacto tanto en el número de *tweets* como en el número de *retweets* de forma simultánea, y quizá hayan sido los dos hitos de mayor relevancia ocurridos durante el periodo de estudio. El primero de ellos con impacto positivo, con el anuncio y posterior inicio de la vacunación a finales de diciembre de 2020. El segundo de ellos con impacto negativo, cuando se conocieron los posibles efectos adversos de la vacuna de Astrazeneca y varios países en cadena decidieron parar la administración de dicha vacuna, a mediados de marzo de 2021.

Para finalizar, un comentario acerca de los *tweets* con polaridad neutra. No se ha podido identificar una relación clara del comportamiento en el número de estos mensajes y los hitos mediáticos. Como se comentó anteriormente, los *tweets* con polaridad neutra también presentan una línea base, que se ha ido reduciendo durante el periodo analizado, y el número de *retweets* se ha reducido significativamente. Pues bien, estos mensajes, o bien se mantienen en su línea base a pesar de la existencia de hitos, o bien siguen con bastante precisión el movimiento, bien de los mensajes con polaridad positiva, bien a los de polaridad negativa. Consideramos que esto último se debe a los errores de clasificación del algoritmo, más acusado para la polaridad neutra.

5 Conclusiones y trabajo futuro

5.1 Conclusiones

1. Python en general, y sus librerías pandas, NLTK y scikit-learn en particular, son **herramientas de primer nivel para el análisis de sentimiento**, logrando resultados para idioma español comparables a los encontrados en la literatura y superiores a los de plataformas comerciales.
2. De los algoritmos tradicionales utilizados, los de la familia SVM superan a los de tipo Naíve Bayes, logrando **SVC** el mejor resultado, clasificando mejor la polaridad positiva que el resto.
3. La **actividad** de los usuarios con **comentarios positivos** es **mucho mayor** que el resto, y el tipo de vocabulario empleado muy diferente.
4. El impacto mediático constante del tema de estudio durante el periodo observado provoca que exista una **"línea base"** de comentarios permanente, para todas las polaridades. Este volumen se mantiene constante en el tiempo por lo que, en estas condiciones, **no se detecta cambio de opinión en el tiempo**, aunque sí un **descenso en el reenvío de mensajes con polaridad neutra**.
5. Por el contrario, se ha detectado que los hitos mediáticos provocan alteraciones en el volumen de comentarios y en su evolución, en mayor medida para *tweets* únicos. En este sentido, se han identificado varios "efectos", o reacciones:
    a. Efecto **fin de semana**, exclusivamente en *tweets* únicos con polaridad positiva.
    b. Efecto **crecimiento por acumulación** únicamente en *retweets* de la polaridad implicada.
    c. Efecto **impacto** aumentando la actividad de *tweets* únicos con la misma polaridad que el hito mediático y disminuyendo la actividad de *tweets* únicos de la polaridad contraria. Sin impacto de importancia en *retweets*.
    d. Efecto **gran impacto**, igual que el efecto impacto pero implicando también a los *retweets* de ambas polaridades.
    e. Efecto **debate**, que provoca un aumento de actividad en *tweets* únicos de ambas polaridades y sin impacto relevante en *retweets*.
    f. Efecto **viral**, con aumento de actividad de *retweets* de una determinada polaridad, a veces también de la contraria y sin impacto de importancia en *tweets* únicos. No siempre está directamente relacionado con un hito mediático.
6. En este estudio **no se ha encontrado una relación** clara entre los hitos mediáticos y el comportamiento de los *tweets* con **polaridad neutra**.

5.2 Mejoras y trabajo futuro

Este TFG abre la posibilidad de introducir mejoras a su contenido, continuar líneas de trabajo o abrir otras nuevas. Entre todas ellas podemos citar:

1. Incorporar los emoticonos al análisis de sentimiento.
2. Utilizar técnicas de aprendizaje profundo y técnicas basadas en diccionarios para comparar los resultados.
3. Mejorar el proceso de prueba de algoritmos y de búsqueda de parámetros óptimos con herramientas tipo MLOps.
4. Mejorar el tratamiento de los tweets con polaridad neutra, tanto en la fase de limpieza y preparación de los datos como en la fase de procesamiento, diferenciando entre neutros reales y neutros que contienen sentencias positivas y negativas, es posible mejorar el rendimiento [98].
5. Automatizar la identificación, captura y almacenamiento de hitos mediáticos.
6. Automatizar la búsqueda de relaciones entre tráfico en redes sociales e hitos mediáticos.

Bibliografía

[1] R. H. Thaler, “From cashews to nudges: The evolution of behavioral economics,” *American Economic Review*, vol. 108, no. 6. pp. 1265–1287, 2018. doi: 10.1257/aer.108.6.1265.

[2] J. Luoto *et al.*, “Behavioral Economics Guidelines with Applications for Health Interventions,” no. May, 2016, Accessed: Aug. 05, 2021. [Online]. Available: https://publications.iadb.org/publications/english/document/Behavioral-Economics-Guidelines-with-Applications-for-Health-Interventions.pdf

[3] “Digital 2021: Global Overview Report — DataReportal – Global Digital Insights.” https://datareportal.com/reports/digital-2021-global-overview-report (accessed Aug. 29, 2021).

[4] “Claves del Estudio Anual de RRSS 2021 IAB by ELOGIA.” https://blog.elogia.net/claves-del-estudio-anual-de-rrss-2021-iab-by-elogia#populares (accessed Aug. 29, 2021).

[5] “▷ Historia de las redes sociales. 【Evolución + cronología】.” https://evaortiz.es/historia-redes-sociales (accessed Aug. 29, 2021).

[6] D. Westerman, P. R. Spence, and B. van der Heide, “Social Media as Information Source: Recency of Updates and Credibility of Information,” *Journal of Computer-Mediated Communication*, vol. 19, no. 2, pp. 171–183, Jan. 2014, doi: 10.1111/JCC4.12041.

[7] ZubiagaArkaitz, AkerAhmet, BontchevaKalina, LiakataMaria, and ProcterRob, “Detection and Resolution of Rumours in Social Media,” *ACM Computing Surveys (CSUR)*, vol. 51, no. 2, p. 32, Feb. 2018, doi: 10.1145/3161603.

[8] D. Antonakaki, P. Fragopoulou, and S. Ioannidis, “A survey of Twitter research: Data model, graph structure, sentiment analysis and attacks,” *Expert Systems with Applications*, vol. 164, p. 114006, Feb. 2021, doi: 10.1016/J.ESWA.2020.114006.

[9] A. Hernández-Fuentes and A. Monnier, “Twitter as a Source of Information? Practices of Journalists Working for the French National Press,” *https://doi.org/10.1080/17512786.2020.1824585*, 2020, doi: 10.1080/17512786.2020.1824585.

[10] B. M. Fowler, “Stealing thunder and filling the silence: Twitter as a primary channel of police crisis communication,” *Public Relations Review*, vol. 43, no. 4, pp. 718–728, Nov. 2017, doi: 10.1016/J.PUBREV.2017.04.007.

[11] B. Wang and J. Zhuang, “Crisis information distribution on Twitter: a content analysis of tweets during Hurricane Sandy,” *Natural Hazards 2017 89:1*, vol. 89, no. 1, pp. 161–181, Jun. 2017, doi: 10.1007/S11069-017-2960-X.

[12] Y. Wang, H. Hao, and L. S. Platt, “Examining risk and crisis communications of government agencies and stakeholders during early-stages of COVID-19 on Twitter,” *Computers in Human Behavior*, vol. 114, p. 106568, Jan. 2021, doi: 10.1016/J.CHB.2020.106568.

[13] “The Infrastructure Behind Twitter: Scale.” https://blog.twitter.com/engineering/en_us/topics/infrastructure/2017/the-infrastructure-behind-twitter-scale (accessed Sep. 05, 2021).

[14] “‘https://developer.twitter.com/en/pricing.’”

[15] “Using Twitter as a data source: an overview of social media research tools (2019) | Impact of Social Sciences.” https://blogs.lse.ac.uk/impactofsocialsciences/2019/06/18/using-twitter-as-a-data-source-an-overview-of-social-media-research-tools-2019/ (accessed Sep. 05, 2021).

[16] “GitHub - Mottl/GetOldTweets3: A Python 3 library and a corresponding command line utility for accessing old tweets.” https://github.com/Mottl/GetOldTweets3 (accessed Sep. 05, 2021).

[17] “Netlytic.org – a cloud-based text and social networks analyzer.” https://netlytic.org/home/ (accessed Sep. 05, 2021).

[18] “NodeXL | Your Social Network Analysis Tool for Social Media.” https://www.smrfoundation.org/nodexl/ (accessed Sep. 05, 2021).

[19] “SocioViz, a free Social Network Analysis tool for Twitter.” https://socioviz.net/SNA/eu/sna/login.jsp (accessed Sep. 05, 2021).

[20] Doug Laney, *“3-D Data Management: Controlling Data Volume, Velocity and Variety.”* Application Delivery Strategies by META Group Research Note, February 6, 2001, p. 949. [Online]. Available: http://goo.gl/Bo3GS

[21] “Big Data Reduction 1: Descriptive Analytics - Lithium Community.” https://community.khoros.com/t5/Khoros-Communities-Blog/Big-Data-Reduction-1-Descriptive-Analytics/ba-p/77766 (accessed Aug. 31, 2021).

[22] S. Stieglitz, M. Mirbabaie, B. Ross, and C. Neuberger, “Social media analytics – Challenges in topic discovery, data collection, and data preparation,” *International Journal of Information Management*, vol. 39, pp. 156–168, Apr. 2018, doi: 10.1016/J.IJINFOMGT.2017.12.002.

[23] “Apache NiFi.” https://nifi.apache.org/ (accessed Sep. 05, 2021).

[24] “StreamSets.” https://streamsets.com/ (accessed Sep. 05, 2021).

[25] “Apache Airflow.” https://airflow.apache.org/ (accessed Sep. 05, 2021).

[26] “Amazon Data Pipeline: servicio ETL administrado (Amazon Web Services).” https://aws.amazon.com/es/datapipeline/ (accessed Sep. 05, 2021).

[27] “AWS Glue: servicio ETL administrado (Amazon Web Services).” https://aws.amazon.com/es/glue/?whats-new-cards.sort-by=item.additionalFields.postDateTime&whats-new-cards.sort-order=desc (accessed Sep. 05, 2021).

[28] “Talend.” https://www.talend.com/products/data-fabric/ (accessed Sep. 05, 2021).

[29] “PowerCenter - Enterprise Data Integration Tool | Informatica.” https://www.informatica.com/products/data-integration/powercenter.html (accessed Sep. 05, 2021).

[30] “Ab Initio.” https://www.abinitio.com/es/ (accessed Sep. 05, 2021).

[31] “Pentaho Enterprise Edition | Hitachi Vantara.” https://www.hitachivantara.com/en-us/products/data-management-analytics/pentaho.html (accessed Sep. 05, 2021).

[32] “Data Factory: servicio de integración de datos | Microsoft Azure.” https://azure.microsoft.com/es-es/services/data-factory/ (accessed Sep. 05, 2021).

[33] “JSON.” https://www.json.org/json-es.html (accessed Sep. 05, 2021).

[34] Z. Parker, S. Poe, and S. v Vrbsky, “Comparing NoSQL MongoDB to an SQL DB,” *Proceedings of the 51st ACM Southeast Conference on - ACMSE ’13*, 2013, doi: 10.1145/2498328.

[35] “DB-Engines Ranking - popularity ranking of database management systems.” https://db-engines.com/en/ranking (accessed Sep. 05, 2021).

[36] “XML Technology - W3C.” https://www.w3.org/standards/xml/ (accessed Sep. 05, 2021).

[37] “Apache Cassandra | Apache Cassandra Documentation.” https://cassandra.apache.org/_/index.html (accessed Sep. 05, 2021).

[38] “AWS | Servicio de base de datos gestionada NoSQL (DynamoDB).” https://aws.amazon.com/es/dynamodb/ (accessed Sep. 05, 2021).

[39] “Apache CouchDB.” https://couchdb.apache.org/ (accessed Sep. 05, 2021).

[40] “RethinkDB: the open-source database for the realtime web.” https://rethinkdb.com/ (accessed Sep. 05, 2021).

[41] “OrientDB Community Edition.” https://orientdb.org/ (accessed Sep. 05, 2021).

[42] “PostgreSQL: The world’s most advanced open source database.” https://www.postgresql.org/ (accessed Sep. 05, 2021).

[43] E. Ahmed *et al.*, “The role of big data analytics in Internet of Things,” *Computer Networks*, vol. 129, pp. 459–471, Dec. 2017, doi: 10.1016/J.COMNET.2017.06.013.

[44] P. Rizwan, K. Suresh, and M. Rajasekhara Babu, “Real-time smart traffic management system for smart cities by using Internet of Things and big data,” *Proceedings of IEEE International Conference on Emerging Technological Trends in Computing,*

Communications and Electrical Engineering, ICETT 2016, Mar. 2017, doi: 10.1109/ICETT.2016.7873660.

[45] H. Wang, O. L. Osen, G. Li, W. Li, H. N. Dai, and W. Zeng, "Big data and industrial Internet of Things for the maritime industry in Northwestern Norway," *IEEE Region 10 Annual International Conference, Proceedings/TENCON*, vol. 2016-January, Jan. 2016, doi: 10.1109/TENCON.2015.7372918.

[46] K. Dolui and S. K. Datta, "Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing," *GIoTS 2017 - Global Internet of Things Summit, Proceedings*, Aug. 2017, doi: 10.1109/GIOTS.2017.8016213.

[47] B. Batrinca and P. C. Treleaven, "Social media analytics: a survey of techniques, tools and platforms," *AI & SOCIETY 2014 30:1*, vol. 30, no. 1, pp. 89–116, Jul. 2014, doi: 10.1007/S00146-014-0549-4.

[48] "Apache Spark™ - Unified Analytics Engine for Big Data." https://spark.apache.org/ (accessed Sep. 06, 2021).

[49] "Apache Hadoop." https://hadoop.apache.org/ (accessed Sep. 06, 2021).

[50] "Apache Mesos." http://mesos.apache.org/ (accessed Sep. 06, 2021).

[51] R. Iqbal, F. Doctor, B. More, S. Mahmud, and U. Yousuf, "Big data analytics: Computational intelligence techniques and application areas," *Technological Forecasting and Social Change*, vol. 153, p. 119253, Apr. 2020, doi: 10.1016/J.TECHFORE.2018.03.024.

[52] B. T.k., C. S. R. Annavarapu, and A. Bablani, "Machine learning algorithms for social media analysis: A survey," *Computer Science Review*, vol. 40, p. 100395, May 2021, doi: 10.1016/J.COSREV.2021.100395.

[53] Z. Zainol *et al.*, "Association Analysis of Cyberbullying on Social Media using Apriori Algorithm," *International Journal of Engineering & Technology*, pp. 72–75, 2018, Accessed: Sep. 08, 2021. [Online]. Available: www.sciencepubco.com/index.php/IJET

[54] A. Cropper, S. Dumančić, and S. H. Muggleton, "Turning 30: New Ideas in Inductive Logic Programming," *IJCAI International Joint Conference on Artificial Intelligence*, vol. 2021-January, pp. 4833–4839, Feb. 2020, Accessed: Sep. 08, 2021. [Online]. Available: https://arxiv.org/abs/2002.11002v4

[55] G. Shen *et al.*, "Depression Detection via Harvesting Social Media: A Multimodal Dictionary Learning Solution," 2017, Accessed: Sep. 08, 2021. [Online]. Available: http://tinyurl.com/zrnrw5j.

[56] A. van den Oord, Y. Li, and O. Vinyals, "Representation Learning with Contrastive Predictive Coding," Jul. 2018, Accessed: Sep. 08, 2021. [Online]. Available: https://arxiv.org/abs/1807.03748v2

[57] S. Katoch, S. S. Chauhan, and V. Kumar, "A review on genetic algorithm: past, present, and future," *Multimedia Tools and Applications 2020 80:5*, vol. 80, no. 5, pp. 8091–8126, Oct. 2020, doi: 10.1007/S11042-020-10139-6.

[58] G. Cheng, C. Yang, X. Yao, L. Guo, and J. Han, "When Deep Learning Meets Metric Learning: Remote Sensing Image Scene Classification via Learning Discriminative CNNs," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, no. 5, pp. 2811–2821, May 2018, doi: 10.1109/TGRS.2017.2783902.

[59] M. v. Mäntylä, D. Graziotin, and M. Kuutila, "The evolution of sentiment analysis—A review of research topics, venues, and top cited papers," *Computer Science Review*, vol. 27, pp. 16–32, Feb. 2018, doi: 10.1016/J.COSREV.2017.10.002.

[60] W. Medhat, A. Hassan, and H. Korashy, "Sentiment analysis algorithms and applications: A survey," *Ain Shams Engineering Journal*, vol. 5, no. 4, pp. 1093–1113, Dec. 2014, doi: 10.1016/J.ASEJ.2014.04.011.

[61] "GitHub - aesuli/SentiWordNet: The SentiWordNet sentiment lexicon." https://github.com/aesuli/SentiWordNet (accessed Sep. 09, 2021).

[62] "SentiWords | HLT-NLP." https://hlt-nlp.fbk.eu/technologies/sentiwords (accessed Sep. 09, 2021).

[63] "GitHub - VADER Sentiment Analysis." https://github.com/cjhutto/vaderSentiment (accessed Sep. 09, 2021).

[64] M. Dolores, M. González, E. M. Cámara, M. Teresa, and M. Valdivia, "CRiSOL: Base de conocimiento de opiniones para el español * CRiSOL: Opinion Knowledge-base for Spanish," *Revista nº*, vol. 55, p. 143, 2015, Accessed: Sep. 09, 2021. [Online]. Available: http://sinai.ujaen.es/coah

[65] X. Saralegi and I. San Vicente, "Elhuyar at TASS 2013," *XXIX Congreso de la Sociedad Española de Procesamiento de Lenguaje Natural (SEPLN 2013). TASS 2013 - Workshop on Sentiment Analysis at SEPLN 2013*, pp. 143–150, 2013.

[66] K. Chaudhary, V. Malik, and A. Kumar, "Sentiment Analysis of Twitter Data Using Naive Bayes Algorithm", Accessed: Sep. 09, 2021. [Online]. Available: http://www.ijritcc.org

[67] S. Aftab, I. Ali, and M. Ahmad, "Sentiment Analysis of Tweets using SVM," *Article in International Journal of Computer Applications*, vol. 177, no. 5, pp. 975–8887, 2017, doi: 10.5120/ijca2017915758.

[68] N. C. Dang, M. N. Moreno-García, and F. D. la Prieta, "Sentiment Analysis Based on Deep Learning: A Comparative Study," *Electronics 2020, Vol. 9, Page 483*, vol. 9, no. 3, p. 483, Mar. 2020, doi: 10.3390/ELECTRONICS9030483.

[69] "Servicios web de analítica y minería de textos | MeaningCloud." https://www.meaningcloud.com/es (accessed Sep. 09, 2021).

[70] "Cloud Natural Language | Cloud Natural Language | Google Cloud." https://cloud.google.com/natural-language (accessed Sep. 09, 2021).

[71] "Text Analytics API: Azure Cognitive Services." https://docs.microsoft.com/es-es/azure/cognitive-services/text-analytics/ (accessed Sep. 09, 2021).

[72] "IBM Watson Natural Language Understanding - Visión general - España | IBM." https://www.ibm.com/es-es/cloud/watson-natural-language-understanding (accessed Sep. 09, 2021).

[73] “GitHub - miso-belica/sumy: Module for automatic summarization of text documents and HTML pages.” https://github.com/miso-belica/sumy (accessed Sep. 09, 2021).

[74] N. Díaz Roussel, “Estudio comparativo de herramientas para tareas de Procesamiento de Lenguaje Natural,” *Trabajo Fin de Grado*, 2020. https://riunet.upv.es/bitstream/handle/10251/149603/D%C3%ADaz%20-%20Estudio%20comparativo%20de%20herramientas%20para%20tareas%20de%20Procesamiento%20de%20Lenguaje%20Natural.pdf?sequence=1&isAllowed=y (accessed Sep. 09, 2021).

[75] “TASS.” http://tass.sepln.org/ (accessed Sep. 09, 2021).

[76] “Grupos de investigación | Sociedad Española de Procesamiento del Lenguaje Natural.” http://www.sepln.org/investigacion/grupos-de-investigacion (accessed Sep. 09, 2021).

[77] “Publicación de la Estrategia de Procesamiento del Lenguaje Natural | Sociedad Española de Procesamiento del Lenguaje Natural.” http://www.sepln.org/actualidad/noticias/publicacion-de-la-estrategia-de-procesamiento-del-lenguaje-natural (accessed Sep. 09, 2021).

[78] M. A. Paredes-Valverde, R. Colomo-Palacios, M. D. P. Salas-Zárate, and R. Valencia-García, “Sentiment Analysis in Spanish for Improvement of Products and Services: A Deep Learning Approach,” *Scientific Programming*, vol. 2017, 2017, doi: 10.1155/2017/1329281.

[79] “Welcome to Python.org.” https://www.python.org/ (accessed Sep. 08, 2021).

[80] I. Stancin and A. Jovic, “An overview and comparison of free Python libraries for data mining and big data analysis,” *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2019 - Proceedings*, pp. 977–982, May 2019, doi: 10.23919/MIPRO.2019.8757088.

[81] “How To Perform Sentiment Analysis in Python 3 Using the Natural Language Toolkit (NLTK) | DigitalOcean.” https://www.digitalocean.com/community/tutorials/how-to-perform-sentiment-analysis-in-python-3-using-the-natural-language-toolkit-nltk (accessed Sep. 11, 2021).

[82] “PyCharm: the Python IDE for Professional Developers by JetBrains.” https://www.jetbrains.com/pycharm/ (accessed Sep. 08, 2021).

[83] “PyDev.” https://www.pydev.org/ (accessed Sep. 08, 2021).

[84] “Home — Spyder IDE.” https://www.spyder-ide.org/ (accessed Sep. 08, 2021).

[85] “IDLE — Python 3.9.7 documentation.” https://docs.python.org/3/library/idle.html (accessed Sep. 08, 2021).

[86] “Sublime Text - Text Editing, Done Right.” https://www.sublimetext.com/ (accessed Sep. 08, 2021).

[87] “Atom.” https://atom.io/ (accessed Sep. 08, 2021).

[88] “welcome home : vim online.” https://www.vim.org/ (accessed Sep. 08, 2021).

[89] “GNU Emacs - GNU Project.” https://www.gnu.org/software/emacs/ (accessed Sep. 08, 2021).

[90] “Twitter Developer Platform docs.” https://developer.twitter.com/

[91] “Apache NiFi Documentation.” https://nifi.apache.org/docs.html (accessed Aug. 19, 2021).

[92] “Datos no relacionales y NoSQL - Azure Architecture Center | Microsoft Docs.” https://docs.microsoft.com/es-es/azure/architecture/data-guide/big-data/non-relational-data (accessed Sep. 05, 2021).

[93] “BSON (Binary JSON) Serialization.” https://bsonspec.org/ (accessed Sep. 05, 2021).

[94] “The MongoDB 3.6 Manual — MongoDB Manual.” https://docs.mongodb.com/v3.6/ (accessed Sep. 05, 2021).

[95] “pandas - Python Data Analysis Library.” https://pandas.pydata.org/ (accessed Sep. 10, 2021).

[96] “Natural Language Toolkit — NLTK 3.6.2 documentation.” https://www.nltk.org/ (accessed Sep. 10, 2021).

[97] “scikit-learn: machine learning in Python — scikit-learn 0.24.2 documentation.” https://scikit-learn.org/stable/ (accessed Sep. 10, 2021).

[98] L. Chiruzzo, M. Etcheverry, and A. Rosá, “Sentiment Analysis in Spanish Tweets: Some Experiments with Focus on Neutral Tweets,” 2020, doi: 10.26342/2020-64-13.

[99] “Máquinas de Vector Soporte (Support Vector Machines, SVMs).” https://www.cienciadedatos.net/documentos/34_maquinas_de_vector_soporte_support_vector_machines#Hiperplano_y_Maximal_Margin_Classifier (accessed Sep. 11, 2021).

[100] “View of Perspectives to Definition of Big Data: A Mapping Study and Discussion.” https://journalsojs3.fe.up.pt/index.php/jim/article/view/2183-0606_004.001_0006/232 (accessed Aug. 31, 2021).

Anexo 1 – Métricas y algoritmos

Se define en este Anexo el concepto de matriz de confusión y las métricas más comunes que se utilizan para evaluar el resultado del entrenamiento de los algoritmos de aprendizaje supervisado. A continuación, se describen los algoritmos de clasificación empleados y el resultado de su entrenamiento.

A1.1 Métricas

A1.1.1 Matriz de confusión

Se trata de una tabla en la que cada columna representa el número de predicciones de cada clase, y cada fila representa las instancias en la clase real. Permite ver, de forma visual, los aciertos y fallos que tiene un modelo tras la fase de aprendizaje. Además, a partir de los valores de cada posición de la matriz, se calculan una serie de métricas que permiten evaluar el rendimiento obtenido por el clasificador y comparar entre varios clasificadores [Tabla 7]. La matriz de confusión normalizada incluye en cada celda la división entre el valor de esa celda en la matriz de confusión y el "Total reales" de esa clase. De esta forma, se obtiene un valor (entre 0 y 1) independiente del número de casos de la clase, que permite comparar mejor.

Tabla 7. *Definición de matriz de confusión*

		Reales			
		(-)	**(0)**	**(+)**	
Clasificados	**(-)**	(-) verdaderos	(-) clasificados como (0) falsos (0)	(-) clasificados como (+) falsos (+)	**Total (-) reales**
	(0)	(0) clasificados como (-) falsos (-)	(0) verdaderos	(0) clasificados como (+) falsos (+)	**Total (0) reales**
	(+)	(+) clasificados como (-) falsos (-)	(+) clasificados como (0) falsos (0)	(+) verdaderos	**Total (+) reales**
		Total clasificados (-)	**Total clasificados (0)**	**Total clasificados (+)**	**Total casos**

A1.1.2 Exactitud (*Accuracy*)

La exactitud indica lo cerca que está el resultado obtenido del valor verdadero. Da una idea del sesgo de una estimación. Se calcula como la proporción de resultados verdaderos (ya sean negativos, neutros o positivos) dividido entre el número total de casos.

$$Exactitud = \frac{(-)\ verdaderos + (0)\ verdaderos + (+)\ verdaderos}{Total\ casos} \quad \textit{(Eq. 1)}$$

A1.1.3 Precisión (*Precision*)

La precisión se refiere a la dispersión del conjunto de valores obtenidos, es decir, cuántos de los valores clasificados en una clase realmente pertenecen a esa clase. Cuanto menor es la dispersión, mayor es la precisión. Se calcula como la proporción de resultados verdaderos de una clase dividido entre el número total de casos clasificados en esa clase.

$$Precisión\ (-) = \frac{(-)\ verdaderos}{Total\ clasificados\ (-)} \quad \textit{(Eq. 2)}$$

$$Precisión\ (0) = \frac{(0)\ verdaderos}{Total\ clasificados\ (0)} \quad \textit{(Eq. 3)}$$

$$Precisión\ (+) = \frac{(+)\ verdaderos}{Total\ clasificados\ (+)} \quad \textit{(Eq. 4)}$$

Python muestra también la media simple (“macro avg”) y la media ponderada (“weighted avg”), teniendo en cuenta los pesos relativos de cada clase:

$$P\ (macro\ avg) = \frac{P\ (-) + P\ (0) + P\ (+)}{3} \quad \textit{(Eq. 5)}$$

$$P\ (weighted\ avg) = \frac{P\ (-) * Tr\ (-)\ + P\ (0) + Tr\ (0) + P\ (+) * Tr\ (+)}{Tr\ (-) + Tr\ (0) + Tr\ (+)} \quad \textit{(Eq. 6)}$$

Donde “P” es ”Precisión” y “Tr” es “Total reales”

A1.1.4 Sensibilidad (*Recall*)

La sensibilidad representa la capacidad del clasificador para discriminar entre clases, esto es, indica cuántos de los valores reales de una determinada clase acabaron finalmente clasificados como pertenecientes a esa clase. Se calcula como la proporción de resultados verdaderos de una clase dividido entre el número total de casos reales de esa clase.

$$Sensibilidad\ (-) = \frac{(-)\ verdaderos}{Total\ (-)\ reales} \qquad \textit{(Eq. 7)}$$

$$Sensibilidad(0) = \frac{(0)\ verdaderos}{Total\ (0)\ reales} \qquad \textit{(Eq. 8)}$$

$$Sensibilidad(+) = \frac{(+)\ verdaderos}{Total\ (+)\ reales} \qquad \textit{(Eq. 9)}$$

$$S\ (macro\ avg) = \frac{S\ (-) + S\ (0) + S\ (+)}{3} \qquad \textit{(Eq. 10)}$$

$$S\ (weighted\ avg) = \frac{S\ (-) * Tr\ (-)\ + S\ (0) + Tr\ (0) + S\ (+) * Tr\ (+)}{Tr\ (-) + Tr\ (0) + Tr\ (+)} \qquad \textit{(Eq. 11)}$$

Donde “S” es ”Sensibilidad” y “Tr” es “Total reales”

A1.1.5 *F1 – Score*

El *F1 – Score* resume en una única métrica la precisión y la sensibilidad. Resulta útil cuando la distribución entre clases es desigual. Se calcula como el doble del producto de la precisión por la sensibilidad, todo ello dividido entre la suma de precisión más sensibilidad.

$$F1 - score\ (-) = \frac{2 * Precisión\ (-) * Sensibilidad\ (-)}{Precisión\ (-) + Sensibilidad\ (-)} \quad \textit{(Eq. 12)}$$

$$F1 - score\ (0) = \frac{2 * Precisión\ (0) * Sensibilidad\ (0)}{Precisión\ (0) + Sensibilidad\ (0)} \quad \textit{(Eq. 13)}$$

$$F1 - score\ (+) = \frac{2 * Precisión\ (+) * Sensibilidad\ (+)}{Precisión\ (+) + Sensibilidad\ (+)} \quad \textit{(Eq. 14)}$$

$$F1s\ (macro\ avg) = \frac{F1s\ (-) + F1s\ (0) + F1s\ (+)}{3} \quad \textit{(Eq. 15)}$$

$$F1s\ (weighted\ avg) = \frac{F1s\ (-) * Tr\ (-)\ + F1s\ (0) + Tr\ (0) + F1s\ (+) * Tr\ (+)}{Tr\ (-) + Tr\ (0) + Tr\ (+)} \quad \textit{(Eq. 16)}$$

Donde “F1s” es ”F1 – score” y “Tr” es “Total reales”

A1.2 Algoritmos

A1.2.1 *SVM*

Una máquina de vector soporte se basa en el concepto de clasificación por separación óptima que, a su vez, se basa en el concepto de hiperplano. En un espacio con un número determinado de dimensiones, un hiperplano posee una dimensión menos. En un espacio de 3 dimensiones sería un plano y en un espacio de 2 dimensiones sería una recta. Un hiperplano divide en dos mitades el espacio en que se encuentra. Los puntos del espacio que cumplen la ecuación del hiperplano están en el propio hiperplano, los que cumplen su inecuación con signo positivo están a un lado y los que la cumplen con signo negativo están en el otro lado [99].

Para un conjunto de datos perfectamente separables, existen infinitos hiperplanos posibles para separarlos. Para elegir el óptimo entre ellos, se selecciona el que se encuentra más alejado de todas las observaciones utilizadas en el entrenamiento, calculando la distancia de cada punto con el hiperplano. Se consigue la separación óptima cuando la distancia mínima entre el hiperplano y los puntos es la máxima posible. Los vectores soporte son hiperplanos que se van generando a medida que se calculan esas distancias, considerando óptimo al mejor de entre ellos tras un determinado número de iteraciones.

Para más de dos clases, no es inmediato generalizar lo anterior. Hay dos opciones, o bien generar hiperplanos comparando todos los posibles pares de clases, o bien comparar una frente a todas las demás. Como no siempre es posible separar perfectamente las clases, se recurre a hiperplanos que, aunque dejen fuera algunos puntos, sean más robustos y clasifiquen bien cuando se añaden nuevos puntos, como hacen los clasificadores SVC. En Python, el método “LinearSVC” equivale a “SVC” con el parámetro “kernel=linear” pero es más rápido. En cambio, “SVC” es más completo en las opciones que ofrece. Para este TFG se ha utilizado “kernel=rbf”, que realiza aproximaciones de tipo exponencial para la definición de los hiperplanos, consiguiendo mejores resultados que las de tipo lineal o polinómico.

A1.2.2 Naïve Bayes

Los métodos bayesianos utilizan el Teorema de Bayes asumiendo “de forma inocente” que los sucesos son totalmente independientes entre sí. El Teorema de Bayes postula la forma de calcular la probabilidad de un evento A en presencia de otro evento B (probabilidad condicionada), si conocemos las probabilidades de A y de B y la probabilidad complementaria de que se produzca B en presencia de A. Esta base probabilística genera algoritmos sencillos, que necesitan poca carga computacional y que arrojan resultados bastante aceptables en la mayoría de las situaciones.

Los diferentes clasificadores Naïve Bayes que se pueden encontrar en “scikit-learn” difieren en el tipo de distribución utilizada para determinar la probabilidad condicionada. Así tenemos “GaussianNB”, “MultinomialNB” o “BernouilliNB” si se utiliza distribución de Gauss, multinomial o de Bernouilli, respectivamente. “ComplementNB” es una adaptación de “MultinomialNB” para conjuntos no balanceados, utilizando estadísticas del complementario de cada clase para calcular los pesos del modelo.

A1.3 Detalle de resultados

A1.3.1 SVC

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.119,random_state=67)
clf = svm.SVC(kernel='rbf', C=10,random_state=3).fit(X_train, y_train)
```

Código 7. *Clasificador SVC*

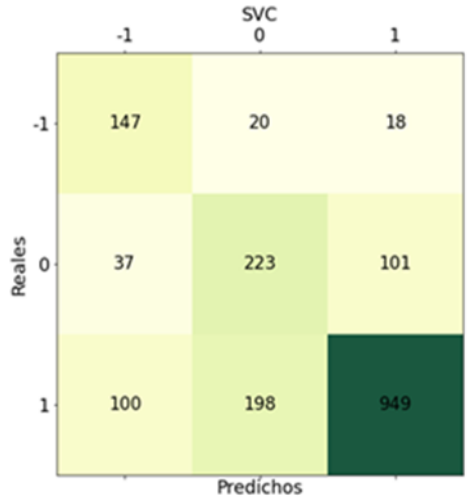

Figura 32. *Matriz de confusión – clasificador SVC*

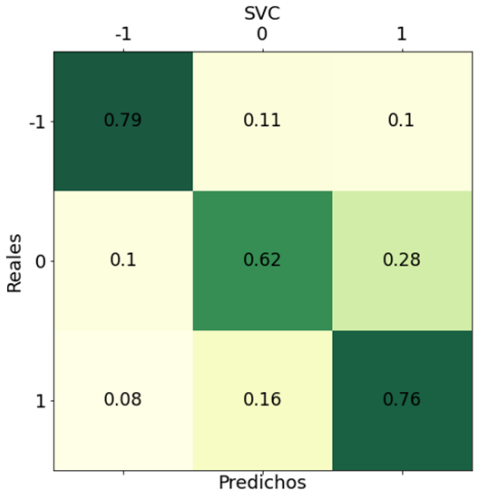

Figura 33. *Matriz de confusión normalizada – clasificador SVC*

Tabla 8. *Métricas - clasificador SVC*

		Accuracy	***0.7356***
Polaridad	***F1-score***	***Precision***	***Recall***
Negativa	0.63	0.52	0.79
Neutra	0.56	0.51	0.62
Positiva	0.82	0.89	0.76
Macro average	0.67	0.64	0.72
Weigthed average	0.75	0.77	0.74

A1.3.2 LinearSVC

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.149,random_state=10)
clf=LinearSVC(C=0.5)
```

Código 8. *Clasificador LinearSVC*

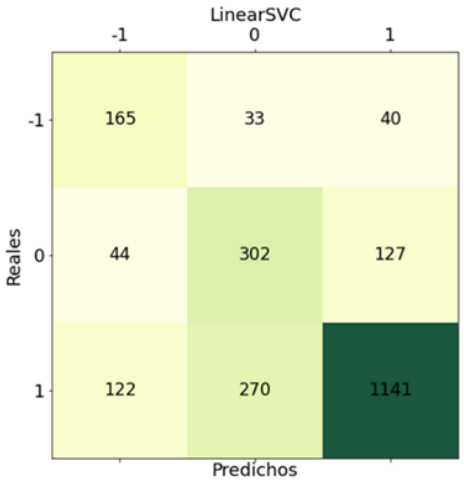

Figura 34. *Matriz de confusión – clasificador LinearSVC*

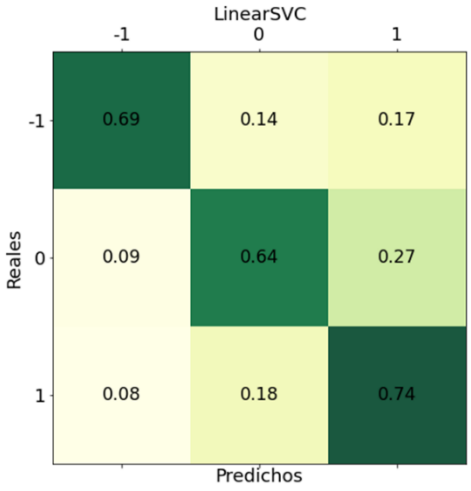

Figura 35. *Matriz de confusión normalizada – clasificador LinearSVC*

Tabla 9. *Métricas - clasificador LinearSVC*

		Accuracy	***0.7166***
Polaridad	***F1-score***	***Precision***	***Recall***
Negativa	0.58	0.50	0.69
Neutra	0.56	0.50	0.64
Positiva	0.80	0.87	0.74
Macro average	0.65	0.62	0.69
Weigthed average	0.73	0.75	0.72

A1.3.3 MultinomialNB

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.118,random_state=67)
clf = MultinomialNB(alpha=0.08)
```

Código 9. *Clasificador MultinomialNB*

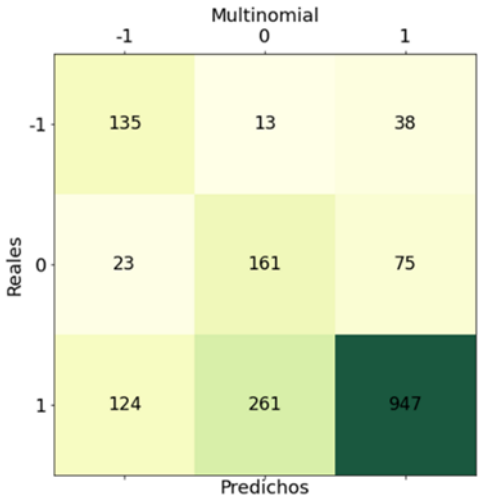

Figura 36. *Matriz de confusión – clasificador MultinomialNB*

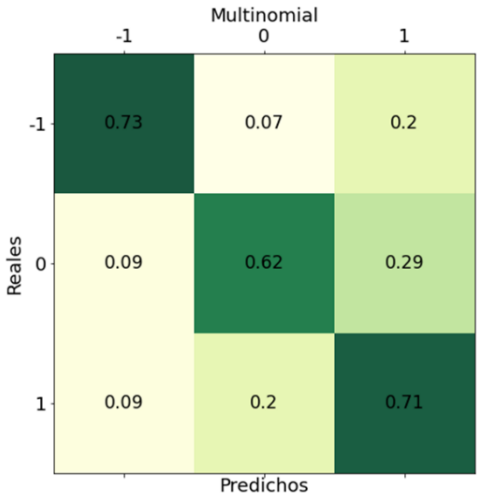

Figura 37. *Matriz de confusión normalizada – clasificador MultinomialNB*

Tabla 10. *Métricas - clasificador MultinomialNB*

		Accuracy	***0.6995***
Polaridad	***F1-score***	***Precision***	***Recall***
Negativa	0.58	0.48	0.73
Neutra	0.46	0.37	0.62
Positiva	0.79	0.89	0.71
Macro average	0.61	0.58	0.69
Weigthed average	0.72	0.77	0.70

A1.3.4 BernouilliNB

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.151,random_state=6)
clf = BernoulliNB(alpha=0.59)
```

Código 10. *Clasificador BernouilliNB*

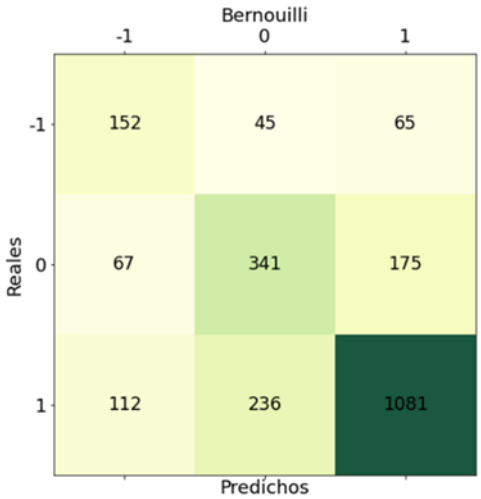

Figura 38. *Matriz de confusión – clasificador BernouilliNB*

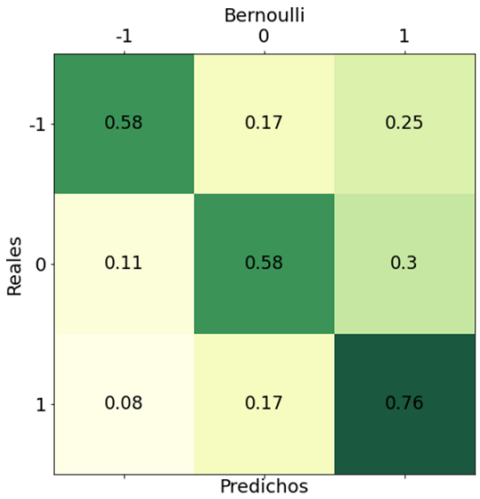

Figura 39. *Matriz de confusión normalizada – clasificador BernouilliNB*

Tabla 11. *Métricas - clasificador BernouilliNB*

		Accuracy	***0.6922***
Polaridad	***F1-score***	***Precision***	***Recall***
Negativa	0.51	0.46	0.58
Neutra	0.57	0.55	0.58
Positiva	0.79	0.82	0.76
Macro average	0.62	0.61	0.64
Weigthed average	0.70	0.71	0.69

A1.3.5 ComplementNB

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.159,random_state=47)
clf=ComplementNB(alpha=0.59)
```

Código 11. *Clasificador ComplementNB*

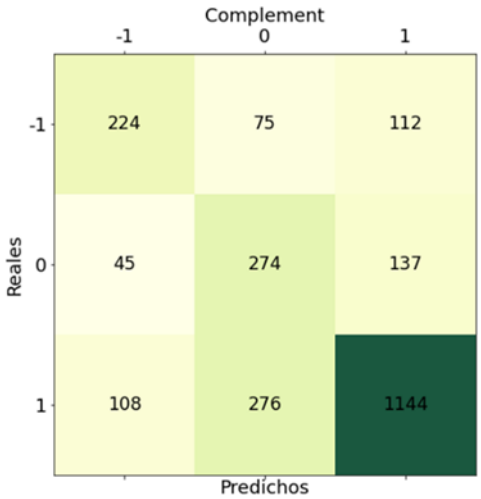

Figura 40. *Matriz de confusión – clasificador ComplementNB*

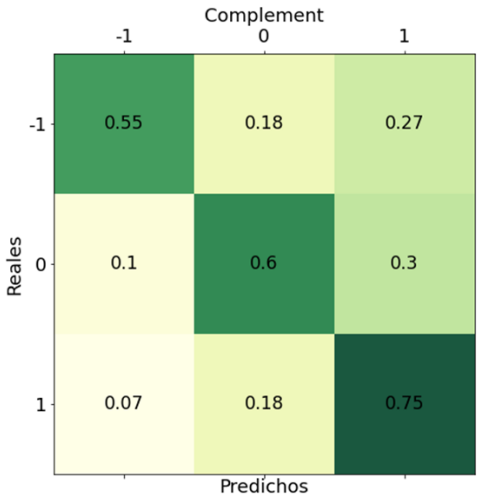

Figura 41. *Matriz de confusión normalizada – clasificador ComplementNB*

Tabla 12. *Métricas - clasificador ComplementNB*

		Accuracy	***0.6856***
Polaridad	***F1-score***	***Precision***	***Recall***
Negativa	0.57	0.59	0.55
Neutra	0.51	0.44	0.60
Positiva	0.78	0.82	0.75
Macro average	0.62	0.62	0.63
Weigthed average	0.69	0.71	0.69

A1.3.6 GaussianNB

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.16,random_state=4)
clf = GaussianNB()
```

Código 12. *Clasificador GaussianNB*

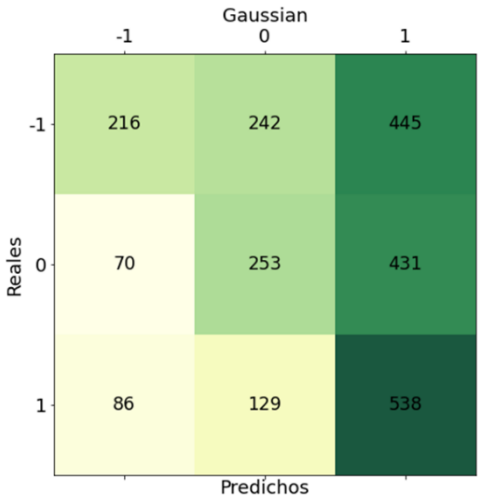

Figura 42. *Matriz de confusión – clasificador GaussianNB*

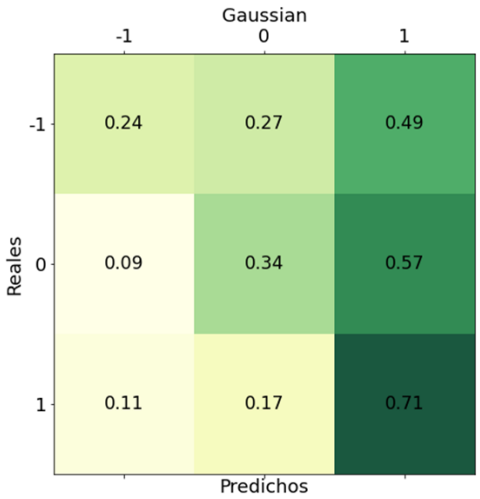

Figura 43. *Matriz de confusión normalizada – clasificador GaussianNB*

Tabla 13. *Métricas - clasificador GaussianNB*

		Accuracy	***0.4178***
Polaridad	***F1-score***	***Precision***	***Recall***
Negativa	0.34	0.58	0.24
Neutra	0.37	0.41	0.34
Positiva	0.50	0.38	0.71
Macro average	0.40	0.46	0.43
Weigthed average	0.40	0.46	0.42

Anexo 2 – Gráficas de evolución temporal

Se incluyen en este Anexo las gráficas generadas para realizar el análisis de evolución temporal y de su relación con los hitos mediáticos. Para cada mes se incluye:

tweets únicos *vs retweets* para polaridad positiva	*tweets* únicos vs *retweets* para polaridad negativa
tweets únicos *vs retweets* para polaridad neutra	
tweets únicos polaridad positiva *vs* polaridad negativa	*retweets* polaridad positiva *vs* polaridad negativa

Nota: Debido a que en ocasiones se han producido errores en el acceso a la API de Twitter, para ciertas fechas no se dispone de datos o de muy pocos. Cuando no hay datos, en las gráficas no aparece esa fecha en el eje x, pero cuando hay muy pocos sí aparece. En esos casos, en la gráfica queda reflejado por un descenso muy pronunciado fácilmente identificable.

A2.1 Diciembre 2020

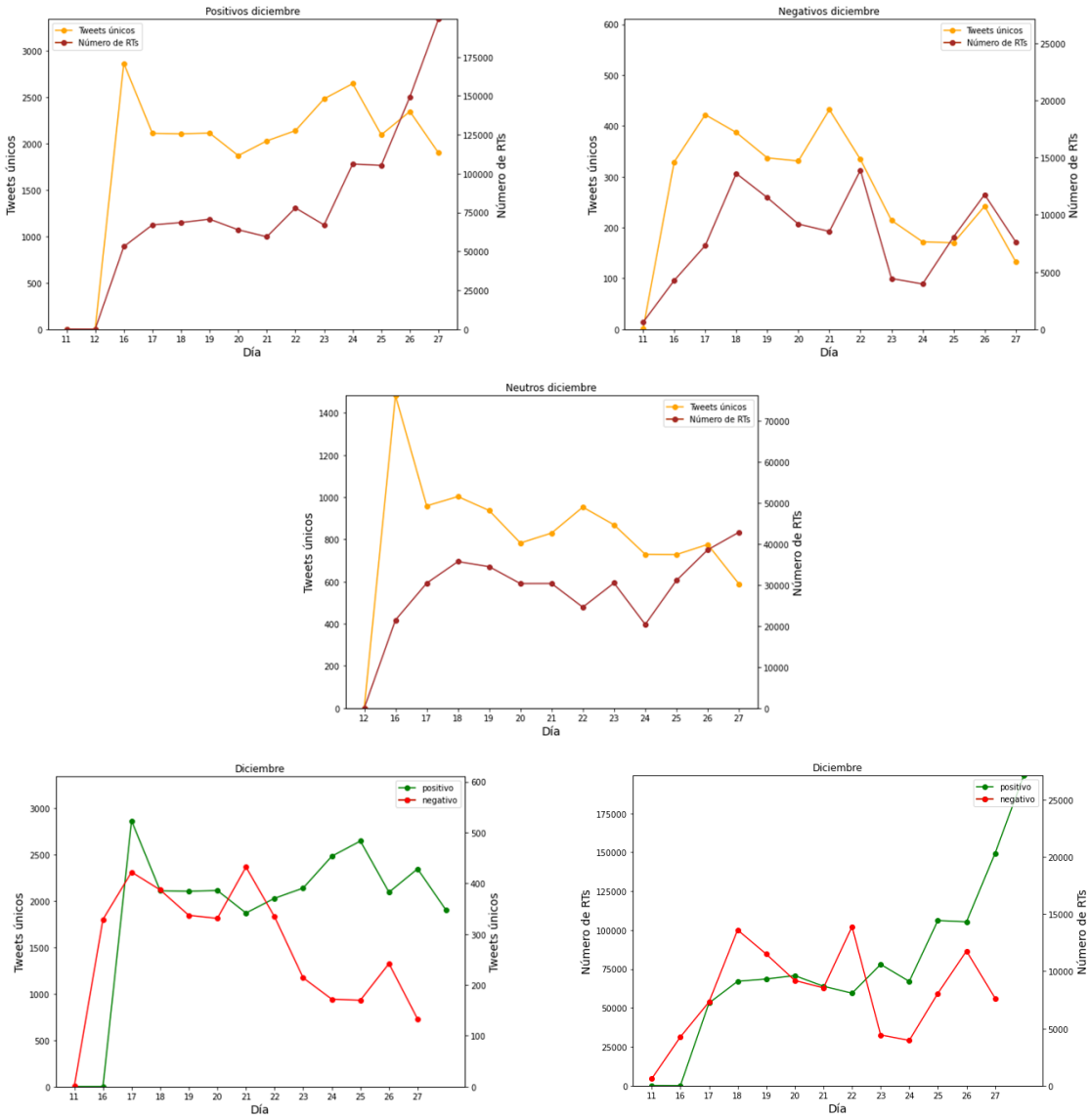

Figura 44. *Evolución polaridad diciembre 2020*

A2.2 Enero 2021

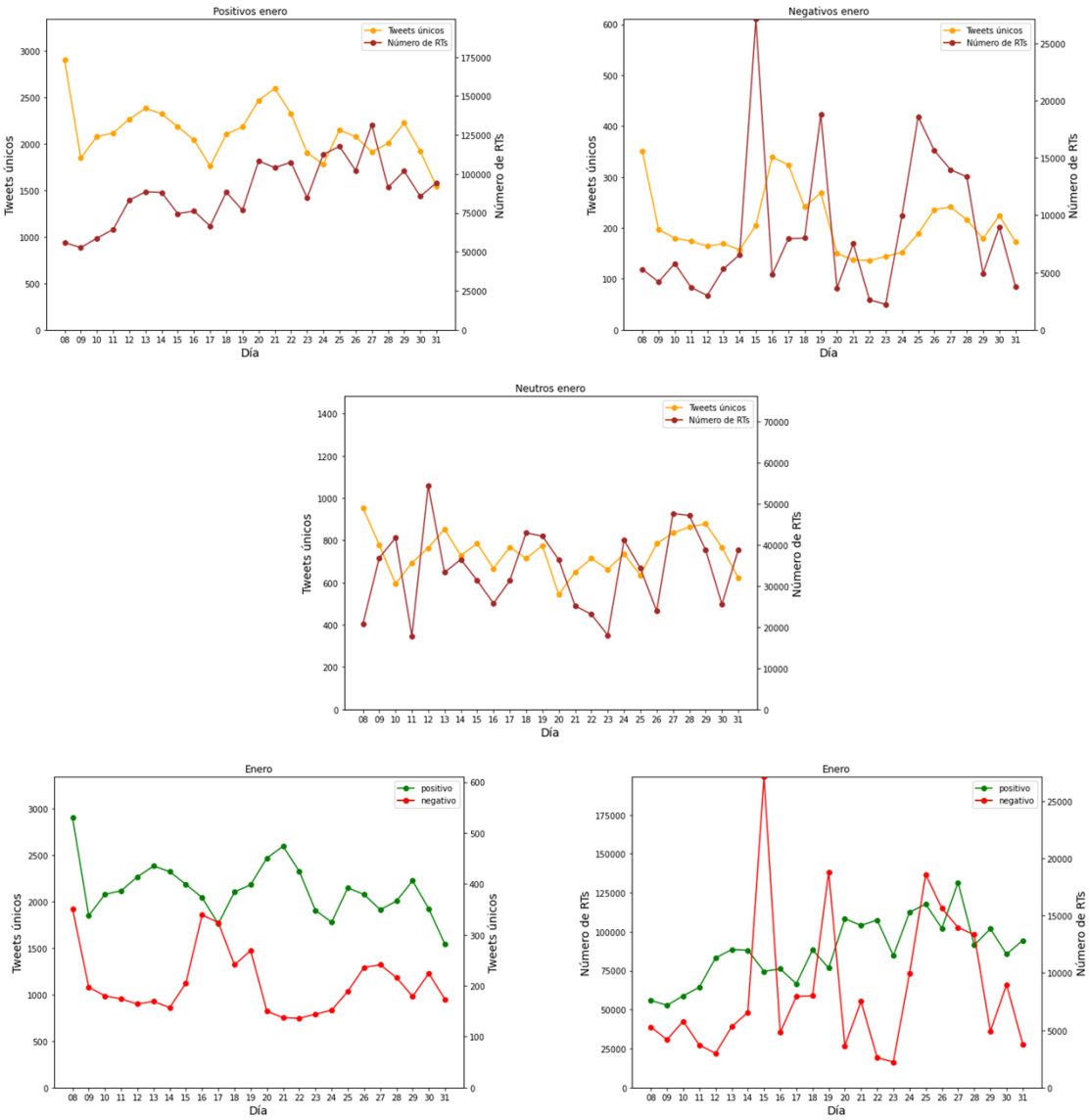

Figura 45. *Evolución polaridad enero 2021*

A2.3 Febrero 2021

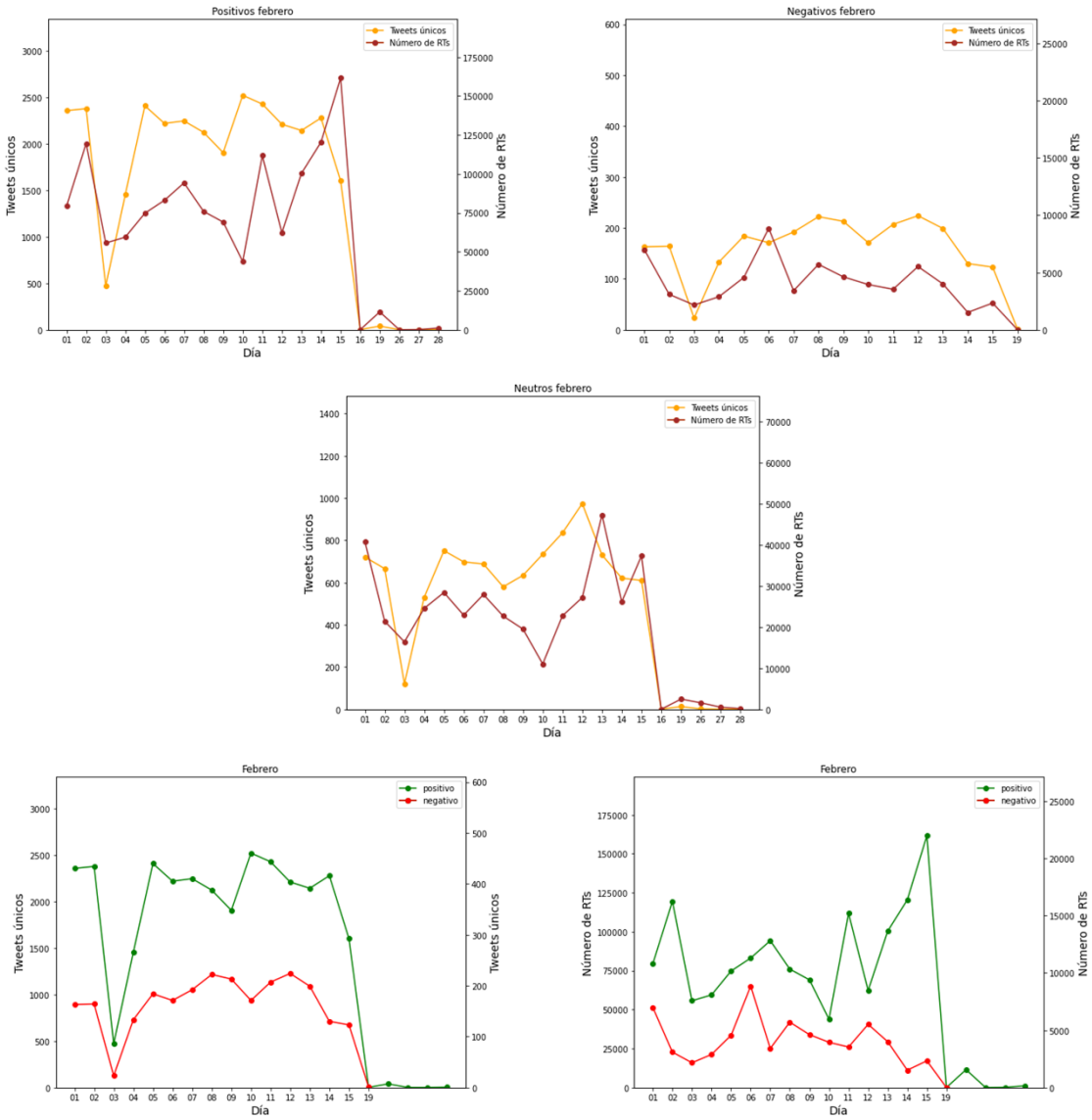

Figura 46. *Evolución polaridad febrero 2021*

A2.4 Marzo 2021

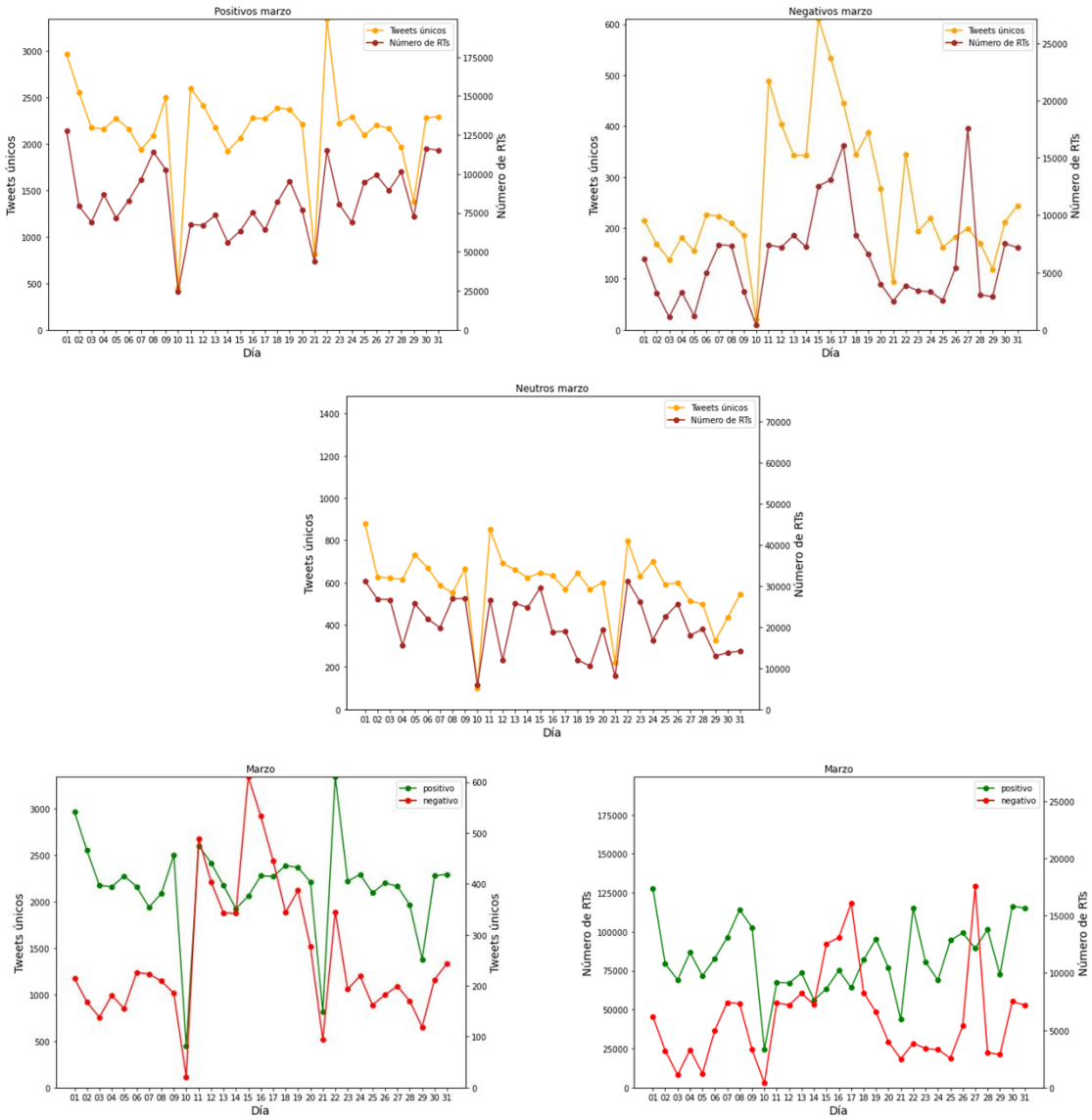

Figura 47. *Evolución polaridad marzo 2021*

A2.5 Abril 2021

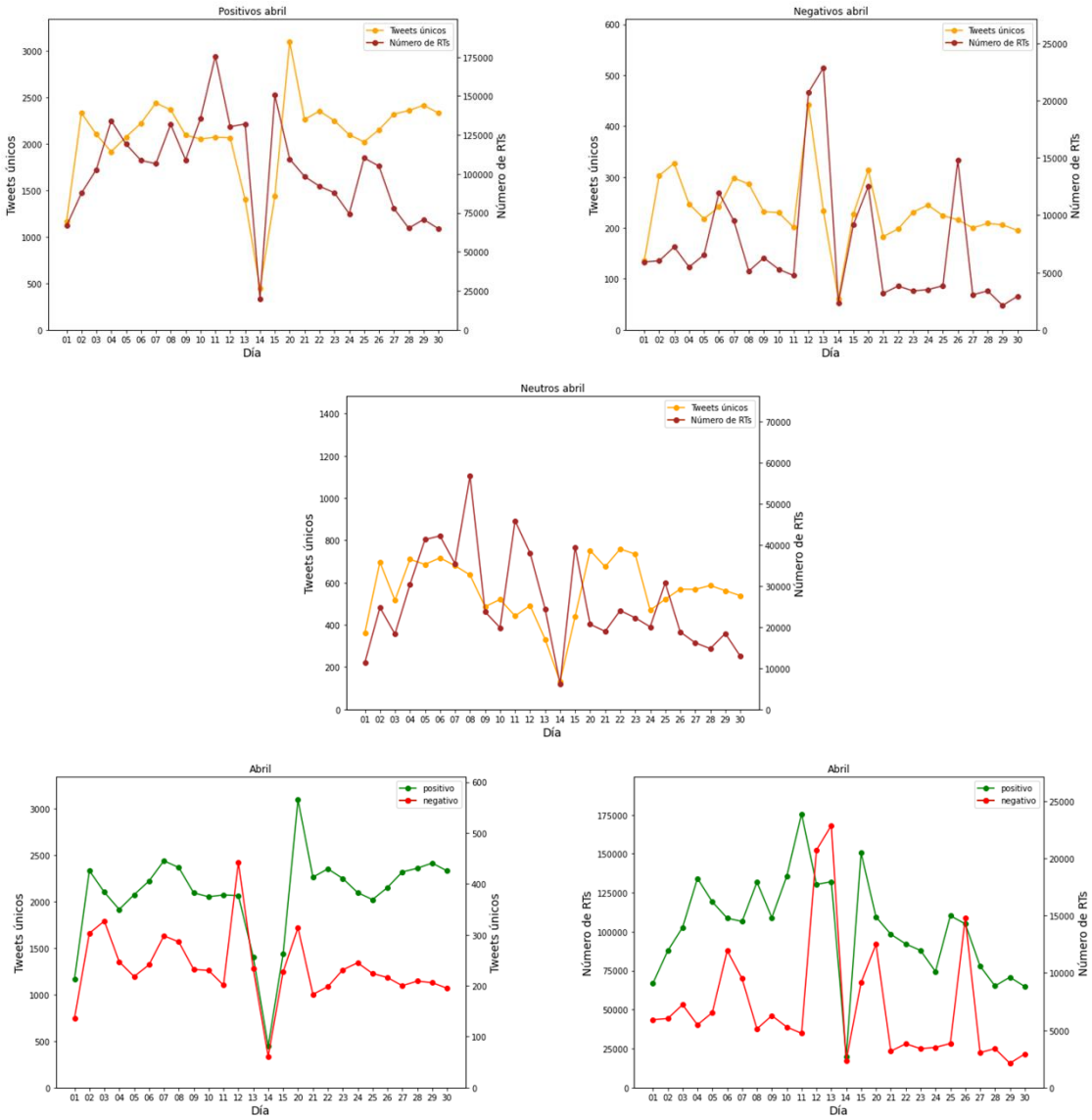

Figura 48. *Evolución polaridad abril 2021*

A2.6 Mayo 2021

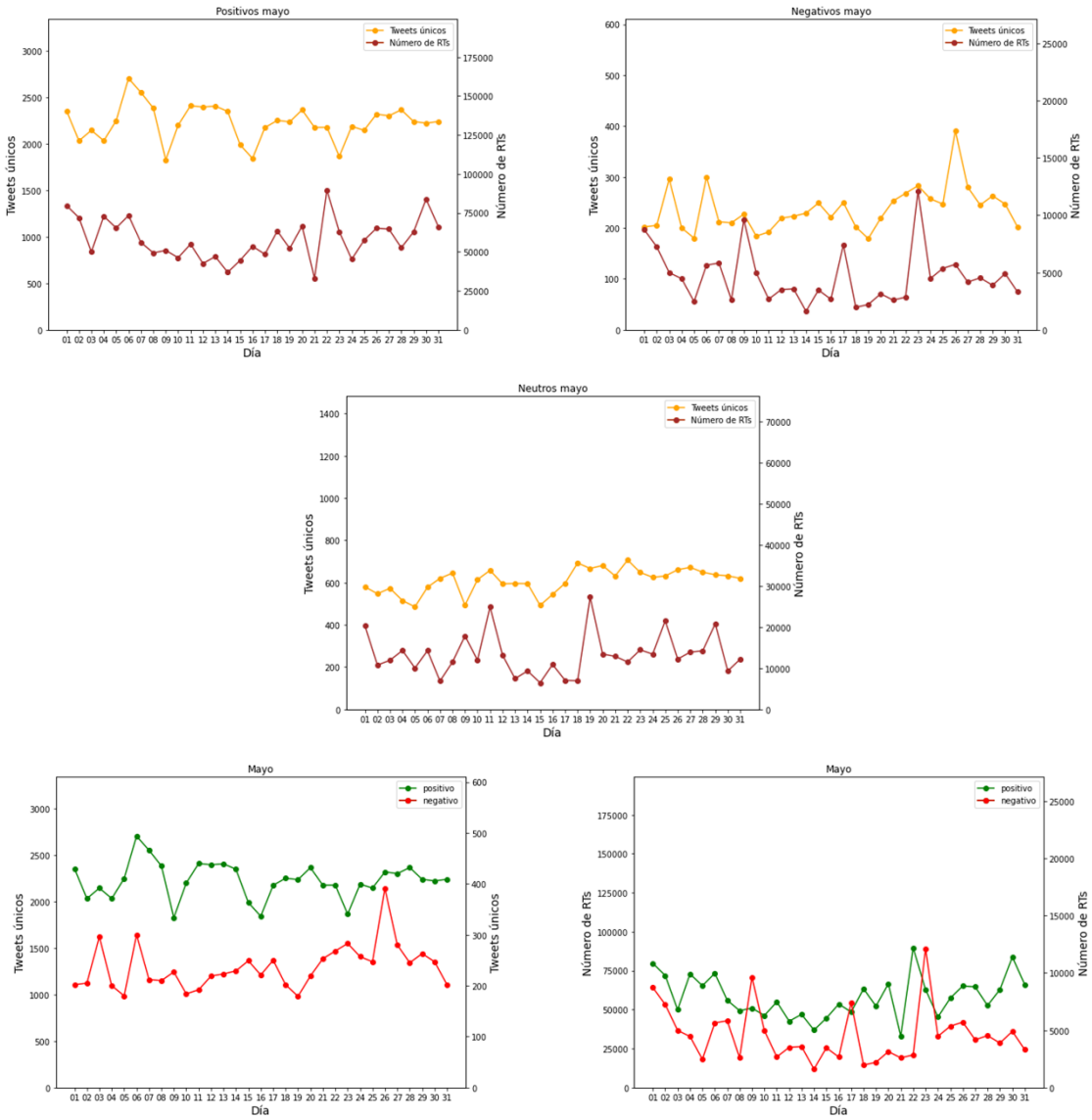

Figura 49. *Evolución polaridad mayo 2021*

A2.7 Junio 2021

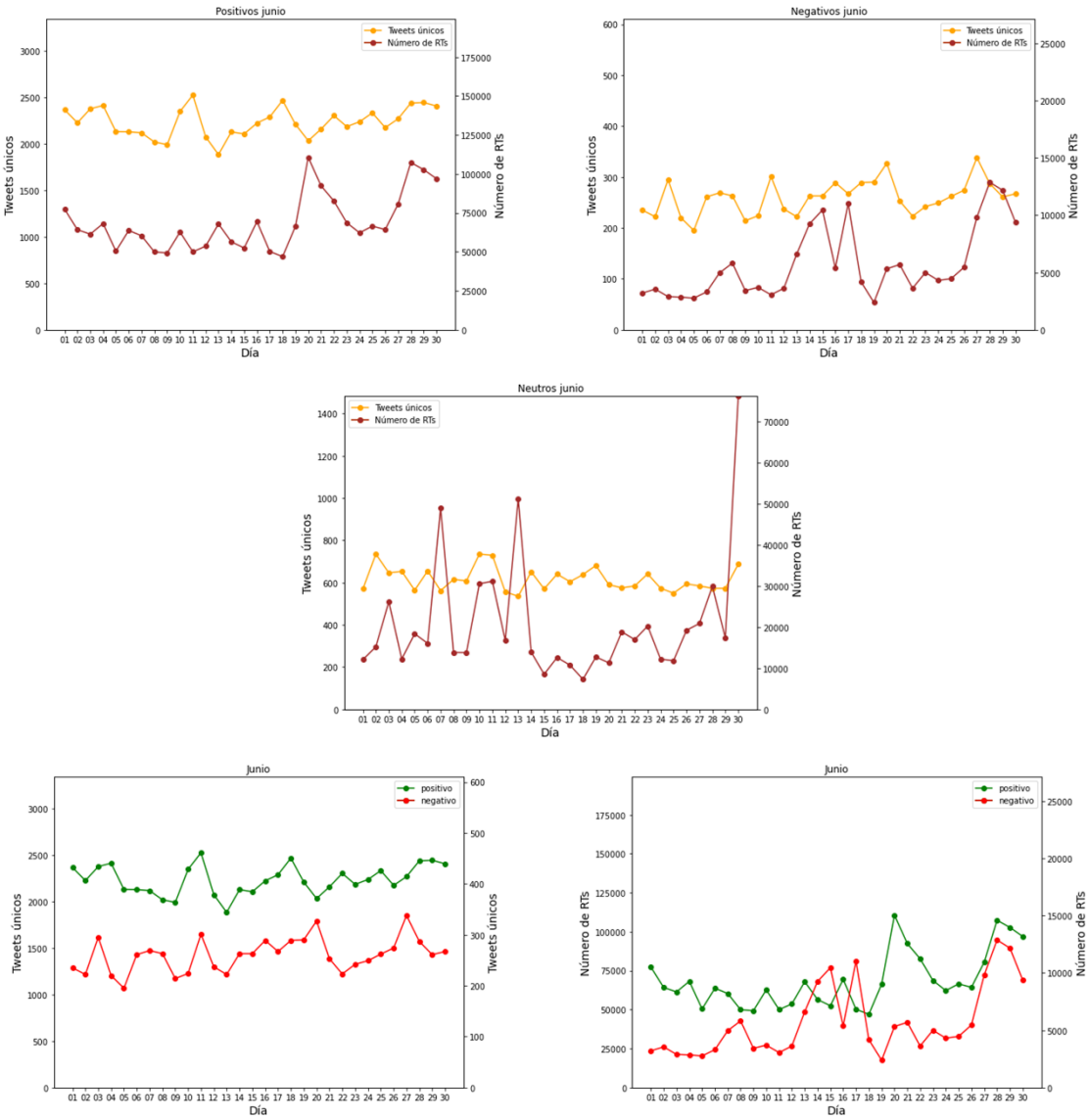

Figura 50. *Evolución polaridad junio 2021*

Anexo 3 – Presupuesto

Para la extracción de datos se ha utilizado espacio en un servidor departamental durante 9 meses, incluido como coste recurrente. Para la investigación, programación y procesado de datos se ha utilizado un ordenador de sobremesa no dedicado (AMD Ryzen 7, 3.60 GHz, 16 GB) y una conexión a internet por fibra óptica.

Se han empleado 260 horas para investigación y búsqueda de información y 240 horas para programación y pruebas. Se incorpora un margen de un 10% para contingencias, un 7% de coste de gestión del proyecto y un 12% de beneficio, todos ellos calculados sobre los gastos fijos. Por último, se añade un 21% por el Impuesto sobre el Valor Añadido actualmente vigente [Tabla 14].

Tabla 14. *Presupuesto de ejecución*

Recurso	Unidades	Precio unitario	Total
Gastos Fijos			
Conexión a Internet (meses)	4 meses	38 €	152 €
Ordenador (% dedicado)	25%	1.700 €	425 €
Ingeniero Informático (horas)	500 horas	16 €	8.000 €
		Subtotal	8.577 €
Contingencias (s/Subtotal)	10%	8.577 €	858 €
Gestión del proyecto (s/Subtotal)	7%	8.577 €	600 €
Margen (s/Subtotal)	12%	8.577 €	1.029 €
		Total Gastos Fijos	11.064 €
Gastos Recurrentes			
Servidor Departamental	9 meses	50 €	450 €
		Total Proyecto	11.514 €
Impuestos (s/Total Proyecto)	21%	11.514 €	2.418 €
		Total	**13.932 €**

Universidad de Alcalá

Escuela Politécnica Superior

ESCUELA POLITECNICA SUPERIOR