Universidades de Burgos, León y Valladolid

Máster universitario

Inteligencia de Negocio y Big Data en Entornos Seguros

TFM del Máster Inteligencia de Negocio y Big Data en Entornos Seguros

Una propuesta para la predicción de la intención de voto utilizando Twitter

Presentado por Jorge Silvestre Vilches
en Universidad de Valladolid
6 de septiembre de 2019

Tutores: Miguel Ángel Martínez Prieto
Aníbal Bregón Bregón

Resumen

Las redes sociales son, en la actualidad, una oportunidad tanto como un reto. El caudal de información que generan de manera constante las convierte en una valiosa fuente de datos para su aplicación a cualquier caso de uso de interés. En esta memoria, se plantea una propuesta de flujo de trabajo para, utilizando datos procedentes de Twitter, realizar las transformaciones necesarias con el fin de utilizar la información obtenida para realizar predicciones acerca del resultado de un evento electoral concreto, las elecciones generales en España del 28 de abril de 2019.

Descriptores

twitter, predicción de resultados electorales, procesamiento de lenguaje natural, aprendizaje automático supervisado, proceso etl

Abstract

Nowadays, social networks are both an oportunity and a challenge. The flow of information they are constantly producing makes them a valuable data source, with application to any use case of interest. In this report, we describe a workflow proposal to use data from Twitter to carry out the transformations that are needed in order to predict the outcome of a specific electoral event: the general elections in Spain on 28 April 2019.

Keywords

twitter, election forecast, natural language processing, supervised machine learning, etl process.

Índice general

Índice de figuras

Índice de tablas

Parte I

Introducción.
Descripción del proyecto.

Capítulo 1

Introducción.

La presente memoria de proyecto describe el proyecto realizado con motivo del Trabajo Fin de Máster, con el que se culmina la titulación de Máster Universitario en Inteligencia de Negocio y Big Data en Entornos Seguros (*Business Intelligence and Big Data in Cyber-Secure Environments*). En esta titulación, las instituciones implicadas en su elaboración (la Universidad de Burgos, la Universidad de León y la Universidad de Valladolid), han llevado a cabo un gran esfuerzo para unificar tres de las materias más pujantes en el entorno tecnológico actual, y cuyo impacto en los próximos años se prevé todavía más intenso: la inteligencia de negocio, el trabajo en entornos Big Data y la ciber-seguridad.

Este proyecto nace con una vocación similar: en él, trataremos de aplicar los conocimientos adquiridos durante la titulación a un caso de uso que involucra aspectos de gran relevancia de la actualidad, como son las redes sociales y la explotación de la extensa información que contienen, o la situación política en la sociedad. Nuestro objetivo a la hora de abordar este proyecto es evaluar la viabilidad de un sistema de predicción de resultados electorales en base a información capturada en redes sociales. En particular, se ha elegido la red social Twitter, dados el volumen y riqueza de la información que contiene, así como la frecuencia con que es utilizada por sus usuarios para expresar sus opiniones en el ámbito designado, la política, de forma abierta y sin ambages.

Tradicionalmente, se ha reconocido al encuestado medio una cierta vocación de mentiroso [1]. Especialmente, después de eventos particulares en los que las encuestas han visto cómo la realidad no refrendaba, ni de lejos, los resultados que predecían [2]. En ocasiones, esto se achaca a una muestra insuficiente, un procedimiento viciado a la hora de recabar los datos o la famosa “cocina”, que puede distorsionar los datos para extraer

unas conclusiones irreales. No obstante, la tendencia del ser humano a mentir en determinados aspectos es también un factor a tener en cuenta [3]. Las razones son variadas: asimilarse a las corrientes mayoritarias (o, al contrario, diferenciarse de ellas), sin que lo expresado refleje realmente el pensamiento del encuestado; responder con opciones diferentes a las reales por considerarlas vergonzantes; una dosis de auto-engaño... Temas como las costumbres, las vacaciones o la ideología política son especialmente complicados a la hora de obtener una respuesta sincera de todos los tipos de encuestados.

En este sentido, Internet ha supuesto una gran vía de escape para los impulsos de sinceridad de las personas [4]. Además del consabido anonimato que asegura el poder esconderse tras un seudónimo o *nickname* (lo cual convierte en innecesario el esconder la mano tras tirar las piedras correspondientes), existe el fenómeno de la "cámara de eco" [5], según el cual, en las redes sociales, uno puede encontrar un entorno en el que se refrenden y refuercen sus ideas, sean estas las que sean. Este fenómeno tiene un efecto colateral, y es la polarización de las opiniones [6][7]: sin argumentos críticos que contrasten y maticen estas opiniones, se entra en un círculo vicioso de reafirmación, en el que, además, será necesario expresarse de forma cada vez más extrema para sobresalir sobre el resto. Esto puede observarse también en medios de comunicación digitales, donde la necesidad de llamar la atención del público conduce, cada vez más, a titulares alarmistas y llamativos.

Twitter es un buen ejemplo de todo lo anterior. La facilidad para publicar en dicha red social (es de registro abierto y accesible desde casi cualquier dispositivo), su informalidad e inmediatez, y su adopción por parte de determinados círculos de opinión han generado un ambiente óptimo para que la gente publique sus opiniones de forma abierta y frecuente. Este hecho nos permitiría acceder al verdadero pensamiento de muchos de sus usuarios, por lo que los datos obtenidos podrían ser un interesante recurso a la hora de complementar o contrastar los recabados en una encuesta al uso.

Sin embargo, la utilización de tan valioso recurso presenta una serie de dificultades que será imprescindible salvar para obtener una solución satisfactoria. Algunas de ellas se derivan del propio concepto de *Big Data*, pero otras son específicas de la naturaleza de los textos publicados en la red social.

El principal escollo que presenta la explotación de Twitter tiene dos vertientes: volumen y variedad. El número de *tweets* publicados a cada momento es ingente (cerca de 9.000 mensajes por segundo[1]), por lo que cualquier procesamiento sobre el *stream* de Twitter en tiempo real debe ser

[1] `https://www.internetlivestats.com/one-second/`

al mismo tiempo rápido y eficiente en términos de consumo de memoria. Si trabajamos en *batch*, la primera condición se relaja, pero la segunda sigue siendo un factor relevante. En este abundante caudal de mensajes se tratan los más diversos temas, lo que dificulta cualquier extracción de información con un objetivo concreto: la cantidad de ruido presente en el canal resulta abrumadora. Por tanto, cualquier tipo de procesamiento pasa por implementar medios automáticos de captura, transformación y filtrado de la información obtenida de Twitter, implementados de manera que el sistema sea suficientemente escalable para ajustarse a los requisitos de cualquier caso de uso sobre un flujo de datos tan voluminoso.

Otro reto será la necesidad de tratar con lenguaje natural, lo que siempre presenta una enorme dificultad si se trata de llevar a cabo de forma automática, pero especialmente en el caso concreto de los *tweets*: su corta longitud y la espontaneidad de su redacción dificultan cualquier extracción de información, ya que recursos como la metáfora, las abreviaturas o las elipsis abundan en un medio como este. Finalmente, tenemos las dificultades derivadas de la escala de los datos que debemos tratar: el volumen de *tweets* publicados a cada minuto es muy significativo, por lo que será necesario explorar tecnologías Big Data que podamos utilizar (como el almacenamiento de los datos en un *Data Lake* para su procesamiento posterior), así como aproximaciones y técnicas que podamos introducir para reducir la dificultad del procesamiento de estos datos.

Una dificultad adicional, más sutil, es el posible sesgo de los datos que podamos obtener de las redes sociales, pues no todas las tendencias políticas tienen la misma presencia en las redes. No sorprendería encontrar que las opciones más extremas puedan quedar sobrerrepresentadas con respecto a su influencia real, pues suelen adoptar estrategias de comunicación más agresivas que las opciones moderadas, y las redes son un gran vehículo para alcanzar sus objetivos. Evaluaremos la presencia de este sesgo una vez contemos con los resultados finales.

Las dificultades que plantea el procesamiento automático de *tweets* conlleva una escasez de soluciones accesibles y efectivas para tratar con este tipo de textos. Esto es aún más cierto, en tanto en cuanto nuestro objetivo pasa por procesar textos en español. Esta escasez de recursos nos forzará a adoptar soluciones creativas, cuyo funcionamiento deberá ser evaluado y verificado. En este sentido, exploraremos la posibilidad de utilizar textos de similar naturaleza y con contenido de plena actualidad, como los comentarios publicados por lectores de medios de comunicación online, para ser capaces de identificar los *tweets* mediante el uso de técnicas de Procesamiento de Lenguaje Natural. A lo largo de la memoria describiremos cada fase de esta evolución, tratando de reseñar los aspectos más importantes y las soluciones

que se intentarán aplicar para solventar cada uno de los problemas indicados previamente.

1.1. Motivación y alcance del proyecto.

Este proyecto se origina en marzo de 2019, en vísperas de unas elecciones generales en nuestro país, fechadas para el día 28 de abril de ese mismo año. Este evento electoral se anticipaba interesante por múltiples factores. En primer lugar, por la situación de incertidumbre política por la que atraviesa nuestro país, que amenazaba con reconfigurar el panorama político debido a la debilidad de algunos partidos y el auge de otros. En segundo lugar, por el debate existente acerca de la veracidad atribuible a las encuestas, motivado fundamentalmente por los resultados que, mes a mes, iba publicando el *CIS* (Centro de Investigaciones Sociológicas), cuestionados por algunos tanto por las conclusiones que arrojaban, como por la metodología que se aplicaba.

Este ambiente de incertidumbre nos hacía preguntarnos si existiría algún medio alternativo para tomar el pulso a la opinión pública en términos de su afinidad política, sin el tamiz ideológico o interesado que pudieran conferir los diferentes servicios de demoscopia a los resultados que ofrecen. En este sentido, tal y como se aludió en la introducción de esta memoria, las redes sociales, y Twitter en particular, podrían constituir un valioso recurso para obtener información de esta naturaleza, especialmente en estos tiempos en los que la política parece estar en boca de todos, sea por razones correctas o incorrectas.

Esta idea motivó la definición de este proyecto como propuesta de Trabajo Fin de Máster. Además de tener cierto potencial de descubrimiento (aunque limitado por el alcance que debe tener un trabajo de estas características), abordar este proyecto daba pie a poner en práctica varios de los conocimientos adquiridos durante la titulación, como el procesamiento de datos en un contexto de Big Data y la aplicación de aprendizaje automático (*machine learning*) a un problema de clasificación de textos.

Por esta razón, se decidió tratar de definir un un sistema *proof of concept* para la obtención de predicciones de intención de voto sobre una muestra de población, formada por los usuarios de Twitter que publican *tweets* expresando sus opiniones políticas. Este sistema debe implementar un flujo de trabajo que nos permita obtener los datos necesarios de la fuente original, y transformarlos de manera que podamos extraer la información objetivo de ellos. Sin embargo, es importante tener en cuenta las restricciones a que está sujeto el proyecto y establecer un alcance que nos permita completar el mismo en tiempo y forma.

La propia fuente de datos constituía un gran reto tanto técnico como tecnológico para ser utilizada en este proyecto. Como se ha mencionado antes, el volumen de datos proporcionado en el *stream* en tiempo real de Twitter es ciertamente grande: en la red social se publican cerca de 9.000 mensajes por segundo, unos 800 millones al día, de los cuales se estima que el 1 % se sirven a través de la API [8], aunque esta referencia data del 2013. Por ello, decidimos evaluar un mecanismo de selección de contenidos sobre la propia fuente basado en los *trending topics* identificados por Twitter: capturaremos únicamente aquellos *tweets* relacionados con una o más de las tendencias del momento, actualizando el filtro periódicamente para adaptarnos a la evolución de las características del flujo, y descartaríamos el resto. Este filtro debía cumplir un doble objetivo:

- Mejorar la escalabilidad del sistema al tiempo que se reduce el ruido presente en los datos capturados, lo que reduciría tanto el volumen de datos a procesar, como la complejidad de los mismos. Utilizando este enfoque, podríamos centrar nuestra atención en la información de mayor relevancia en cada instante, logrando que una parte significativa del "ruido de fondo" en el *streaming* no sea capturada.
- Facilitar la adaptabilidad del proceso a otros casos de uso. El estudio de los *tweets* en relación a las tendencias nos proporciona una forma de agrupación "semántica" (pues la mayoría de los *tweets* de una tendencia abordarán el mismo tema) que podría ser explotada para descartar fácilmente los *tweets* que no pertenezcan a *trending topics* acordes con nuestros intereses. Simplemente modificando el mecanismo de clasificación de estos *trending topics*, podríamos seleccionar por simple correspondencia los *tweets* relevantes.

El criterio de decisión acerca de la relevancia de un *trending topic* también debe ser abordado, pues su solución no es inmediata. Es difícil caracterizarlo o asignarle una categoría por sí mismo, ya que suelen tener una longitud demasiado corta para ser procesado mediante clasificadores tradicionales, y en ocasiones pueden resultar ambiguos incluso a ojos de un humano. Por tanto, será necesario definir un mecanismo que nos permita identificar, ahondando en el objetivo de la generalización de la solución, las tendencias relevantes para el caso de uso que seleccionemos. En este proyecto proponemos abordar esta cuestión mediante *tweet pooling*, o procesamiento de *tweets* en conjunto, para caracterizar la semántica de la tendencia a la que pertenecen.

Por otro lado, el trabajo con redes sociales implica un fuerte componente de Procesamiento de Lenguaje Natural (o NLP, de sus siglas en inglés *Natural Language Processing*). El lenguaje natural siempre supone un reto para

su procesamiento computacional, pues la complejidad de sus estructuras, la variabilidad de los elementos que lo componen y su tendencia a los errores hace imposible abordarlo mediante técnicas simples e inmediatas. En este sentido, será necesario definir un flujo de procesamiento dentro de nuestras posibilidades, pero suficientemente efectivo, para extraer información valiosa de una fuente especialmente caótica, como es Twitter. Para ello, nos ayudaremos de recursos útiles en el estado del arte, como la plataforma Google Cloud, para abordar de forma asequible tan complejos problemas.

En resumen, deberemos explorar los medios necesarios para aportar una solución que proporcione, al menos, un punto de partida para verificar la premisa que motiva este Trabajo Fin de Máster. Dado el carácter de investigación que atribuimos al proyecto, hemos considerado oportuno formular esta premisa en forma de *pregunta de investigación*, que enunciamos a continuación:

PI-01 ¿Es posible predecir, dentro de un margen de error prudente, los resultados de un evento electoral atendiendo a la información más prominente disponible en redes sociales (por ejemplo, la incluida en las tendencias del momento), en un período de tiempo previo a dicho evento?

A la luz de esta premisa, podemos identificar una serie de objetivos que guíen el desarrollo de este trabajo hacia la obtención de una respuesta satisfactoria de esta pregunta (ver Tabla 1.1).

1.2. Estructura del documento.

El presente documento se divide en diferentes capítulos, en los que abordaremos todos los aspectos implicados en el desarrollo del proyecto que abordamos. En el Capítulo 2 se describe la planificación establecida para el proyecto, así como la metodología con la que se desarrollarán los trabajos y la previsión de costes derivados de los mismos. En el Capítulo 3 se lleva a cabo una breve revisión del contexto en que tiene lugar el proyecto, así como del estado del arte en la materia, que nos permitirá identificar algunas guías de utilidad para abordar los diferentes retos que plantea el proyecto.

En el Capítulo 4 se describe, desde un punto de vista técnico, las características del producto software que deberá ser implementado, y que determinarán los distintos procesos de desarrollo que se describen en los siguientes capítulos. En el Capítulo 5 se explica el proceso de obtención y refinamiento de datos, en el que tendrá un papel fundamental el clasificador cuyo desarrollo se documenta en el Capítulo 6. Los datos procesados serán

Objetivo	Descripción
OBJ-01	Implementar un mecanismo de captura de información adaptable a la evolución del flujo de datos, atendiendo a la información de actualidad más relevante en cada momento para favorecer la escalabilidad del sistema.
OBJ-02	Diseñar y construir un proceso automático de filtrado, transformación y enriquecimiento de datos que permita explotar la información capturada con vistas al caso de uso planteado.
OBJ-03	Desarrollar un sistema supervisado de clasificación semántica de *tweets* utilizando recursos de más fácil acceso y tratamiento que los *tweets*, evaluando si la diferente naturaleza de los textos a clasificar no conlleva una pérdida de efectividad importante.
OBJ-04	Construir un mecanismo de predicción de resultados electorales utilizando información de redes sociales y análisis de sentimiento.

Tabla 1.1: Preguntas de investigación del proyecto.

utilizados por el sistema de predicción descrito en el Capítulo 7, cuyos resultados deberían permitirnos responder a la pregunta de investigación que gobierna el proyecto en su conjunto. Finalmente, expondremos nuestras conclusiones y definiremos las líneas de trabajo futuro en el Capítulo 8.

Capítulo 2

Metodología y planificación.

En este capítulo se describe la metodología utilizada para gestionar el desarrollo del proyecto, así como la estimación del esfuerzo que requerirá el llevarlo a cabo. A partir de esta estimación, se elaborará una planificación temporal sobre la que realizaremos una estimación inicial de costes derivados del proyecto. Finalmente, se revisarán estas valoraciones iniciales tras la conclusión del proyecto para reflejar las posibles desviaciones que se hayan producido con respecto a las previsiones realizadas.

2.1. Metodología de desarrollo.

Dado el carácter experimental del proyecto, será necesario escoger una metodología flexible que nos permita reaccionar de forma ágil ante cambios en los requisitos, reflejándolos lo antes posible en el plan de trabajo. En este caso se ha optado por utilizar el marco de trabajo Scrum para diseñar la planificación del proyecto, así como para llevarlo a cabo.

Scrum [9] es un marco de trabajo *Agile* que se articula sobre el principio de *entregas frecuentes* y *software funcionando*, de manera que un equipo de trabajo (en este caso unipersonal) sea capaz de, en iteraciones cortas, incrementar progresiva y continuamente el valor del producto en desarrollo mediante la adición sucesiva de nuevas funcionalidades que puedan ser explotadas inmediatamente tras su entrega. Este principio es especialmente apropiado para proyectos donde la modularidad sea alta, puesto que pueden definirse los diferentes módulos como objetivos o incrementos para las iteraciones planificadas.

Scrum introduce el concepto de *sprint* para definir las iteraciones. Los *sprints* son períodos de tiempo de duración fija, que comienzan estableciendo

los objetivos a alcanzar durante el mismo y terminan con la entrega de un incremento del producto desarrollado y la revisión del avance global del proyecto. Los requisitos del proyecto se describen como historias (*stories*), que son posteriormente divididas en tareas (*tasks*) para el reparto del trabajo dentro del equipo de desarrollo. Las historias y tareas que deben ser abordadas en cada *sprint* se deciden al comienzo del mismo como resultado del consenso entre el equipo de trabajo y el *Product Owner* del proyecto.

El *Product Owner* juega un papel fundamental en el proceso Scrum, pues es quien posee la visión del proyecto, conoce las necesidades de los *stakeholders* y orienta al equipo de desarrollo en la dirección y necesidades de negocio del proyecto en su conjunto. En este caso, el papel de *Product Owner* lo ejercen los tutores del Trabajo, que deberán velar por que el Trabajo cumpla con los requisitos esperados, así como de proporcionar al equipo de trabajo (el alumno) la guía necesaria para llevarlo a buen puerto. Al final de cada *sprint* se llevará a cabo una reunión (*sprint review*) con el *Product Owner* para informar del avance del proyecto, mostrar los resultados obtenidos y esbozar los pasos a realizar en los siguientes *sprints*. En el ámbito de este Trabajo, esta *sprint review* servirá también como reunión de inicio para el siguiente *sprint*, dado que deberán fijarse los objetivos para el *sprint* inmediatamente posterior.

Además de los productos software que se vayan generando, se considerará la memoria del proyecto como el entregable final del proyecto, por lo que su elaboración también se incluirá como historia que debe ser abordada por el equipo de desarrollo.

2.2. Planificación del trabajo.

La asignatura de Trabajo Fin de Máster consta de 9 créditos ECTS, que se traducen en 225 horas de trabajo efectivo de acuerdo con la guía docente de la misma. En consecuencia, tanto el alcance como la duración del proyecto deberán ajustarse a estos requisitos temporales.

El desarrollo del proyecto se estructurará en tres *sprints*, con una duración de dos semanas cada uno. La fecha prevista para el comienzo del proyecto es el 22 de mayo de 2019, concluyendo el último *sprint* el 26 de junio del mismo año. No obstante, las tareas relativas a la captura de datos de Twitter se llevaron a cabo con anterioridad, durante la segunda mitad de marzo, a fin de ser capaces de capturar, en tiempo real, información sobre el evento electoral elegido a lo largo del mes de abril. El tiempo dedicado a esta operación fueron 18 horas, que asignaremos al primer *sprint* por simplicidad.

En el marco de los objetivos generales del proyecto, se han identificado las historias de trabajo:

H-01 Extracción de información de Twitter

Esta historia comprende el diseño e implementación de un sistema de captura de *trending topics* y *tweets* en tiempo real a través de la API de Twitter para generar el conjunto de datos sobre el que se realizará todo el procesamiento posterior. Será necesario un estudio previo de la API de Twitter, tanto en lo referente a métodos de extracción como a la naturaleza de los datos que provee.

H-02 Limpieza y transformación de datos

Esta historia incluye las operaciones de limpieza, selección y consolidación de los datos capturados. Deberá definirse un *pipeline* de procesamiento que satisfaga el mapa lógico de datos, que guía la transformación desde los datos en crudo hasta el modelo de datos final, el cual también deberá determinarse. Al trabajar con lenguaje natural, es necesario realizar ciertas operaciones de tratamiento que habiliten los datos para su procesamiento por medios automáticos. El resultado debe ser un conjunto de datos preparado para su uso.

H-03 Clasificación y filtrado de datos

Tras la transformación de los datos, es necesario determinar qué datos de todos los capturados pueden ser aplicados al caso de uso elegido, la predicción de la distribución de intención de voto. Se estudiarán los medios necesarios para llevar a cabo la clasificación de los datos en relativos a política y no relacionados con la misma. De esta forma, se generará un *dataset* reducido y enriquecido con los datos imprescindibles para la realización de predicciones, implementando un sencillo proceso de transformación que descarte la información innecesaria y consolide la más relevante.

H-04 Predicción de datos

Una vez contemos con datos relevantes y aptos para su explotación, se diseñará e implementará un mecanismo de predicción de resultados electorales que trate de aproximar, con distintos plazos de adelanto, lo resultados finales par el evento político elegido.

H-05 Documentación del proyecto

Cada fase del proyecto debe ser documentada y consolidada en una memoria, que deberá cumplir con la estructura establecida en la

titulación, sin perjuicio de cualquier otra documentación que se pueda o deba generar durante el proceso (documentación del código, recursos adicionales, etcétera). Esta memoria abarcará cada uno de los aspectos del proceso de desarrollo, incluyendo la contextualización, planificación, descripción de componentes y realización de conclusiones relacionados con el proyecto.

Para cada una de las historias se han definido las siguientes tareas, aunque se trata de una aproximación especulativa al comienzo del proyecto y podrían agregarse nuevas tareas o modificar las ya existentes durante el proceso de desarrollo, como es prerrogativa del equipo de desarrollo de acuerdo a los principios de Scrum. El grado de dificultad de cada tarea se estimará utilizando una técnica común en Scrum, la estimación de póquer [10] en su variante de sucesión de Fibonacci: de esta manera, podemos estimar la complejidad relativa de cada tarea planteada, y dividirlas de forma equilibrada entre los *sprints*.

Tarea	Descripción
T-01-01	*Análisis de la API de Stream.* (2 puntos) Analizar los componentes y métodos requeridos para la captura periódica de *trending topics* para la localización "España" y la captura en tiempo real de *tweets* pertenecientes a los mismos.
T-01-02	*Descripción de la fuente de datos.* (2 puntos) Describir el formato y la estructura de los datos proporcionados como resultado de las consultas a la API de Twitter.
T-01-03	*Elección del mecanismo de acceso a la API.* (1 punto) Determinar cómo se accederá a los distintos métodos de la API, pudiendo ser de forma directa o a través de librerías de terceros. Diseñar la implementación del medio elegido.
T-01-04	*Desarrollo del componente de captura.* (3 puntos) Implementar los medios elegidos para realizar la recogida de datos, y completar el proceso de captura.

Tabla 2.1: Tareas de la Historia 01.

Tarea	Descripción
T-02-01	*Análisis y diseño del modelo de datos objetivo.* (2 puntos) Consolidar un modelo de datos adecuado para el caso de uso al que se pretende aplicar el *dataset* generado. Elaborar un mapa lógico de datos que relacione los datos de partida y el resultado deseado.
T-02-02	*Diseño del proceso de limpieza de datos.* (3 puntos) Definición de las operaciones necesarias para alcanzar el resultado final que se persigue. Análisis de las herramientas a utilizar para llevarlas a cabo.
T-02-03	*Implementación del proceso.* (3 puntos) Implementación del componente de procesamiento de datos.

Tabla 2.2: Tareas de la Historia 02.

Tarea	Descripción
T-03-01	*Elección del mecanismo de clasificación de textos.* (1 punto) Elegir un método de clasificación automática de textos que se adecúe a las necesidades semánticas y de escala del proyecto.
T-03-02	*Integración de la herramienta elegida en el sistema.* (1 punto) Diseño del componente de filtrado de datos en base al resultado de la clasificación sobre el *dataset* generado.
T-03-03	*Implementación del componente de clasificación y filtrado.* (2 puntos) Desarrollo del código necesario para la transformación del *dataset* inicial al conjunto de datos apto para su uso en etapa de predicción.

Tabla 2.3: Tareas de la Historia 03.

Tarea	Descripción
T-04-01	*Selección de mecanismo de predicción.* (1 punto) Evaluar y escoger un mecanismo que nos permita llevar a cabo la predicción objetivo utilizando los datos de que disponemos.
T-04-02	*Diseño de la transformación de los datos necesaria.* (2 puntos) Determinar si es necesaria alguna transformación adicional de los datos, así como las operaciones de enriquecimiento requeridas (análisis de sentimiento).
T-04-03	*Implementación del componente de predicción.* (3 puntos) Desarrollar el componente de predicción en base al *dataset* generado en fases anteriores.
T-04-04	*Elaboración de conclusiones.* (1 punto) Analizar los resultados obtenidos y evaluar su calidad.

Tabla 2.4: Tareas de la Historia 04.

Tarea	Descripción
T-05-01	*Elaboración de introducción y contextualización del proyecto.* (2 puntos) Redacción de la introducción al proyecto, junto con los objetivos planteados y la descripción del contexto en que se lleva a cabo.
T-05-02	*Definición del plan de trabajo.* (1 punto) Consolidación del plan de trabajo, incluyendo la descripción de la metodología utilizada y la planificación que se asume para llevar a cabo el proyecto.
T-05-03	*Documentación del proceso de captura.* (1 punto) Documentar el desarrollo de implementación del componente de captura de información de Twitter, describiendo los recursos utilizados en el mismo.
T-05-04	*Documentación del proceso de limpieza de datos.* (2 puntos) Documentar el análisis llevado a cabo, así como las operaciones necesarias para transformar los datos desde su estado en crudo hasta el formato en que puedan ser explotados. Describir los recursos y herramientas utilizados durante esta fase.
T-05-05	*Documentación del mecanismo de clasificación y filtrado.* (2 puntos) Documentar el proceso de desarrollo del componente de clasificación y filtrado de la información obtenida de Twitter.
T-05-06	*Documentación del diseño de la predicción.* (2 puntos) Documentar el proceso de desarrollo del componente de predicción de datos.
T-05-07	*Redacción de conclusiones del proyecto.* (1 punto) Llevar a cabo un proceso de reflexión sobre el trabajo realizado en relación a los objetivos planteados, y realizar un balance de los esfuerzos dedicados.

Tabla 2.5: Tareas de la Historia 05.

En total, se han estimado un total de 39 puntos de historia, a distribuir entre los tres *sprints* previstos. De acuerdo con la complejidad base establecida (las tareas de 1 punto de historia) y nuestra experiencia, consideramos que una equivalencia de 6 horas de trabajo efectivo por cada punto de historia es adecuada para el desarrollo del proyecto. Esto arroja una duración total del proyecto, en horas, de 234, a razón de 39 horas semanales. Es decir, se prevé una dedicación de 6,5 horas diarias, seis días a la semana para completar los trabajos planificados. En esta estimación no se incluyen ni el acto de defensa del proyecto (con una duración de dos horas), ni las entrevistas con los tutores del proyecto. En total, la carga de trabajo esperada se alinea con los requisitos establecidos en la guía docente para el Trabajo Fin de Máster.

Aunque de acuerdo con Scrum los objetivos de cada *sprint* se determinan al comienzo del mismo en función de la marcha del proyecto, se ha hecho una distribución inicial de las tareas como se describe en la Tabla 2.6. Puede consultarse el diagrama de Gantt correspondiente a esta planificación en la Figura 2.1.

Sprint	Inicio	Fin	Tareas			Total puntos
1	22/05	04/06	T-01-01	T-01-02	T-01-03	14
			T-01-04	T-05-01	T-05-02	
			T-05-03	T-02-01		
2	05/06	18/06	T-02-02	T-02-03	T-05-04	12
			T-05-05	T-03-01	T-03-02	
3	19/06	02/07	T-03-03	T-04-01	T-04-02	13
			T-04-03	T-04-04	T-05-06	
			T-05-07			

Tabla 2.6: Calendario previsto al comienzo del proyecto.

2.3. Presupuestos.

A partir de la planificación temporal establecida, podemos elaborar una previsión de los costes que conllevará el desarrollo del proyecto. En esta relación se incluirán también aquellos elementos de terceros que, sin suponer un gasto por ser gratuitos o de libre acceso, han sido utilizados en el ámbito del Trabajo.

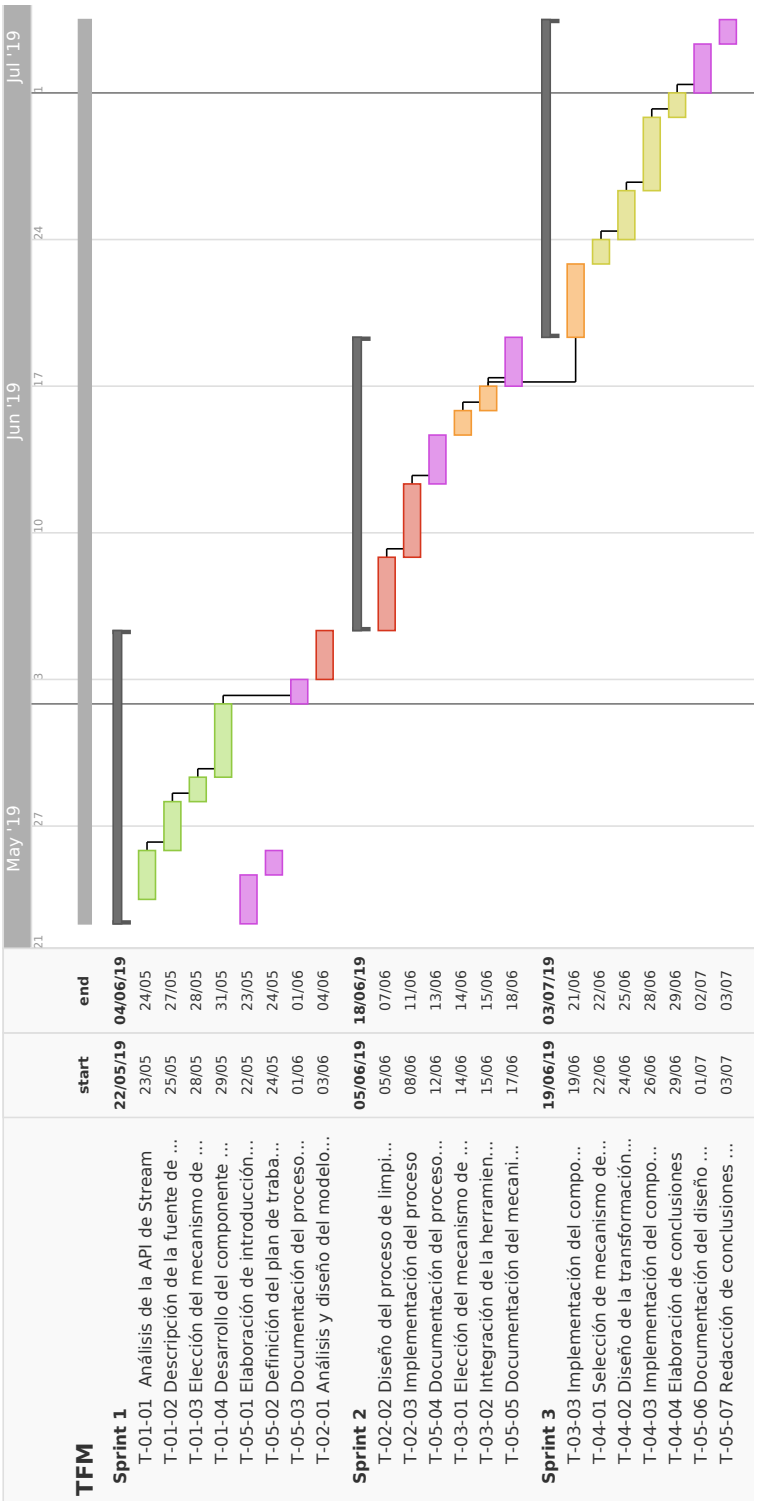


Figura 2.1: Diagrama de Gantt del proyecto.

2.3.1. Recursos técnicos.

Tanto el código fuente generado en el desarrollo del proyecto software como la redacción de la memoria se han llevado a cabo utilizando un ordenador portátil personal (Intel Core i5-4210U 64 bit 1,7-2,4 GHz, 4GB RAM, 500 GB HDD) utilizando herramientas software de libre distribución. Podemos atribuir a un equipo de esta naturaleza (portátil de gama media-baja) una vida útil de unos tres años, por lo que es necesario calcular el coste de amortización del elemento que se corresponda con el tiempo de vida previsto para el proyecto .

El sistema se desplegará en una máquina virtual cedida por la Universidad de Valladolid, cuyo uso no tendrá un coste directo en el proyecto, puesto que el cálculo del coste de amortización resultante de su uso, además de difícil, será posiblemente muy pequeño en comparación con otros componentes de coste en el proyecto dada la naturaleza de este elemento. Las características básicas de este componente son las siguientes: Intel 8-core 64 bit 2 GHz, 10 GB RAM, 1TB HDD.

Asimismo, será imprescindible una conexión a Internet tanto para la conexión con la máquina virtual en que se desplegará el sistema, como la edición y salvado en la nube de los materiales generados durante el desarrollo.

Herramienta	Coste ud.	%	Total
Ordenador portátil personal	495,00 €	3,125	15,48 €
Conexión a Internet	30,00 €	150	45,00 €
Máquina virtual (cedida UVa)			0,00 €
Distribución Anaconda			0,00 €
Notepad++			0,00 €
Overleaf			0,00 €
Dropbox			0,00 €
Gravit Design			0,00 €
Windows 10 Home (incluido en ordenador)			0,00 €
Microsoft Office 2016 (incluido en ordenador)			0,00 €
		Total:	60,48 €

Tabla 2.7: Elementos técnicos y tecnológicos.

2.3.2. Recursos humanos.

A pesar de que el proyecto es unipersonal, el autor deberá asumir diferentes roles con diferentes rangos salariales. En la Tabla 2.8 se indican los

distintos roles, así como la estimación de costes que supondría su contratación. La dedicación semanal es análoga a la habitual en la empresa (40 horas semanales, unas 160 horas al mes), por lo que el coste por hora estimado es el mismo que si la configuración de jornada fuera la habitual de jornada completa de 8 horas, de lunes a viernes.

Rol	Salario mensual	Horas	Coste
Jefe de proyecto	2.500 €	26,00	406,25 €
Analista	2.200 €	133,25	1.832,20 €
Desarrollador Python	1.800 €	94,25	1.060,30 €
		Total:	3298,75 €

Tabla 2.8: Costes derivados de los recursos humanos.

2.4. Balance final del proyecto.

La distribución del trabajo realizados en el durante el desarrollo del proyecto se ha visto afectada por diversos factores, obligándonos a desviarnos, en algunos casos significativamente, de la previsión de tiempos y costes que se llevó a cabo al comienzo del mismo. En esta sección se indican las modificaciones introducidas y sus causas, evaluando en cada caso el impacto que han tenido sobre la organización final del proyecto.

2.4.1. Planificación temporal.

Durante el primer *sprint* se lograron todos los objetivos marcados. Se comenzaron los trabajos abordando las tareas T-05-01 y T-05-02, con el fin de sentar las bases sobre las que llevar a cabo el desarrollo del resto del proyecto, así como para establecer un plan de trabajo que sirviera como referencia tanto al autor como a los tutores. Las tareas correspondientes a la historia H-01 se completaron adecuadamente en tiempo y forma, así como las labores de documentación del trabajo realizado, como se establecía en la tarea T-05-03.

La primera semana del segundo *sprint* transcurrió de acuerdo a las previsiones iniciales, completando con éxito todas las tareas de la Historia 02. Sin embargo, no se pudo completar la tarea T-05-04, que incluía una documentación completa sobre las tareas de dicha historia, por lo que se anotó esta tarea como trabajo pendiente para el siguiente *sprint*.

Al abordar la tercera historia se produjo un imprevisto de gran importancia para el correcto desarrollo del proyecto. La planificación temporal y de esfuerzo para esta historia asumía que dispondríamos de un recurso de clasificación de textos que nos permitiera llevar a cabo esta actividad con una solución de poca complejidad, basada en la realización de consultas a un servicio remoto. No obstante, al acometer la tarea T-03-01 se descubrió que el componente de clasificación de textos del servicio elegido, la *Natural Language* API de la plataforma Google Cloud, no era compatible con el idioma objetivo del proyecto, el español. Tras una búsqueda infructuosa de posibles alternativas que pudieran satisfacer los requisitos marcados para el proyecto, se acordó con los tutores ampliar el alcance del proyecto para dar cabida al desarrollo de una solución propia.

Esto supuso la ampliación de la historia T-03, así como la reestructuración de las tareas que la integran, quedando su configuración como se refleja en la Tabla 2.9. Entre el incremento del alcance de la historia y el retraso que supuso adaptar el proyecto a este imprevisto, resultó imposible cumplir con los objetivos marcados para el segundo *sprint*, y se decidió agregar un *sprint* adicional para abordar las nuevas tareas identificadas y llevar el proyecto al punto en que debía encontrarse antes de comenzar el tercer *sprint*, tal y como se había planificado. En este punto, se optó por retrasar la entrega del proyecto de la convocatoria de julio a la convocatoria extraordinaria de septiembre, considerando que no se podrían lograr todos los objetivos del proyecto a tiempo.

Las tareas relacionadas con el clasificador se llevaron a cabo en este tercer *sprint*, si bien quedó como tarea pendiente completar una documentación adecuada para su inclusión en la memoria del proyecto. Finalmente, durante el cuarto *sprint* se realizaron las tareas previstas relacionadas con la historia H-04. Debido al retraso en la culminación del proceso de documentación, así como la cercanía de las vacaciones de verano, se decidió terminar las tareas de revisión de la documentación en la primera semana de septiembre, en la que el proyecto fue completado con éxito. No se organizó ningún *sprint* para realizar este remate del proyecto, y por tanto no se refleja en la planificación.

Todos los cambios introducidos se han reflejado en el diagrama de Gantt actualizado que se muestra en la Figura 2.2.

2.4.2. Previsión económica.

Debido a la prolongación de los tiempos de desarrollo del proyecto, es necesario actualizar la relación de costes atribuibles al proyecto. En las Tablas 2.10 y 2.11 se incluyen los elementos implicados que han visto incrementado su impacto con respecto a la previsión inicial.

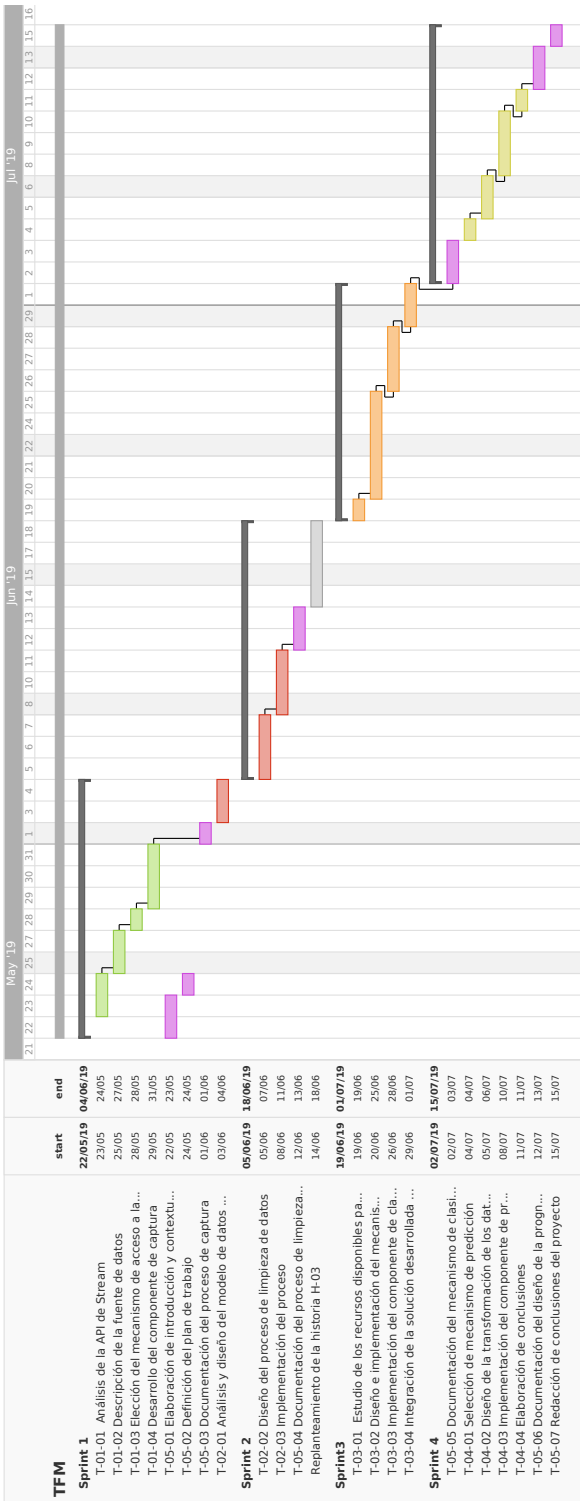


Figura 2.2: Diagrama de Gantt final del proyecto.

Tarea	Descripción
T-03-01	*Estudio de los recursos disponibles para la implementación del clasificador.* (1 punto) Buscar un recurso que permita llevar a cabo la construcción de un sistema de clasificación de *tweets* de acuerdo a su semántica, acorde al criterio necesario para el desarrollo del proyecto (política/no política).
T-03-02	*Diseño e implementación del mecanismo de obtención y tratamiento de datos para su uso en el entrenamiento del clasificador.* (5 puntos) Generación de un mecanismo de extracción de información basado en *web scraping* para suplir la falta de recursos válidos en español para satisfacer los objetivos marcados.
T-03-03	*Construcción del modelo de clasificación.* (3 puntos) Selección, configuración, entrenamiento y evaluación del modelo más adecuado para la implementación del clasificador en el contexto del proyecto.
T-03-04	*Integración de la solución desarrollada en el sistema.* (2 puntos) Integración del clasificador binario desarrollado en el sistema, de manera que se pueda utilizar para realizar las tareas de filtrado requeridas sobre los datos capturados.

Tabla 2.9: Tareas de la Historia 03 (actualizadas).

Herramienta	Coste ud.	%	Total
Ordenador portátil personal	495,00 €	4,7	23,20 €
Conexión a Internet	30,00 €	225	67,50 €
		Total:	90,70 €

Tabla 2.10: Costes finales de los elementos técnicos.

Rol	Salario mensual	Horas	Coste
Jefe de proyecto	2500 €	26	406,25 €
Analista	2200 €	153	2.103,75 €
Desarrollador Python	1800 €	107,25	1.206,60 €
		Total:	3.716,60 €

Tabla 2.11: Distribución de costes entre los diferentes roles

Capítulo 3

Contexto de desarrollo.

3.1. Conceptos básicos.

Antes de identificar el alcance y objetivos del proyecto, es necesario caracterizar el contexto en el que se enmarca el mismo, pues nos ayudará a motivar el interés de la propuesta que se va a elaborar y fundamentar los retos que deben ser afrontados para llevarla a término.

3.1.1. Big Data.

En la introducción, hemos etiquetado el problema de trabajar con la API de *Stream* de Twitter como un problema Big Data, dado el volumen y la diversidad de los datos que esta contiene. Sin embargo, más allá de esta intuición, es necesario aportar una definición para el concepto de Big Data para comprender todas sus implicaciones.

En [11] se lleva a cabo una revisión de de las múltiples definiciones que ha recibido este fenómeno a lo largo de los años, tratando de llegar a una definición única que pueda abarcar todos los matices que lo caracterizan. La conclusión de los autores es que el concepto de Big Data puede resumirse en la siguente definición: "El Big Data representa los recursos de información caracterizados por tener un gran volumen, generarse a una gran velocidad y presentar una gran variedad, y que por tanto requieren de tecnologías y métodos analíticos específicos específicos para su transformación en valor."[1]

De esta manera, encontramos que el *Big Data* no es sencillamente un gran volumen de datos, sino que son datos caracterizados por las conocidas como

[1]Original: "*Big Data represents the Information assets characterized by such a High Volume, Velocity and Variety to require specific Technology and Analytical Methods for its transformation into Value*".

"5 Vs del Big Data" [12]: Volumen, Velocidad, Variedad, Veracidad (o calidad) y Valor. En el caso que nos ocupa, las tres primeras son características obvias de la fuente de datos que queremos abordar. El valor es, precisamente, el objetivo que buscamos al emprender este proyecto, y es poder obtener información de calidad (veraz) con una finalidad concreta. En este caso, el caso de uso elegido es la predicción electoral (*electoral forecasting*), pero la variedad presente en la fuente permitiría extrapolar el trabajo realizado a otras áreas.

Para extraer este valor, y de acuerdo con la definición, es imprescindible contar con técnicas analíticas y tecnologías específicas que nos permitan abordar un problema de esta complejidad. A lo largo de este proyecto se hará mención de algunas de estas herramientas, aunque por razón de alcance del proyecto y recursos disponibles no llegaremos a trabajar en un entorno Big Data "verdadero". En su lugar, exploraremos, con una muestra suficientemente grande de un problema Big Data real, pero asumible para un computador personal, una propuesta de aproximación que, más adelante, pueda ser adaptada utilizando tecnologías Big Data.

3.1.2. Twitter.

Twitter es una red social de propósito general cuyo funcionamiento comenzó en julio de 2006. Se trata de una plataforma con vocación de agilidad y tiempo real, caracterizada por la corta longitud de sus mensajes: los ya icónicos 140 caracteres (aunque en 2017 se incrementó este límite a 280 [13]). Este hecho marcó sus inicios como plataforma de *microblogging*, en contraposición al auge, en aquel momento, de la comunicación digital en formato blog.

Los blogs estaban formados por artículos, que se publicaban con una cierta periodicidad. Cada uno de estos artículos solía estar formado por un texto largo junto con enlaces y recursos multimedia, por lo que requerían una importante labor de redacción, revisión y composición. Twitter rompió con esta tendencia introduciendo el concepto de *tweet*, el mensaje típico de la plataforma caracterizado por el límite impuesto a su longitud. Dado su pequeño tamaño, este formato fomentaba la inmediatez y la condensación de los contenidos, y facilitaba enormemente la accesibilidad de cualquier usuario tanto a la publicación de contenidos como al consumo de los mismos.

Los elementos básicos de Twitter son los usuarios y los propios mensajes, aunque existen otras entidades relevantes en la idiosincrasia de la red social como las tendencias o *trending topics*, los *hashtags* o las listas de usuarios. Los usuarios se identifican por un nombre único (*screen name*), que se utiliza,

precedido por una arroba (@), para apelar directamente al usuario dentro de un mensaje. Adicionalmente, los usuarios tienen un nombre descriptivo y no necesariamente único que se muestra en su perfil y sus mensajes. Los tweets han ido evolucionando y actualmente pueden contener hiperenlaces, elementos multimedia de distinta índole (fotos, vídeos, recursos embebidos), encuestas, etcétera. Además, han pasado de ser entidades únicas y aisladas a poder formar parte de estructuras más complejas: conversaciones, que son concatenaciones de mensajes entre varios usuarios con menciones explícitas a cada uno de ellos, y, más recientemente, *hilos* de tweets, que agrupan varios mensajes emitidos por el mismo autor en sucesión, estructurándose en forma de respuestas a sus propios mensajes.

Las tendencias agrupan automáticamente mensajes de naturaleza similar en base a diferentes criterios, fundamentalmente la concentración temporal y la repetición de cadenas de caracteres. Estas cadenas pueden ser series de palabras que se repiten a menudo, o *hashtags* utilizados frecuentemente. Los *hashtags* son concatenaciones de palabras sin espacios, precedidas de una almohadilla (#), que los usuarios utilizan para etiquetar los tweets como parte del cuerpo del mensaje.

Las interacciones entre usuarios se materializan de dos formas. La relación directa entre los usuarios se define mediante la figura del seguidor o *follower*, que permite a un usuario recibir los mensajes de los usuarios a los que sigue. Estas "relaciones de amistad" son unidireccionales y no necesariamente recíprocas, a diferencia de otras redes sociales. De hecho, no requiere una acción de aceptación por parte del usuario seguido. También pueden utilizar las menciones anteriormente descritas para apelar directamente a otro usuario, con el que puede tener relación o no, dentro de un mensaje. Un usuario también puede interactuar con los contenidos de otros usuarios de diferentes formas: respondiendo a ellos mediante un nuevo tweet, marcándolos como "favoritos" o compartiéndolo con sus propios seguidores mediante la acción de *retweet* (re-publicación de un mensaje ajeno en el propio tablón de contenido del usuario).

La masividad de este servicio, así como la disponibilidad de una API pública y gratuita para el acceso a sus contenidos tanto en *batch* como en tiempo real (*streaming*) han motivado una enorme actividad científica en torno a esta plataforma. La variedad de sus usuarios, su seguimiento en tiempo real de la actualidad y la riqueza tanto de los contenidos en sí como de los metadatos que se les asocian pueden ser de gran utilidad para la realización de estudios de toda índole, como análisis sociológicos, desarrollo de sistemas de aprendizaje automático, predicciones de eventos (sociales, económicos, etcétera) o seguimiento en tiempo real de sucesos por todo el

mundo.

El acceso a Twitter es público y, dentro de unos límites, gratuito, bien a través de su API (en su versión 5.0 desde febrero de 2019), bien por medio de cualquiera de los *wrappers* disponibles para la misma[14]. Estos *wrappers* son implementaciones software que facilitan el acceso a la API en múltiples entornos; por ejemplo, en diferentes lenguajes de programación:

- En Java, la librería de acceso a la API de Twitter más conocida y madura es Twitter4J[2], cuya versión 1.0 vio la luz en 2007 y actualmente se encuentra en su versión 4.0.8. Permite una fácil interacción con la API en entornos Java (incluyendo Android), gestionando automáticamente la autenticación o el mantenimiento de la conexión en *streaming*, entre otras funciones.
- En Python hay una gran variedad de alternativas, entre las que se cuentan tweepy[3], python-twitter[4] o twython[5], entre otros. Proporcionan similar funcionalidad a Twitter4J en un entorno de desarrollo en Python.

Parte de la API de Twitter se describe en la Sección 5.4, donde se indicará también la funcionalidad equivalente en la librería utilizada, Tweepy.

3.1.3. Inteligencia de negocio.

Acorde con la temática de la titulación, sería posible encuadrar el presente proyecto en el ámbito de la Inteligencia de Negocio (*Business Intelligence*). Partamos de una definición formal de este concepto [15]: "Un sistema de Inteligencia de Negocio combina la adquisición de datos, el almacenamiento de datos y la gestión del conocimiento con herramientas analíticas para presentar la información compleja y competitiva a los responsables de la planificación y la toma de decisiones en la empresa" [6].

Si en el caso del Big Data se hacía especial incidencia en el volumen y la variedad de los datos, en la inteligencia de negocio primarán la veracidad y el valor, especialmente el valor que la información puede proporcionar en un ámbito empresarial. En consecuencia, existen dos conceptos clave en cualquier aplicación de este concepto:

[2] `http://twitter4j.org/en/` (visitado: 01/06/19)

[3] `https://github.com/tweepy/tweepy` (visitado: 01/06/19)

[4] `https://github.com/bear/python-twitter` (visitado: 01/06/19)

[5] `https://github.com/ryanmcgrath/twython` (visitado: 01/06/19)

[6] Original: "*BI systems combine data gathering, data storage, and knowledge management with analytical tools to present complex internal and competitive information to planners and decision makers.*"

- Los procesos de Extracción, Transformación y Carga de datos (*Extraction, Transformation and Load*, ETL): En general, la forma en que se presentan los datos tras su generación no es adecuada para satisfacer las necesidades del negocio, sino que deben ser sometidos a diferentes operaciones de limpieza, filtrado y enriquecimiento que extraigan la información significativa de los datos, y la suministren en una forma refinada y apta para su consumo inmediato. Los procesos ETL definen el flujo de trabajo necesario para extraer los datos de la fuente original, aplicar las operaciones de transformación adecuadas y su despliegue para su utilización inmediata en un caso de uso concreto.
- Los procesos de visualización de datos: La información más relevante en cada instante debe ser presentada de manera que aquellos que vayan a consumirla puedan hacerlo de la manera más sencilla y rápida posible. Para ello, recursos como la representación gráfica de los datos o la interactividad prestan un gran servicio en cualquier entorno empresarial.

Aunque este proyecto no sirve a ningún propósito empresarial o de negocio, sí contiene múltiples rasgos que permiten tender puentes entre el mismo y la Inteligencia de Negocio. Nuestro objetivo es transformar datos de Twitter, por lo demás caóticos, en información que sea fácilmente interpretable por medio de gráficas de predicción en base a tendencias. Por tanto, recurriremos al concepto de proceso ETL para organizar nuestro flujo de trabajo.

3.1.4. Aprendizaje automático.

El aprendizaje automático (*machine learning*) es una forma de procesamiento de datos y extracción de información que se ha postulado como una de las principales vías para el procesamiento de datos en entornos Big Data [16]. El aprendizaje automático consiste en la identificación de patrones ocultos en los datos, los cuales podremos utilizar para extrapolar la información que contienen y generar nuevo conocimiento.

Existen diferentes enfoques para abordar el *machine learning*, aunque los principales son únicamente dos: mediante aprendizaje supervisado, o mediante aprendizaje no supervisado. A grandes rasgos, el aprendizaje supervisado parte de un conjunto de ejemplos *etiquetados*, es decir, previamente identificados. Por ejemplo, para un problema de clasificación en categorías, un conjunto de datos etiquetados consistirá en ejemplos, definidos por una serie de características (*features*), y con una clase o categoría asignada. A partir de esos ejemplos podemos identificar las características que mejor

describen cada clase, y así clasificar cada nuevo ejemplo desconocido con una de las clases. Es apropiado, por tanto, cuando las clases que pueden presentar los datos son conocidas.

En el aprendizaje no supervisado se carece de etiquetas para los ejemplos, pero es posible realizar análisis de los ejemplos en base a las características que presentan. Un ejemplo de aprendizaje sin supervisión es el *clustering* o agrupamiento, donde los ejemplos con características similares son reunidos en *clusters* por medio de la aplicación de un algoritmo. En este caso, las clases no son conocidas, sino que se descubren a partir del proceso.

En este proyecto se hará uso del primer enfoque, aunque el aprendizaje no supervisado ha sido aplicado con éxito en el ámbito de la clasificación de textos. No obstante, dado que contamos con un conjunto de clases predefinido (básicamente, textos que tratan de política y textos que tratan de cualquier otro tema), podemos hacer uso de técnicas de aprendizaje supervisado para hacer posibles nuestros objetivos respecto al filtrado automático de información.

3.1.4.1. Predicción.

Un ejemplo de caso de uso concreto de aprendizaje automático es la realización de predicciones. En este caso, los patrones a descubrir son tendencias en los datos que nos permitan pronosticar los valores que presentarán en unas circunstancias diferentes a las conocidas; por ejemplo, en un momento distinto, en unas condiciones de temperatura diferentes, etcétera. De esta forma, es posible generar información completamente nueva (la predicción) en base a datos existentes y conocidos.

Durante el proyecto se harán uso de técnicas de predicción para obtener, en base a datos capturados en tiempo real, proyecciones de la intención de voto de los votantes para el evento electoral elegido, en un ejercicio similar al que ya se hace, aunque por medios manuales, en el ámbito de la demoscopia.

3.2. Estado del arte.

Las redes sociales han llamado la atención de los investigadores desde el comienzo de su andadura. Una fuente de datos pública, en constante movimiento y evolución, y en la que se vierten contenidos de toda naturaleza, ofrece un enorme abanico de posibilidades para abordar otros tantos casos de uso. En este aspecto, la cuestión nunca es el “qué”, sino el “cómo”. Basta con una simple búsqueda en cualquier motor de indexación de artículos

científicos para descubrir el volumen de investigación que se ha dedicado a varios de los ámbitos que visitaremos dentro de este proyecto.

En particular, la explotación de Twitter como fuente de información para la predicción de resultados electorales se remonta al año 2010, y desde entonces se han multiplicado las propuestas que tratan de proporcionar una solución adecuada. En todas ellas se constata la dificultad de tratar con la información presente en Twitter: la corta longitud de los mensajes, la frecuencia de errores de escritura y la cantidad de ruido presente en los datos dificultan cualquier tipo de procesamiento automático, sea recuperación de información (*Information Retrieval*), análisis de sentimiento (*Sentiment Analysis*), modelado temático (*Topic Modeling*)... Existen multitud de técnicas aplicables a textos normales, pero que suelen dejar de funcionar adecuadamente al aplicarlas sobre textos cortos.

No obstante, algunas de las técnicas han demostrado proporcionar una eficacia reseñable. En el ámbito del aprendizaje no supervisado, es común el uso de LDA (*Latent Dirichlet Allocation*) [17], un algoritmo no supervisado de *clustering* en base a las características semánticas implícitas (*latentes*). Este algoritmo asume que las palabras utilizadas en un texto guardan relación entre sí, pero, al mismo tiempo, un texto contendrá en sí cierta diversidad de temas. A diferencia de los enfoques basados en TF-IDF (un método de análisis de frecuencias en el que se evalúa la relevancia de cada término de un *corpus* documental con respecto al *corpus* completo), LDA aprovecha la estructura interna de los documentos (coocurrencia de términos, frecuencia intradocumental, etcétera) para caracterizar los contenidos del texto y proporcionar una distribución de probabilidades con los temas más probables.

En otros enfoques se ha optado por tratar de subsanar las deficiencias que presentan los *tweets* en lugar de adaptar las técnicas para lidiar con ellas. Por ejemplo, una técnica habitual para compensar la corta longitud de los mensajes es el *pooling*: en lugar de tratar de procesar o clasificar *tweet* por *tweet*, se reúne una muestra de ellos para que, en conjunto, ofrezcan más información que cada uno por separado. En [18] proponen varios criterios para hacer *pooling*: por autor, por tendencia o *trending topic*, por cercanía temporal, por *hashtag*... Una vez agrupados, aplican LDA sobre los grupos generados hasta obtener un número pre-determinado de *clusters*, y evalúan el nivel de agrupamiento conseguido tras utilizar las diferentes políticas de *pooling*. De acuerdo con sus resultados, el agrupamiento por *hashtag* proporciona mejores resultados que el resto, proporcionando conjuntos más cohesionados y diferenciados. Aunque nuestro enfoque pasa por utilizar un clasificador supervisado y en el artículo se utiliza una técnica sin supervisión, sí haremos uso de técnicas de *pooling* para tratar de incrementar la confianza

de las clasificaciones de tendencias, por lo que será importante tomar en consideración el papel de los *hashtags* durante el proceso.

En [19] también utilizan LDA, pero proponen una forma alternativa de llevar a cabo el *pooling*: utilizar palabras clave. Este enfoque es más cercano al nuestro en el sentido de que utiliza palabras clave procedentes de los *tweets* (nuestra intención, tras utilizar las tendencias como filtro inicial del *stream*, es caracterizar el partido político al que un mensaje es afín en función de los *trending topics* a los que se asocie). Los resultados reportados son superiores a los ofrecidos por el artículo anterior, lo que sugiere que el uso de palabras clave, además de los *hashtags*, podría proporcionar una mayor cobertura de los datos relevantes, de manera que se recupere un número mayor de resultados.

En nuestro país, este tipo de aproximaciones ha conseguido cierta atención. [20] y [21] llevan a cabo diferentes estudios de eventos electorales en nuestro país, algunos de ellos únicos como las dos convocatorias de elecciones generales en 2015 y 2016 por la repetición de elecciones, que les permitió a los autores del segundo artículo llevar a cabo un interesante estudio comparativo. En general, todas estas aproximaciones hacen uso de *hashtags* manualmente identificados para reunir información relevante, y después aplican análisis de sentimiento para caracterizar bien los mensajes, bien a los usuarios que los emiten. Otros buscan caracterizar a los usuarios en función del comportamiento que muestran en las redes durante estos eventos, como [22] o [23], explotando algunos de los datos proporcionados por Twitter,como los conceptos de *retweet* y menciones de usuario, lo que demuestra el volumen de información disponible en Twitter que puede utilizarse para caracterizar tendencias, describir relaciones complejas, etcétera.

Finalmente, ya en un ámbito más general, en [24] llevan a cabo una revisión de las técnicas existentes para la realización de análisis predictivo a partir de datos recogidos de redes sociales. En la misma se proponen algoritmos de aprendizaje automático como SVM o Naive Bayes, que tendremos la oportunidad de probar a lo largo del proyecto.

Parte II

Desarrollo del proyecto.

Capítulo 4

Análisis y diseño del sistema.

En este capítulo se llevará a cabo el análisis técnico del sistema, donde identificaremos y describiremos los diferentes requisitos que dirigirán el desarrollo del mismo. En primer lugar se determinarán las características principales del sistema, que posteriormente detallaremos mediante los correspondientes requisitos funcionales. Identificaremos también las diferentes restricciones que deberán tenerse en cuenta durante el desarrollo del proyecto. Finalmente, se enunciarán algunos detalles de diseño en relación a la arquitectura del sistema y la organización del sistema de almacenamiento subyacente.

4.1. Árbol de características

En la Figura 4.1 se muestra el árbol de características del sistema [25], cuyos elementos se listan, con su identificador asociado, en la Tabla 4.1. Cada característica tendrá asociado un identificador único para facilitar su trazabilidad a lo largo del proyecto, con la forma C-XX. Las sub-características, a su vez, se identificarán con identificadores derivados de la característica que las contiene: C-XX-YY.

El sistema tendrá cuatro componentes principales, que se integrarán en las diferentes fases del flujo de trabajo completo que se definirá más adelante. El desarrollo de las características C-01, C-03 y C-04 se discutirá en los Capítulos 5, 6 y 7, respectivamente. La característica C-02 contendrá funcionalidades transversales, utilizadas en diferentes puntos del *pipeline* de procesamiento, y se irá describiendo a medida que dichas funcionalidades hagan su aparición.

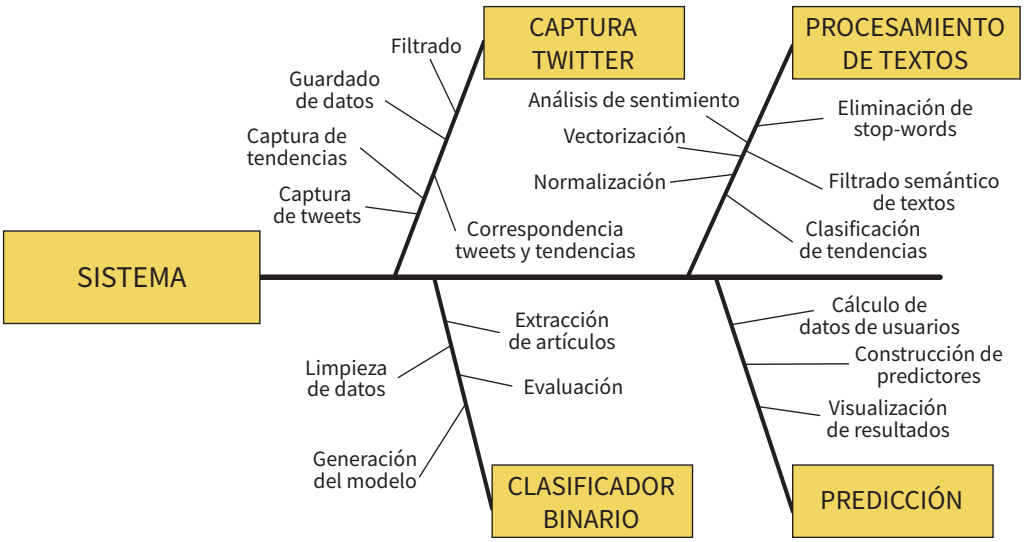


Figura 4.1: Árbol de características del sistema.

ID	Nombre
C-01	**Captura de datos en Twitter**
C-01-01	Captura de tendencias
C-01-02	Captura de *tweets*
C-01-03	Identificación de correspondencia *tweet*-tendencia
C-01-04	Filtrado de información relevante
C-01-05	Almacenamiento de los datos
C-02	**Procesamiento de textos en lenguaje natural**
C-02-01	Eliminación de *stop-words*
C-02-02	Normalización de textos
C-02-03	Vectorización de textos
C-02-04	Análisis de sentimiento
C-02-05	Filtrado semántico de textos
C-02-06	Clasificación de tendencias
C-03	**Clasificador binario**
C-03-01	Extracción de artículos de prensa
C-03-02	Construcción de *datasets*
C-03-03	Generación de modelo de clasificación
C-03-04	Evaluación del modelo
C-04	**Predicción**
C-04-01	Cálculo de datos de usuarios.
C-04-02	Construcción de predictores
C-04-03	Visualización de resultados

Tabla 4.1: Tabla de características del sistema.

4.2. Restricciones.

En esta sección se listan las restricciones que será necesario afrontar durante el desarrollo del proyecto, y que motivarán ciertas decisiones de diseño e implementación. Principalmente, se corresponden con limitaciones de tipo técnico o tecnológico, como el número de peticiones máximas a los servicios remotos para evitar la saturación de los servidores o el cese del servicio, o las características del sistema en que se desplegará el producto desarrollado. Asignaremos un identificador único a cada restricción con el siguiente formato: RE-XX.

ID	Restricción
RE-01	El sistema no podrá refrescar la lista de tendencias con frecuencia inferior a 5 minutos.
RE-02	El sistema deberá respetar la frecuencia máxima de reconexiones a la API de Twitter, así como los tiempos mínimos de espera tras cada intento de reconexión (ver Subsección 5.1.1).
RE-03	El sistema deberá respetar la tasa máxima de peticiones a la API de Natural Language de Google Cloud (ver Sección 4.5).
RE-04	El sistema no podrá actualizar la captura del *stream* de Twitter, debiendo cerrar la actual y crear una nueva con los parámetros actualizados.
RE-05	El sistema debe ser capaz de procesar ficheros de, a lo sumo, 2GB en un entorno con hasta 10GB de memoria disponibles.
RE-06	Se utilizará Python y sus recursos como lenguaje de programación para agilizar el desarrollo.
RE-07	El sistema deberá realizar las peticiones de páginas de artículos a una tasa máxima de un artículo por segundo para no afectar al servicio del servidor web del medio.

Tabla 4.2: Restricciones del proyecto.

4.3. Requisitos funcionales.

En esta sección se especifica la funcionalidad que debe presentar el sistema desarrollado para implementar con éxito las características descritas previamente. Se consolidará esta funcionalidad en forma de requisitos funcionales, que caracterizarán, de manera breve y concreta, las operaciones individuales necesarias para asegurar, en su conjunto, el funcionamiento del sistema.

Agruparemos los requisitos funcionales atendiendo a la característica en la que se enmarca la funcionalidad que describen. En las Tablas 4.3 a 4.6 se incluye el listado completo de los requisitos identificados para este proyecto. Cada requisito funcional se identificará mediante un código con las iniciales RF-XX.

ID	Requisito
RF-01	El sistema debe establecer conexiones a la API *Search* de Twitter.
RF-02	El sistema debe establecer y mantener conexiones a la API de *Streaming* de Twitter.
RF-03	El sistema debe obtener la lista de *trending topics* en España de forma periódica, de acuerdo con la configuración del usuario (respetando el límite inferior indicado en la RE-01).
RF-04	El sistema debe activar el proceso de captura del *stream* de *tweets* aplicando un filtro basado en la lista de *trending topics*.
RF-05	El sistema debe guardar la información de *trending topics* en el almacenamiento persistente.
RF-06	El sistema debe guardar el flujo de *tweets* capturados (íntegros y sin tratar) en el almacenamiento persistente.
RF-07	El sistema debe recuperarse de cualquier error en la conexión con la API.

Tabla 4.3: Requisitos funcionales de la Característica 01 - Captura de Twitter.

4.4. Requisitos de usuario

4.5. Interfaces externas.

4.5.1. API de Twitter.

Dado su carácter central en el proyecto y su papel como fuente de datos del proceso de refinamiento de datos, la API de Twitter se describe en la sección correspondiente al análisis de las fuentes de datos del proceso ETL (ver Capítulo 5, Subsección 5.1.1).

4.5.2. API de Google Cloud NLP.

Google Cloud es una plataforma en la nube que provee de una larga lista de servicios para el desarrollo y despliegue de aplicaciones en la infraestructura de Google, asegurando en gran medida la disponibilidad y escalabilidad

ID	Requisito
RF-08	El sistema debe ser capaz de interpretar los datos proporcionados por la API de Twitter y transformarlos a estructuras de datos que soporten operaciones masivas sobre datos y funciones de agregación.
RF-09	El sistema debe generar un identificador único para cada entidad *trending topic* detectada.
RF-10	El sistema debe guardar los datos de *trending topics* en el almacenamiento persistente tras su transformación.
RF-11	El sistema debe descartar los *tweets* en otro idioma distinto del español.
RF-12	El sistema debe aplicar las operaciones de procesamiento de lenguaje natural necesarias para normalizar los textos de los *tweets*.
RF-13	El sistema debe determinar la correspondencia entre los *tweets* y los *trending topics* capturados.
RF-14	El sistema debe construir una muestra de textos para cada elemento de las listas de *trending topics*.
RF-15	El sistema debe enriquecer los datos de *tweets* con la información de tendencias.
RF-16	El sistema debe vectorizar los textos que forman la muestra de cada *trending topic* de acuerdo al vocabulario generado en el RF-29.
RF-17	El sistema debe clasificar los textos de las muestras utilizando el clasificador generado en la C-03-03.
RF-18	El sistema debe clasificar los *trending topics* como política o no política en base a las muestras recogidas.
RF-19	El sistema debe filtrar los *tweets* correpondientes a *trending topics* sobre política.
RF-20	El sistema debe aplicar las operaciones de procesamiento de datos de Twitter para un período determinado de tiempo.
RF-21	El sistema debe aplicar las operaciones de clasificación de información de Twitter para un período determinado de tiempo.
RF-22	El sistema debe guardar los datos de *tweets* filtrados en el almacenamiento persistente.

Tabla 4.4: Requisitos funcionales de la Característica 02 - Transformación de datos de Twitter.

de las operaciones. El catálogo de servicios proporcionados en la plataforma es muy amplio, abarcando ámbitos tan variados como la computación y el almacenamiento en la nube, la gestión de bases de datos en diferentes motores, la gestión de Big Data o el aprendizaje automático. En el marco de este proyecto utilizaremos un servicio ubicado en la sección de *Machine Learning*: Natural Language API, que nos permitirá implementar un caso

ID	Requisito
RF-23	El sistema debe extraer las URL de los artículos publicados en un rango de fechas de los periódicos considerados.
RF-24	El sistema debe poder capturar una copia del código fuente de la página.
RF-25	El sistema debe interactuar con la página web para cargar la información de comentarios.
RF-26	El sistema debe clasificar en política y no política los comentarios a partir de los metadatos del artículo en que se publicaron.
RF-27	El sistema debe aplicar sobre los comentarios las operaciones de procesamiento de lenguaje natural previstas en el RF-12.
RF-28	El sistema debe generar dos conjuntos de datos disjuntos a partir del construido para propósitos de entrenamiento y evaluación de modelos.
RF-29	El sistema debe generar un vocabulario apto para la vectorización de textos a partir de los comentarios capturados.
RF-30	El sistema debe vectorizar los comentarios de acuerdo al vocabulario generado.
RF-31	El sistema debe entrenar y validar diversos modelos de clasificación para caracterizar el comportamiento de los mismos con el conjunto de datos.
RF-32	El sistema debe evaluar distintos parámetros de configuración del algoritmo de clasificación binaria para su adecuación a los datos utilizados.
RF-33	El sistema debe entrenar y evaluar el modelo de clasificación binaria generado.
RF-34	El sistema debe exportar el vocabulario generado para su utilización posterior.
RF-35	El sistema debe exportar el modelo generado para su utilización posterior.

Tabla 4.5: Requisitos funcionales de la Característica 03 - Generación del modelo de clasificación binaria.

de uso habitual en el procesamiento de lenguaje natural, como es el análisis de sentimiento de textos.

Inicialmente, se planteó aprovechar las capacidades de este servicio para la clasificación de textos, el cual debería haber dado soporte a las operaciones de caracterización de *trending topics* en nuestro flujo de trabajo, pero este servicio en particular solo es compatible con el idioma inglés. Dado que nuestro foco de interés es el español, este hecho lo descartaba como opción válida y nos obligó a buscar alternativas, como se explicará después.

ID	Requisito
RF-36	El sistema debe cargar en memoria los datos de *tweets* procesados para su transformación final.
RF-37	El sistema debe extraer la información de los usuarios de la información de *tweets* disponibles para un período de tiempo dado.
RF-38	El sistema debe entrenar diferentes modelos de regresión utilizando los datos generados.
RF-39	El sistema debe proporcionar un medio de visualización y exploración configurable de los datos generados.
RF-40	El sistema debe poder persistir los datos de usuarios generados para su reutilización.

Tabla 4.6: Requisitos funcionales de la Característica 04 - Mecanismo de predicción de datos.

ID	Requisito
RU-01	El usuario podrá fijar el período de refresco de la lista de tendencias.
RU-02	El usuario podrá indicar la fecha o fechas de las que deben obtenerse artículos de periódicos.
RU-03	El usuario podrá indicar la fecha o fechas en las que los datos de Twitter deben procesarse.
RU-04	El usuario podrá indicar la fecha o fechas cuyos *tweets* deben filtrarse de acuerdo a sus *trending topics*.
RU-05	El usuario podrá fijar el umbral exigido en las muestras de *tweets* para caracterizar las tendencias.
RU-06	El usuario podrá indicar el período de tiempo utilizado para entrenar los modelos de regresión en el componente de predicción.
RU-07	El usuario podrá determinar el modelo de regresión aplicado para la generación de la visualización.
RU-08	El usuario podrá indicar los parámetros de configuración más relevantes de los modelos de regresión generados.
RU-09	El usuario podrá indicar el número de días para los que debe generarse y visualizarse una predicción.

Tabla 4.7: Requisitos de usuario.

Dada la amplitud de la API de Google Cloud, nos ceñiremos exclusivamente a la funcionalidad que explotamos en el ámbito del Trabajo. En primer lugar, es necesario disponer de una cuenta de Google e ingresar en Google Cloud para crear un proyecto. Una vez creado, será necesario activar las diferentes APIs que se utilizarán en el mismo para poder acceder a sus

Restricción	Valor
Número de peticiones por minuto	600
Número de peticiones diarias	800.000
Límite gratuito de peticiones mensuales	5.000

Tabla 4.8: Restricciones aplicables de la Natural Language API.

recursos. Google Cloud no dispone de un plan de uso gratuito y libre como tal: es necesario aportar información de pago para activar el uso de las diferentes APIs (como una tarjeta de crédito activa). Sin embargo, sí tiene cuotas de uso gratuito[1], y además ofrece una cierta cantidad de crédito gratuita (265 euros, el equivalente a 300$) a utilizar durante el primer año. Previsiblemente, la cuota gratuita y una pequeña parte del crédito otorgado serán más que suficientes para cubrir las necesidades del proyecto, por lo que el uso de esta plataforma no tendrá ningún impacto en los costes del mismo. Las restricciones impuestas por la API se listan en la Tabla 4.8.

La API dispone de una librería cliente implementada en Python[2] (además de otros lenguajes), a través de la cual accederemos a los servicios de la API por medio de las credenciales correspondientes. Estas credenciales se obtienen a través del *Dashboard* de la API para un proyecto en concreto a través de la consola. Se ha creado un proyecto denominado *TweetMonitoring*, para que el que se ha habilitado únicamente la *Natural Language* API (ver Figura 4.2).

La *Natural Language* API proporciona diversas funcionalidades [26], como la clasificación de textos, el análisis de entidades o el análisis de sentimiento. Para el acceso a esta última, la librería cliente suministrada dispone de la función *analyze_sentiment*, que genera, para un texto dado, dos valores para caracterizar el sentimiento contenido en el mismo (ver Tabla 4.9).

La petición debe modelarse como *Document*, una estructura de datos propia de la librería que, además del texto a analizar, puede contener información sobre la codificación del texto, el tipo de documento suministrado (HTML o texto plano) y el idioma del texto (en este caso, '`es`'). Si no se especifica, el idioma es detectado automáticamente.

[1] `https://cloud.google.com/natural-language/pricing?hl=es` (visitado: 24/05/19)

[2] `https://cloud.google.com/natural-language/docs/reference/libraries` (visitado: 24/05/19)

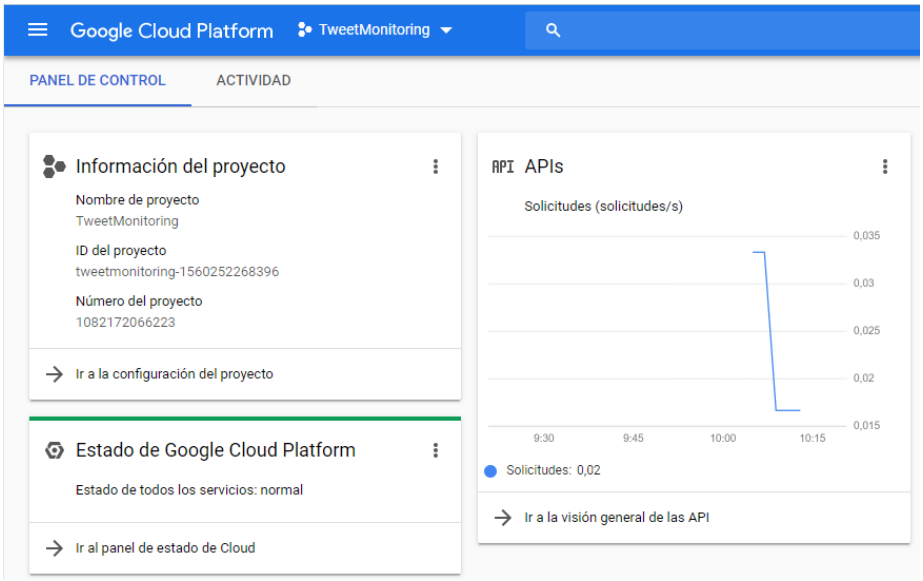


Figura 4.2: Captura de la consola de Google Cloud.

Nombre	Rango	Descripción
Polaridad (*score*)	$[-1, 1]$	Indica la naturaleza positiva o negativa del sentimiento imbuido en el texto, asignando valores menores que 0 para mensajes de contenido negativo, y mayores que 0 si el contenido es positivo.
Intensidad (*magnitude*)	$(0, +\infty)$	Indica el nivel de presencia de sentimiento en el mensaje. Su valor agrega las intensidades de todas las características encontradas en el texto que expresan algún tipo de sentimiento, por lo que es indicativa de la fuerza de la carga de sentimiento de texto completo.

Tabla 4.9: Respuesta de la API para el análisis de sentimiento.

4.6. Arquitectura lógica.

En la Figura 4.3 se describe la estructura lógica de la aplicación, compuesta de los siguientes elementos:

- Fuentes de datos externas de las que el sistema extraerá datos en bruto que alimentarán la funcionalidad prestada por el sistema.
- Interfaces externas con las que interactuará el sistema para obtener un servicio concreto.
- Almacenamiento persistente (en amarillo), donde se almacenan los

productos iniciales, intermedios y finales del procesamiento del sistema.

- Lógica de la aplicación (en gris). Se divide en tres módulos, que se desarrollarán en los capítulos a continuación.

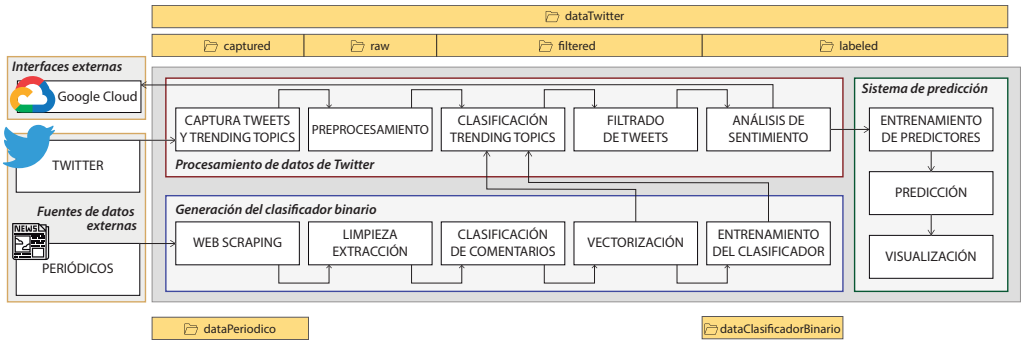


Figura 4.3: Flujo de trabajo de los componentes del sistema.

4.7. Descripción del almacenamiento.

En la versión actual del proyecto se utilizará un espacio de almacenamiento tradicional, sobre el que definiremos una estructura de directorios adecuada para diferenciar los datos que manejaremos de acuerdo a su naturaleza. Utilizaremos dos tipos de directorios, todos ellos ubicados en el directorio base del proyecto.

4.7.1. Directorios de datos.

Los directorios de datos albergan todos los conjuntos de datos (iniciales, intermedios o totalmente refinados) que se generarán a través del proceso de transformación diseñado, así como los productos generados durante la generación de los modelos de aprendizaje automático (modelos serializados para su reutilización, conjuntos de entrenamiento y evaluación, etcétera). En total, definiremos tres directorios de datos:

- Directorio *dataTwitter*: Contiene todos los datos capturados de Twitter, así como los resultados intermedios del procesamiento de datos posterior. La estructura interna de este directorio se discute en el Capítulo 5, Subsección 5.4.1.
- Directorio *dataPeriodico*: Contiene los ficheros de artículos y comentarios en estado bruto (como documento HTML). Cada periódico origina un nuevo subdirectorio, dentro del cual se almacenan por separado los

ficheros de artículos y comentarios. Su estructura interna se describe en el Capítulo 6, Subsección 6.2.1.

- Directorio *dataClasificadorBinario*: Contiene los datos de entrenamiento y evaluación definidos para la preparación de los modelos de aprendizaje de clasificación binaria, así como los modelos resultantes del ajuste y entrenamiento.

En la Figura 4.4 puede verse la jerarquía completa de los directorios de datos.

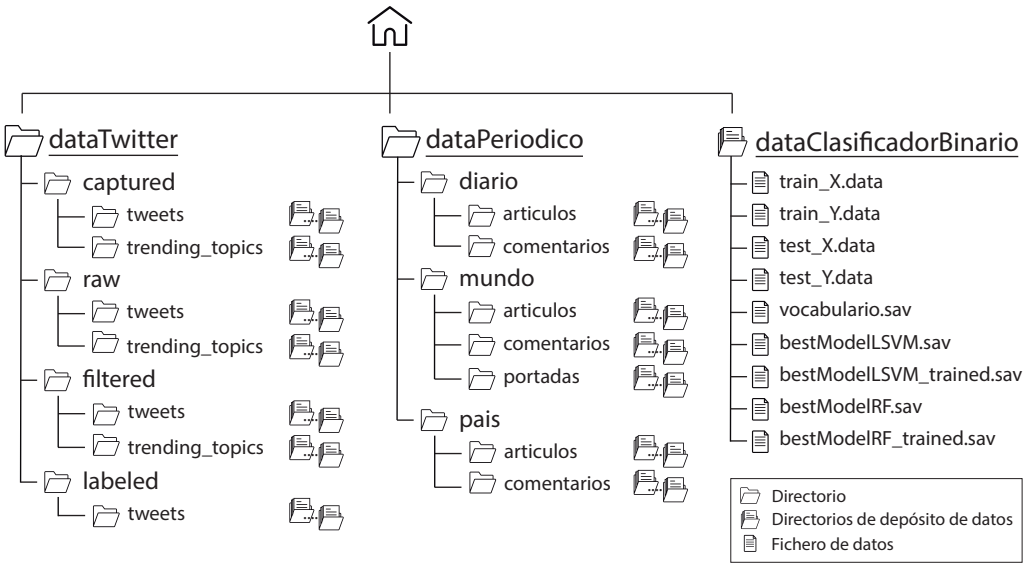


Figura 4.4: Estructura de directorios de datos.

4.7.2. Directorios de fuentes.

Contienen los diferentes *scripts* Python que implementan la funcionalidad necesaria para cada una de las fases del proceso. La relación completa de ficheros fuente y su ubicación en el espacio de almacenamiento se muestra en la Figura 4.5. A continuación se describe sucintamente la funcionalidad implementada en cada grupo de *scripts*:

- Directorio *common*: Contiene ficheros con la lógica de operaciones utilizadas frecuentemente a lo largo del flujo de trabajo, como operaciones de lectura de ficheros de datos o funciones de tratamiento de textos. Alberga también diferentes parámetros de configuración, que se reúnen en un único fichero para facilitar su actualización, y las credenciales de conexión con las diferentes interfaces externas.

- Directorio *scriptsCapturaTwitter*: Contiene la lógica del mecanismo de captura de *trending topics* y *tweets*, así como el código necesario para recuperar el sistema de problemas de conexión y el mantenimiento de un *log*.
- Directorio *scriptsPreprocesamiento*: Contiene los ficheros fuente necesarios para procesar los datos en crudo procedentes de Twitter y llevar a cabo operaciones de pre-procesamiento necesarias para permitir y aligerar el procesamiento posterior, como la eliminación de mensajes en otros idiomas y la identificación de *trending topics* en los *tweets* capturados. Además, también se lleva a cabo la construcción de muestras de *tweets* para cada una de las tendencias descubiertas.
- Directorio *scriptsScrape*: Contiene los ficheros ejecutables para realizar *web scraping* sobre los periódicos seleccionados para un intervalo de fechas fijado por el usuario a través de línea de comandos.
- Directorio *clasificadorBinario*: Recoge los *scripts* utilizados para generar el modelo de vectorización de textos necesario para la explotación de los datos en un algoritmo de aprendizaje automático, así como las sucesivas fases de construcción y evaluación de modelos para la clasificación de textos.
- Directorio *scriptsSeleccion*: Contiene el *script* de clasificación de *trending topics* en base a las muestras construidas y posterior filtrado de los *tweets* no relevantes.

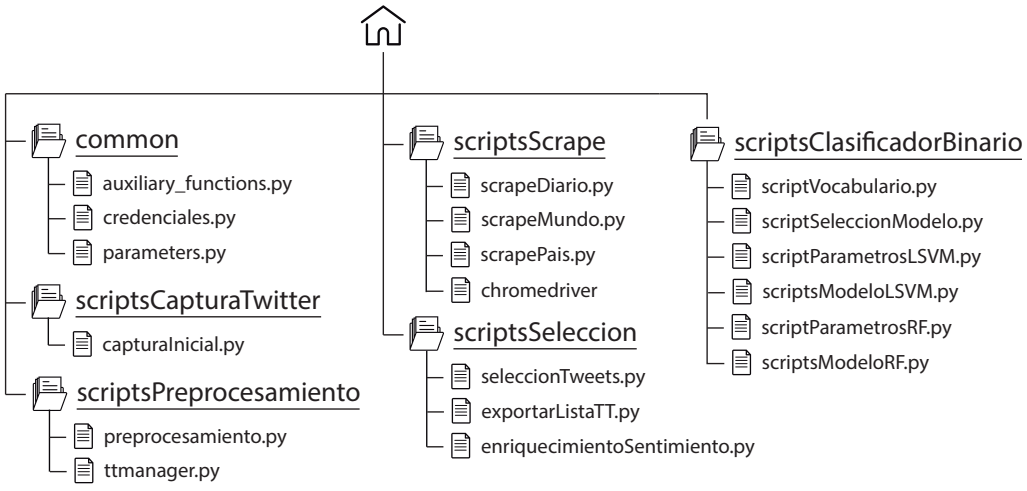


Figura 4.5: Estructura de directorios de ficheros fuente.

Capítulo 5

Procesamiento de datos de Twitter.

En este capítulo se describe el proceso de ETL (Extracción, Transformación y Carga) que implementa el *pipeline* descrito en la Sección 4.6, y que se aplicará sobre los datos para su uso en el ámbito del proyecto. Esta preparación consistirá en sucesivas fases de refinamiento que nos permitirán obtener unos datos significativos, sin redundancia y con una presentación adecuada para su explotación en las actividades de aprendizaje y predicción, partiendo de un conjunto de datos "en crudo" obtenidos directamente de las fuentes de datos seleccionadas. Estos datos serán posteriormente filtrados y enriquecidos para su introducción en el mecanismo de predicción desarrollado, dado que este solo utilizará una pequeña parte de todo el flujo capturado.

5.1. Análisis de fuentes de datos.

En primer lugar, es necesario caracterizar la fuente de los datos que utilizaremos para nutrir nuestro flujo de trabajo. Esto incluye tanto la forma en que se accederá a los datos –la API de Twitter–, como la forma en que se podrán encontrar los datos en su origen –principalmente como objetos JSON–.

5.1.1. API de Twitter.

La captura de datos se lleva a cabo por medio de la API pública de Twitter. La API contiene múltiples módulos que dan acceso a información

variada de la red social, aunque podríamos destacar dos: la API REST de búsqueda, que permite el acceso a *tweets* históricos publicados en la red social; y la API de *Streaming*, que proporciona acceso a la información publicada en tiempo real. Dispone de distintos planes de acceso o *endpoints* que proporcionan funcionalidades o capacidades avanzadas de trabajo con los datos generados en Twitter:

- *Standard*: Accesible con credenciales de usuario desarrollador. Permite acceder a la función de tiempo real y al servicio de búsqueda para los últimos siete días.
- *Premium*: Requiere el registro de un entorno de desarrollo dentro de la cuenta de desarrollador. En su versión gratuita (*Sandbox*), proporciona acceso a la búsqueda tanto en los últimos treinta días, como para en el archivo completo de Twitter, aunque con importantes limitaciones tanto en volumen de datos obtenidos, como en la configuración de criterios de búsqueda. La versión de pago amplía estos límites[1].
- *Enterprise*: Amplía las restricciones del plan *Premium* mediante un plan de pago sujeto a consumo. Orientado a su uso en empresas.

En este trabajo se trabajará exclusivamente con la versión *Standard* de la API, pues bastará con tener acceso a la funcionalidad de *streaming* y la de consulta de *trending topics*. Cabe señalar que la *Streaming* API no da acceso a la totalidad de la información contenida en Twitter, sino a una pequeña parte (se estima que se corresponde con una muestra del 1 % de los mensajes publicados, aunque no hay datos oficiales al respecto). No obstante, podemos asumir que el volumen de información será suficientemente significativo para satisfacer los objetivos marcados para el proyecto.

Las limitaciones de este plan que pueden afectar al proyecto se describen en la Tabla 5.1, expresadas en peticiones por ventana de tiempo (la ventana de tiempo establecida por Twitter es de 15 minutos). Se puede consultar el número de peticiones disponibles mediante el método de la API `application/rate_limit_status`.

Acción	Peticiones
Consultar lista de *hashtags*	75
Consultas de peticiones disponibles	180

Tabla 5.1: Límites de la API para las peticiones realizadas.

[1] `https://developer.twitter.com/en/pricing/search-30day` (visto: 23/05/19)

La conexión con las APIs de Twitter requiere dos pares de claves: uno para la autenticación mediante OAuth (*Consumer API keys*), y otro para el acceso a la API en sí (*Access Tokens*). Para obtenerlas, es necesario tener una cuenta personal de Twitter, darse de alta como desarrollador y registrar una *app* en el portal del desarrollador de Twitter[2]. Aunque es necesario obtener autorización para registrarse como desarrollador, pero la espera tras la presentación del formulario correspondiente es breve.

Una vez registrada la aplicación, las claves requeridas se encuentran en los detalles de la misma. Estas claves se incluirán en el código, para permitir la conexión del sistema con las APIs de Twitter, en forma de texto plano: las credenciales son específicas de la aplicación y fácilmente revocables, por lo que para el uso previsto para este sistema no es necesario establecer medidas de seguridad más restrictivas. En caso de desarrollar un sistema orientado a un público mayor, no se difundirían las claves obtenidas para trabajar en este proyecto (pues serían rápidamente anuladas por exceso de tráfico), y se exigiría a los usuarios la introducción de sus propias claves, que deberían cifrarse adecuadamente para asegurar su protección.

5.1.2. Obtención de tendencias.

La API de Twitter dispone de un método específico para acceder a la información de tendencias o *trending topics*[3]. Las llamadas a este método proporcionan una lista estática de las 50 tendencias más relevantes que se refresca cada 5 minutos (es un contenido cacheado). Esto limita la frecuencia máxima con que podemos refrescar los filtros de captura de nuestro sistema, puesto que consultas realizadas en la misma ventana entre dos refrescos nos proporcionará las mismas tendencias, independientemente de que estas hayan evolucionado en el flujo de datos.

El resultado de la consulta es un array de objetos JSON compuesto por los campos descritos en la Tabla 5.2. Asimismo, cada tendencia se describe mediante un objeto JSON con los campos de la Tabla 5.3.

5.1.3. Obtención de tweets.

La API de *streaming* nos permite obtener una muestra de los contenidos de Twitter en tiempo real, filtrando los mismos de acuerdo a distintos criterios que se especificarán más adelante. La API proporciona un flujo de

[2] `https://developer.twitter.com/en/apps` (visitado: 22/03/19)

[3] `https://developer.twitter.com/en/docs/trends/trends-for-location/api-reference/get-trends-place` (visitado: 22/03/19)

Campo	Descripción
locations	Array de localizaciones geográficas donde se detectaron las tendencias
as_of	Momento de captura de la lista
created_at	Momento de cacheo de la lista de *trending topics*
trends	Array de *trending topics* individuales (ver Tabla 5.3)

Tabla 5.2: Estructura de la respuesta de la API de tendencias.

Campo	Descripción
name	Cadena de caracteres de la tendencia
query	Equivalente a *name*, pero codificado como cadena válida para una URL
url	URL completa de búsqueda para obtener los tweets de la tendencia
promoted_content	Indica si el *trending topics* responde a una campaña de publicidad de Twitter Ads
tweet_volume	Número de tweets pertenecientes a la tendencia. A menudo nulo

Tabla 5.3: Estructura de un objeto *trending topic*.

tweets individuales, codificados como objetos JSON con un alto grado de anidamiento.

La estructura interna de un tweet es rica y compleja, y depende en gran medida de sus contenidos: texto plano, *hashtags*, menciones a usuarios, naturaleza del mensaje (original, cita o retweet). Los metadatos de un tweet no solo describen las características del mensaje, sino que pueden contener información de otras *entidades* que están relacionadas con el mensaje o embebidas en él (contenido multimedia, información de geolocalización, etcétera). En el Apéndice B se referencian los componentes más habituales dentro de un *tweet* descrito como JSON.

5.1.4. Modelo de datos de Twitter.

La API proporciona información de múltiples tipos de entidades de datos, en muchos casos embebida en los metadatos de los *tweets*. La relación entre las distintas entidades de datos puede observarse en la Figura 5.1. No se han incluido otras entidades que no intervienen en el proyecto, como las listas de usuarios, o relaciones entre entidades, como el hecho de que un

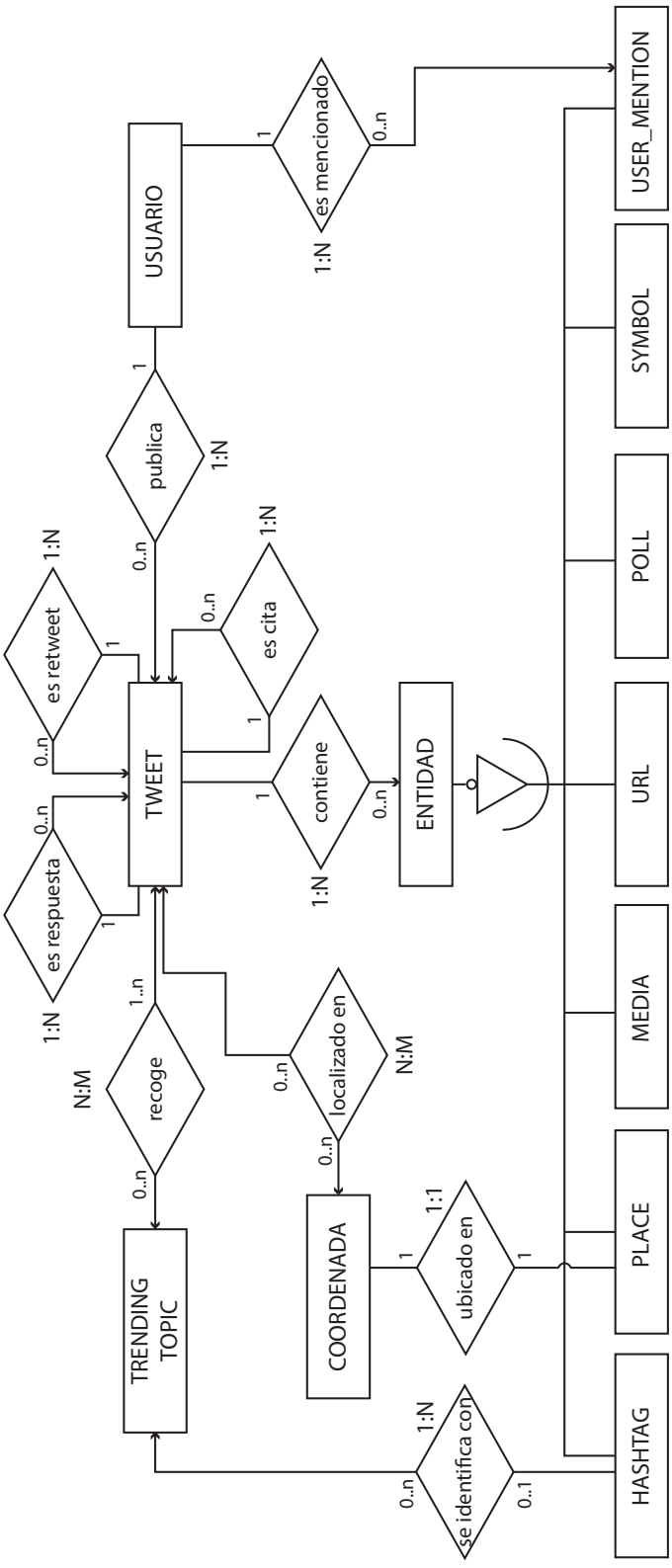


Figura 5.1: Modelo de datos de Twitter (simplificado).

usuario haga *retweet* o marque como "favorito" un *tweet*. A continuación se explica brevemente cada una de estas entidades, cuyos atributos completos se extraen de la relación de atributos de los objetos JSON correspondientes, descritos en el Apéndice B, y que no repetiremos por no tener un impacto significativo en el proyecto.

a. *Tweet* (*status*): Se corresponde con un mensaje individual emitido por un usuario. Contiene información del mensaje en sí (marca de tiempo, contenido, número de interacciones), de otros contenidos diferentes al texto (multimedia, menciones, *hashtags*), y del usuario que lo publicó (ver siguiente ítem).

b. Usuario (*user*): Describe un usuario registrado en la red social. Esta información comprende los identificadores del usuario, datos incluidos en el perfil (descripción del usuario, sitios web, número de mensajes, etcétera) y la configuración estética de su perfil.

c. Entidades (*entities*): Este campo alberga diferentes objetos que pueden ser incluidos en un *tweet* y que han sido identificados por Twitter durante el procesamiento del mismo. Algunos ejemplos de objetos que recoge este campo son los *hashtags* que incluye el *tweet*, encuestas, enlaces, menciones a usuarios, etcétera. Es decir, todos aquellos elementos individuales que exceden el simple texto dentro del *tweet*.

d. Entidades extendidas (*extended entities*): Es un concepto introducido para adaptarse a la evolución y la variabilidad de los contenidos de un tweet[4]. Incluye un único atributo, *media*, que recoge todos los contenidos que exceden la definición original del campo *entities.media* en forma de array de objetos JSON, representando los distintos objetos multimedia que se incluyan en el mismo.

e. Recursos multimedia (*media*): Cuando un tweet contiene una imagen incrustada (no enlazada mediante un hiperenlace), este campo describe sus principales propiedades: ruta de la imagen, nombre, dimensiones, etcétera. Si se adjuntan más fotos o contenidos multimedia de distinta naturaleza, se utiliza el siguiente campo.

[4]Por ejemplo, cuando se empezó a permitir incluir hasta cuatro fotos en un tweet, en lugar de una, o se introdujeron los vídeos y GIF animados. Un cambio en la implementación del campo *entities.media* original provocaría problemas de compatibilidad en todo el software que utilizara la API, mientras que implementando las entidades extendidas pueden enmascararse los cambios y permitir un funcionamiento estable hasta que se introduzcan los cambios necesarios en las aplicaciones cliente para explotar los nuevos contenidos.

f. Menciones a usuarios (*mentions*): Recoge los datos básicos de los usuarios que son mencionados en el tweet, tales como su identificador y su nombre.

g. URL: Los hiperenlaces incluidos en el tweet se tratan como objetos individuales. Se incluye la ruta enlazada, su posición en el tweet y el enlace acortado que genera Twitter automáticamente.

h. Hashtags: Los *hashtags* incluidos en el tweet se recogen como objetos individuales, indicando el texto (sin la almohadilla) y la posición en el tweet.

i. Localizaciones geográficas (*places*): Almacena información de geolocalización del tweet como localización exacta mediante coordenadas geográficas o como un lugar de Twitter (*Twitter Place*), que indica un lugar en particular (una ciudad, un país) que no señala una localización exacta.

5.2. Modelo de datos final.

Al finalizar el procesamiento de los datos se habrá descartado la información de la mayor parte de las entidades, puesto que en el alcance actual del proyecto solo está previsto trabajar con la información más básica. Esta información se refleja en el modelo Entidad-Relación incluido en la Figura 5.2. En el futuro, sería interesante explorar las posibilidades que ofrecería el resto de la información para nuestro propósito, como la información geográfica o la generación de redes de usuarios e interacciones (relaciones seguidor-seguido, menciones a usuarios en los *tweets*), lo que motivaría la expansión de este modelo tan reducido.

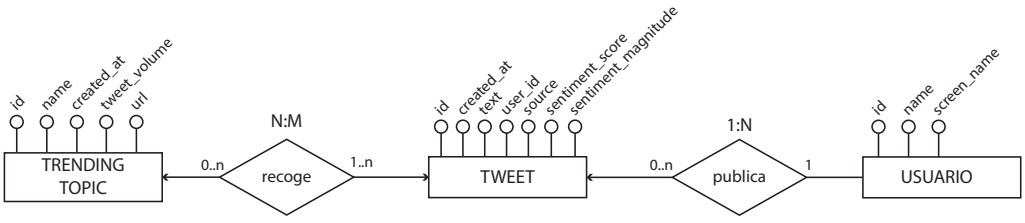


Figura 5.2: Modelo de datos final.

5.3. Mapa lógico de datos.

En la Tabla 5.4 puede verse el mapa lógico de datos, que refleja las transformaciones de datos que median entre el modelo descriptivo de la fuente y el modelo de datos final del proyecto.Se incluyen en la tabla únicamente aquellos atributos iniciales que tienen continuidad en el proceso de transformación, pues los atributos de la fuente son muy numerosos y se eliminan todos, salvo los indicados en la tabla, en la fase de PREDICT.

Los atributos inexistentes en cada fase del proceso se denotan con un guion (-), mientras que los atributos que no cambian en una fase se marcan con un signo de igualdad (=).

5.4. Implementación ETL.

En esta Sección se aborda la implementación concreta que se ha desarrollado para asegurar la transformación de los datos desde su forma inicial hasta el modelo de datos definido en la Sección 5.2.

5.4.1. Diseño del almacenamiento de datos de Twitter.

En la descripción del almacenamiento de datos realizada en la Subsección 4.7.1 se prevé la creación de cuatro sub-directorios distintos dentro del del directorio correspondiente a los datos de Twitter. Cada subdirectorio contendrá los resultados de la ejecución de cada una de las cuatro fases en que dividimos el procesamiento de la información de Twitter:

- Directorio de captura: *captured*. Recoge los directorios donde se depositarán los datos de *trending topics* y *tweets* a medida que se capturen mediante la API de Twitter.
- Directorio de almacenamiento de datos en crudo: *raw*. Aloja los datos resultantes del pre-procesamiento, es decir, los *trending topics* consolidados junto con las muestras de mensajes que utilizaremos para clasificarlos, y los *tweets* enriquecidos con los *trending topics* detectados en ellos.
- Directorio de *tweets* filtrados: *filtered*. Contiene los *tweets* pertinentes, una vez se han determinado los *trending topics* relevantes y se han filtrado los *tweets* que están relacionados con los mismos.

CAPTURE		RAW	FILTER	LABEL	PREDICT		FINAL	
Tipo	Atr.					Tipo	Atr.	
TT	-	Creación	Filtrado de registros	=	=	string	id	TT
	name	string	Normalización	=	=	string	name	
	created_at	string	=	=	Conversión tipo	Date	created_at	
	tweet_volume	string	=	=	Conversión tipo	int	tweet_volume	
	url	string	=	=	=	string	url	
TWEET	id	string	=	Filtrado de registros	=	=	string	id	TWEET
	created_at	string	=		=	Conversión tipo	Date	created_at	
	text	string	Normalización		=	=	string	text	
	-		-		Cálculo	=	float	sentiment_score	
	-		-		Cálculo	=	float	sentiment_magnitude	
	-		-		-	Cálculo	Array	puntuacion	
TWEET	id_usuario	string	=		=	=	string	id_usuario	USUARIO
	name	string	=		=	=	string	name	
	screen_name	string	=		=	=	string	screen_name	
	-		-		-	Cálculo	float	puntuacion	
	-		-		-	Cálculo	float	max_puntuacion	
	-		-		-	Cálculo	string	partido	

Tabla 5.4: Mapa lógico de datos.

- Directorio de *tweets* etiquetados: *labeled*. Almacena los *tweets* relevantes, una vez han sido procesados para determinar el sentimiento que contienen y el sujeto a que se refieren.

Los datos obtenidos durante la captura se almacenan directamente en el directorio correspondiente. Dentro del mismo se localizarán sendos directorios para alojar los datos de *trending Topics* y los *tweets* correspondientes, con un subdirectorio para cada fecha de captura, siguiendo el criterio de denominación: *fecha=AAAA-MM-DD*. Así:

- Los *trending topics* se almacenarán a razón de una lista por fichero, volcando la respuesta de la API a un fichero nombrado con el siguiente criterio: *trending_AAAA-MM-DD_hh-mm*, de acuerdo al momento de obtención de la lista en cuestión.

- Los *tweets* se escribirán secuencialmente en ficheros, a razón de un *tweet* (en forma de objeto JSON) por línea. Cada hora se generará un nuevo fichero, independientemente del tamaño que alcance el mismo durante esa hora. No obstante, sería posible imponer una condición adicional de cambio de fichero al alcanzar un tamaño determinado, si esto conviniese para la escalabilidad del proceso. Los ficheros seguirán un criterio de nombres análogo al de los *trending topics*: *tweets_AAAA-MM-DD_hh*.

Una vez pre-procesados, consideraremos los datos en estado *raw*. Los *tweets* se almacenarán siguiendo el mismo criterio que se ha descrito arriba. Los trending topics, en cambio, se transformarán a un fichero CSV que recoge todas las tendencias capturadas durante el día, siguiendo el patrón de nombres *trending_AAAA-MM-DD*. Adicionalmente, se almacenará, con el nombre *samples_AAAA-MM-DD*, un fichero JSON que contiene las muestras de *tweets* recogidas para cada *trending topic* con vista a su clasificación.

5.4.2. Captura.

En primer lugar, es necesario caracterizar los medios por los que se accede a la API de Twitter, así como las decisiones de diseño que se han tomado para asegurar una captura adaptable a la evolución del flujo de datos y respetuosa con las restricciones impuestas para el uso de dicha API, de forma que el proceso de captura sea seguro y estable, y no se detenga por errores comunes (como el exceso de peticiones por ventana de tiempo o una pérdida momentánea de conexión).

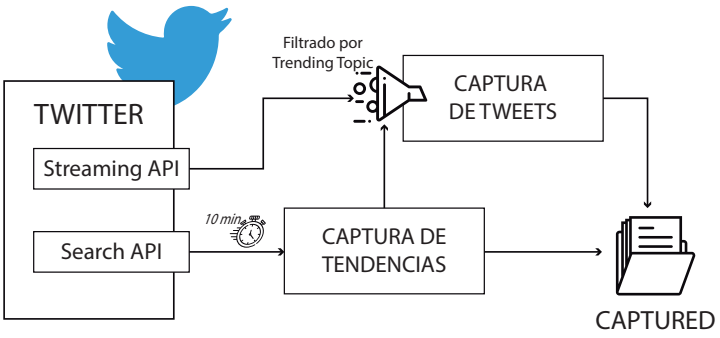


Figura 5.3: Descripción del proceso de captura.

5.4.2.1. Librería *Tweepy*.

En lugar de acceder directamente a la API de Twitter, se utilizará una librería que facilite la gestión de las conexiones y las credenciales de acceso, así como la modificación adaptativa de los filtros aplicados a lo largo del mismo. Se ha elegido *Tweepy* [27], una librería implementada en Python cuyo desarrollo lleva activo desde 2009. En el momento de publicación de este trabajo, se encuentra en su versión 3.7.0, liberada en noviembre de 2018.

Tweepy proporciona una interfaz de métodos muy similar a la propia API de Twitter, con lo que es sencillo apoyarse en la documentación oficial de Twitter en adición a la propia documentación de Tweepy o su Github público[5]. De esta forma, utilizaremos los métodos análogos a los descritos de la API oficial para obtener la información necesaria para el procesamiento posterior.

Obtención de trending topics. Tweepy dispone del método `trends_place` para obtener una lista de Trending Topics restringida a una localización geográfica, que es un trasunto exacto de `GET trends/place` en cuanto a parámetros y resultados. Los parámetros que acepta este método pueden verse en la Tabla 5.5.

Acceso a la API de Streaming. El método utilizado para el acceso al componente de tiempo real de Twitter es *filter*[6], dentro del módulo de la librería dedicado a la conexión con la API de Streaming. Este método ofrece una larga lista de parámetros con los que configurar el flujo de datos de entrada, como puede verse en la Tabla 5.6.

[5] `https://github.com/tweepy/tweepy` (visitado: 24/03/19)

[6] `https://developer.twitter.com/en/docs/tweets/filter-realtime/api-reference/post-statuses-filter` (visitado: 24/03/19)

`API.trends_place`

Parámetro	Tipo	Descripción
id	Lista	Identificadores geográficos de Yahoo Where on Earth (WOEID).
exclude	Lista	Opcional. Lista de *trending topics* que deben ser ignorados.

Tabla 5.5: Parámetros del método de obtención de *trending topics*.

`STREAM.filter`

Parámetro	Tipo	Descripción
track	Lista	Opcional. Lista de términos que deben estar presentes en los *tweets* capturados.
is_async	Booleano	Activa la ejecución en segundo plano del receptor del streaming.
encoding	Cadena	Codificación de texto de la respuesta.
follow	Lista	Lista de IDs de usuario. Solo se capturan *tweets* enviados por estos usuarios, o respuestas a los mismos.
filter_level	Cadena	Indica el nivel mínimo de importancia requerido en los mensajes del stream (asignado automáticamente por Twitter). Valores: [‘none’, ‘low’, ‘medium’]
stall_warnings	Booleano	Activa los mensajes de aviso de desconexión del cliente si el flujo excede la capacidad de procesamiento del mismo, provocando un encolamiento por parte de la API.
locations	Lista	Opcional. Lista de pares latitud y longitud que forman un polígono. Solo se proporcionarán *tweets geolocalizados* que se hayan enviado dentro de dicho polígono.
languages	Lista	Opcional. Lista de elementos separados por comas con los códigos de idioma.

Tabla 5.6: Parámetros del método de obtención de *tweets* del flujo en tiempo real.

5.4.2.2. Proceso de captura.

El proceso de captura que se ha diseñado comprende dos componentes:

a. Obtención de listas *trending topics*. Este subproceso se lanzará con una

periodicidad definida, de manera que podamos adaptar la configuración de la captura a la evolución del flujo con una frecuencia adecuada.

b. Monitorización del *stream* de *tweets*. Este subproceso se ejecuta de forma continua hasta que se interrumpe manualmente. Tras cada actualización de la lista de *trending topics*, se refrescará la conexión con la API de *Streaming* para aplicar los nuevos criterios de filtrado.

   Debido a las restricciones del plan gratuito de Twitter, que no permite actualizar una monitorización en curso, en cada modificación de la lista de tendencias será necesario detener la captura, definir un nuevo proceso de monitorización con los nuevos parámetros y relanzar el proceso. Idealmente, esta configuración debería hacerse "en caliente", sin detener el proceso, para minimizar la pérdida de información.

El período de refresco de la lista de *Topics* se ha establecido en 10 minutos. Este valor se sitúa por encima del límite mínimo establecido por la frecuencia con que Twitter actualiza la lista de *trending topics* que ofrece (5 minutos), de manera que no obtengamos dos veces la misma lista procedente de la API.

Este valor es tentativo y configurable mediante una variable global del sistema. En el estado del arte, con casos de uso similares al desarrollado aquí, son usuales intervalos de 30 minutos, que ofrece un buen compromiso entre adaptabilidad del sistema y complejidad del procesamiento necesario. No obstante, una granularidad más fina permite detectar nuevas tendencias más rápidamente y se adapta mejor a nuestras necesidades: una parte importante de la información que deseamos capturar se corresponde con eventos localizados temporalmente, como debates televisados o reacciones a noticias de última hora. Es decir, de surgimiento instantáneo y con una reacción en redes sociales muy rápida (creación de *hashtags* para la publicitación del programa, expectación tras el anuncio del evento, agitación del público por parte de los acontecimientos durante el evento...).

Finalmente, en ambos casos se impondrán las restricciones necesarias para limitar el alcance geográfico de la monitorización. Para obtener los *trending topics*, se indicará, por medio del parámetro *id*, en las peticiones realizadas el identificador de España, de manera que obtengamos las tendencias detectadas en todo el territorio nacional. En lo que respecta a los tweets, no podemos imponer restricciones geográficas, pues perderíamos todos aquellos mensajes que no tengan información de geolocalización explícita. En su lugar, controlaremos su procedencia verificando el idioma en que fueron escritos, que previsiblemente se corresponde con el español o una de las lenguas co-oficiales del país en las distintas regiones.

5.4.3. Pre-procesamiento.

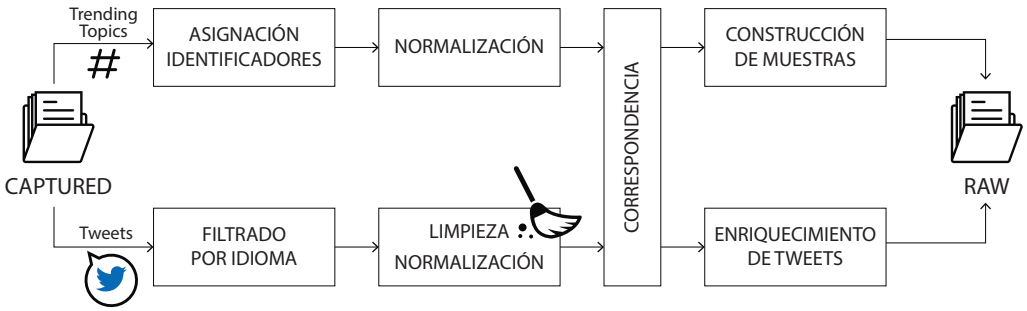


Figura 5.4: Descripción del flujo de pre-procesamiento.

La primera fase de transformación comprende operaciones sencillas orientadas a eliminar información irrelevante y normalizar los datos. De esta manera, simplificaremos su procesamiento tanto reduciendo la complejidad de los datos como su volumen. Posteriormente, se enriquecerán los datos resultantes estableciendo relaciones entre los *tweets* y los *trending topics*, puesto que Twitter no proporciona esta información y es un requisito indispensable para múltiples operaciones que se llevarán a cabo más adelante.

Inicialmente, se ha previsto una frecuencia de ejecución diaria de este proceso, pero podría fácilmente adaptarse a una ejecución cada hora o incluso tras cada actualización de la lista de *trending topics* si queremos obtener información refinada de forma inmediata. En cualquiera de los casos, este pre-procesamiento consiste en las fases que explicamos a continuación.

5.4.3.1. Identificación de Trending Topics.

El almacenamiento de los *trending topics* planteado origina un gran número de ficheros de pequeño tamaño, algo ineficiente para la infraestructura planteada para su tratamiento posterior. Además, no están identificados de forma unívoca, por lo que es conveniente asignarles un identificador para facilitar operaciones posteriores.

Procesaremos los *trending topics* de manera que se almacenen los datos de un día en un solo fichero, guardándolos además en un formato tabular para facilitar su legibilidad (en concreto, un CSV con '|' como carácter separador). Asimismo, se asignará a cada *trending topics* un ID formado por la concatenación de la marca de tiempo del instante de generación de la lista y un número incremental de dos cifras (00, 01, etcétera).

5.4.3.2. Filtrado de tweets en otros idiomas.

Twitter proporciona información sobre el idioma detectado en cada uno de los *tweets*. Utilizaremos esta información para descartar, al comienzo del proceso, aquellos mensajes que no hayan sido escritos en español, por razones de dominio de interés (los votantes en el evento electoral objetivo se expresarán, en general, en este idioma) y técnicas (el clasificador utilizado más adelante ha sido entrenado con textos en español).

5.4.3.3. Normalización del texto de los tweets y los trending topics.

Para facilitar tanto la determinación de la correspondencia *tweet-trending topic* como el trabajo automatizado con el contenido de los *tweets*, es necesario un proceso de normalización de texto. Esta operativa es habitual en tareas de Procesamiento de Lenguaje Natural, y su objetivo es reducir la variabilidad de los componentes del texto y refinar la presentación de información que poseen. El proceso seguido consiste en las siguientes operaciones:

1. Obtener el texto completo del *tweet*: Los mensajes con una longitud superior a 140 caracteres almacenan una versión truncada del texto en el atributo correspondiente. El contenido íntegro del *tweet* es un atributo (*full_text*) de un nuevo tipo de objeto embebido en el *tweet*: el atributo *extended_tweet*. Esta decisión se tomó para asegurar la retrocompatibilidad cuando se incrementó la longitud máxima a 280 caracteres.

2. Eliminación de cadenas que no forman parte del lenguaje natural, como direcciones URL, menciones de usuario o códigos de *retweet*.

3. Eliminación de caracteres irrelevantes: Se mantienen los caracteres alfanuméricos y espacios simples, eliminando o sustituyendo (en el caso de espacios múltiples o especiales) el resto de caracteres, incluyendo los signos de puntuación.

4. Eliminación de *stop-words* o palabras vacías. Estas son palabras usadas frecuentemente en un lenguaje con el propósito de dar cohesión al texto, pero que no portan un significado propio. Ejemplos de palabras vacías en castellano son las preposiciones o las partículas “que” o “y”.

5. Normalización: Se convierte el texto a minúsculas y se sustituyen caracteres especiales como vocales acentuadas por vocales simples para reducir la influencia de las faltas de ortografía cometidas en la redacción del texto.

6. Segmentación o *tokenización* por palabras: Se transforman las cadenas de caracteres resultantes en listas de cadenas, donde cada elemento es una palabra del texto original.

Durante el desarrollo se ha experimentado con la lematización del texto para reducir aún más la variabilidad, pero no proporcionaba beneficios apreciables en la calidad de la clasificación de los textos. La lematización consiste en la eliminación de desinencias de las palabras para reducirlas a una forma básica (por ejemplo, cambiar cualquier conjugación verbal en el infinitivo que le corresponde, o los plurales por singulares). No debe confundirse con el *stemming*, que reduce cada palabra a su raíz léxica (por ejemplo, *cantando* se transformaría en *cant*). Por otra parte, el resultado obtenido resultaba más difícil de interpretar, pues en ocasiones la librería utilizada, *Spacy*, introducía términos extraños (por ejemplo, *españolar*).

5.4.3.4. Correspondencia entre un tweet y las tendencias asociadas.

La identificación de los *tweets* asociados a una tendencia se llevará a cabo en dos fases:

- Los *hashtags* que contiene un *tweet* se proporcionan como parte de sus metadatos, en el atributo *entities*, por lo que es sencillo determinar si pertenece a alguno de los *hashtags* presentes en la lista de tendencias.
- El resto de *trending topics* (no *hashtags*) se producen principalmente por la repetición de uno o varios términos en un cierto volumen de mensajes. Para identificar los *tweets* que los conforman, buscaremos en el cuerpo de los mismos los términos que componen el *trending topic*.

Los *hashtags* y los *trending topics* se someten a las mismas operaciones de normalización que el texto para facilitar su localización en el *tweet*, ya que la búsqueda de términos clave en el texto de los *tweets* se lleva a cabo mediante búsqueda exacta. Se han llevado a cabo pruebas con coincidencia difusa (*fuzzy matching*) a través de la librería *fuzzywuzzy*, basada en la distancia de Levenhstein[7], pero dado el mecanismo de captura que se ha implementado (búsqueda exacta por palabras clave), este sistema no ofrece un gran beneficio y puede introducir ruido en el proceso.

[7]La distancia de Levenhstein o distancia de edición entre dos cadenas de caracteres es el número mínimo de operaciones sobre caracteres que se deben aplicar para convertir la primera en la segunda. Las operaciones posibles son la sustitución, la eliminación y la inserción de caracteres.

Una vez identificados, se agrega al *tweet* como nuevo campo, *id_tts*, un *array* de cadenas de texto con los identificadores de los *trending topics* que le corresponden.

5.4.3.5. Construcción de muestras de *trending topics*.

Dado que los *trending topics* suelen ser demasiado cortos (una o dos palabras, o una en el caso de los *hashtags*), es difícil caracterizar su temática por sí mismos. Por ello, una posible solución pasa por construir una muestra de *tweets* que se asimilan a dicho *trending topic* y evaluar su relevancia individual. En función del grado de relevancia global de la muestra, se decidirá si el *trending topic* debe mantenerse o descartarse del flujo de trabajo.

Con este fin, durante el proceso de determinación de la correspondencia *tweets-trending topics* se irá construyendo dicha muestra para cada tendencia. Aunque podríamos utilizar todos y cada uno de los *tweets* asociados a la tendencia, se incrementaría la exigencia del procesamiento posterior, además de no resultar particularmente beneficioso en términos de exactitud: se asume que cualquier muestra aleatoria de un tamaño suficiente permitirá caracterizar el *trending topic*, considerando que los *tweets* de una tendencia tendrán un alto grado de coherencia semántica. Por tanto, recogeremos un número determinado de *tweets* para cada tendencia, cuyo valor se determinará en código mediante el parámetro correspondiente (inicialmente fijado en 100).

Para no favorecer un sesgo hacia los primeros o últimos *tweets* capturados para cada tendencia, se recogerán todos los *tweets* hasta que se alcance el tamaño máximo de la muestra, y después se llevará a cabo una política de reemplazo estocástico. Es decir, con una probabilidad prefijada, cada nuevo *tweet* podrá sustituir uno de los *tweets* que la componen, elegido de forma aleatoria.

5.4.4. Procesamiento avanzado.

Tras las transformaciones básicas descritas, se dispondrá de la información necesaria para filtrar los *tweets* semánticamente relevantes en el ámbito del proyecto. Como paso previo, será necesario caracterizar, utilizando un clasificador binario (que se discute, más adelante, en el Capítulo 6), las tendencias detectadas en base a las muestras construidas. De esta manera, descartaremos todos aquellos *tweets* pertenecientes a tendencias sin valor, reduciendo enormemente el volumen de datos sobre los que aplicar un procesamiento computacionalmente más costoso, y económicamente caro.

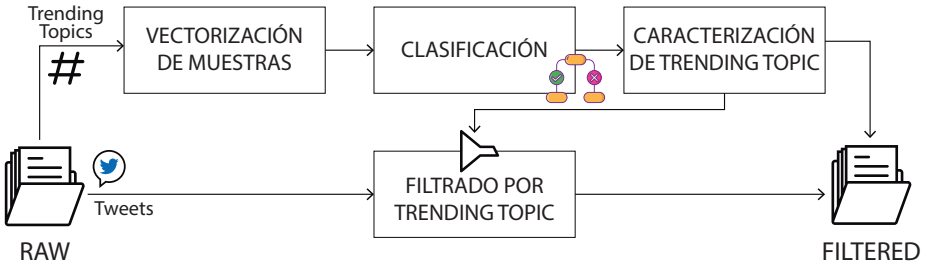


Figura 5.5: Descripción del flujo de procesamiento avanzado.

Sobre los datos restantes, podremos aplicar análisis de sentimiento de cara a identificar la filiación política del autor y así realizar la estimación de intención de voto sobre la población considerada.

5.4.4.1. Vectorización.

La aplicación del modelo de clasificación requiere la transformación de los textos de entrada en vectores de palabras, utilizando como base un *vocabulario* determinado: en cada posición del vector, se indica la presencia o no en el texto de una palabra de dicho vocabulario. De esta manera, podemos codificar un texto como un conjunto de números, que podemos utilizar como vector de características descriptivo de cada texto de entrada o ejemplo. La generación del vocabulario utilizado se realizará utilizando el conjunto de datos que se empleará para entrenar el clasificador, y por tanto se describirá en el Capítulo 6, Subsección 6.2.4.

Una vez convertidos en vectores de características los textos pertenecientes a cada muestra recogida, podremos proceder a clasificarlos, y con ellos a caracterizar las tendencias.

5.4.4.2. Clasificación de Trending Topics.

Para determinar si un *trending topic* trata sobre política o no, evaluaremos el número de *tweets* de su muestra que son anotados como 'política' por el clasificador. Como el tamaño de las muestras recogidas es muy variable, estableceremos ciertos umbrales para la cantidad de textos relevantes en la muestra para regular la exigencia del criterio de selección. En primer lugar, se impondrá un tamaño mínimo de muestra de 5 *tweets*: de esta manera, excluiremos tendencias con baja representatividad en el conjunto de datos (recordemos que la API de *Streaming* no proporciona el flujo completo, sino una pequeña parte de los mensajes publicados).

Si la muestra es suficientemente grande, se impondrá una proporción mínima de mensajes considerados 'política' del 50 %. Si su tamaño se sitúa por debajo de un cierto valor (establecido en 50 textos), se incrementará la proporción hasta el 75 %. Así, el sistema será más permisivo cuando puede disponer de evidencias más fundamentadas sobre la naturaleza de la tendencia, y más exigente cuando la información sea más escasa. De esta manera, podemos configurar el sistema para adaptarlo a las características del clasificador, balanceando estos parámetros para compensar posibles errores del clasificador. Finalmente, si la muestra se hallara por debajo del mínimo indicado antes, la tendencia se descarta directamente.

5.4.4.3. Selección de tweets en base a sus Trending Topics.

Una vez identificados los *trending topics* relevantes, filtraremos los *tweets* para mantener únicamente aquellos que contienen al menos una tendencia relevante. De esta manera, evitamos la necesidad de clasificar todos y cada uno de los textos, con el consiguiente riesgo de perder muchos de los textos individuales a causa del error cometido por el modelo durante la clasificación. Además, ampliamos el abanico de datos (a priori relevantes) al que tenemos acceso, de acuerdo con la ya mencionada coherencia interna de cada tendencia. El filtrado de los *tweets* se llevará a cabo utilizando los identificadores de tendencia que asignamos a cada *tweet* durante el pre-procesamiento. En el Apéndice C puede verse una lista completa de los *trending topics* que permanecen tras el filtrado.

5.4.4.4. Filtrado y segmentación manual de Trending Topics.

Aunque en el filtrado de *trending topics* se han eliminado la mayoría de las tendencias irrelevantes, de cara a asegurar un mejor resultado durante la realización de predicciones se ha decidido incluir una fase de revisión manual de las tendencias restantes. Al mismo tiempo, se clasificarán las tendencias pertinentes en diferentes categorías, de acuerdo a la formación política a la que hagan referencia. Se ha determinado que deberán identificarse seis categorías: cinco correspondientes a los principales partidos nacionales (PSOE, PP, Ciudadanos, Unidas Podemos y VOX), y uno para el resto de formaciones políticas. Los *trending topics* se distribuirán entre los diferentes partidos políticos bajo el criterio de que "los *trending topics* asociados al partido son positivos para el mismo", de manera que si un *tweet* positivo se adhiere a una tendencia asociada a un partido, se incremente el valor de puntuación del partido, y si el *tweet* es negativo, se reduzca.

En cierto punto se planteó un modelo más complejo de segmentación de los *trending topics* que pasaba por identificar, para cada partido, dos conjuntos de tendencias que se correspondieran con las que son positivas para el mismo, y las que son negativas. De esta manera, podríamos entretejer relaciones entre los partidos, reflejando que algunas tendencias que son positivas para uno son intrínsecamente negativas para otros, y así lograr una mejor caracterización de cada usuario: al ser el voto único y exclusivo, cualquier muestra de apoyo en una dirección debería resultar en detrimento de todas las demás, o al menos de algunas. Por ejemplo, si una persona se expresa en términos amables con respecto a Unidas Podemos, sería correcto reducir en la misma medida su simpatía por el Partido Popular para acentuar más la caracterización ideológica del usuario. No obstante, se decidió postergar la evaluación de este modelo para respetar el alcance marcado para el trabajo, por lo que quedará como trabajo futuro.

Para posibilitar esta operación, se exportará en el directorio *scriptsSeleccion* un fichero de texto plano, *lista_tts_export.py*, con todos los *trending topics* relevantes para el período considerado, a razón de una tendencia por línea. Estos elementos deberán disponerse, en un nuevo fichero, como un objeto JSON con seis atributos, uno por cada categoría, cuyo valor sea un *array* con todos los términos correspondientes a ese partido. El fichero se denomina *lista_final.txt*, y se ubicará en la misma localización que la lista exportada.

Utilizando los *trending topics* seleccionados, filtraremos una vez más los *tweets*, manteniendo solo los pertenecientes a dichas tendencias, y pasaremos a procesarlos para obtener su información de sentimiento. En el Apéndice C se incluye una posible selección manual de los *trending topics* resultantes del paso anterior.

5.4.4.5. Análisis de sentimiento de tweets.

El último paso consiste en enriquecer los datos de los *tweets* identificando el sentimiento que portan, para finalmente seleccionar la información de los *tweets* prevista en el modelo de datos final descrito en la Sección 5.2 de este mismo Capítulo.

Se utilizará la *Natural Language* API del servicio Google Cloud de procesamiento en la nube (descrito en la Sección 4.5, Capítulo 4), a la que se accederá por medio de la librería cliente implementada en Python que suministra la propia compañía.

Se procesará cada texto seleccionado de forma individual, aplicando solo parte del pre-procesamiento indicado en apartados anteriores; en concreto, la eliminación de cadenas de caracteres idiosincrásicas de Twitter y que

no forman parte del lenguaje natural, como las URLs, las menciones de usuarios o los códigos de *retweet*. La propia API se encargará de aplicar la normalización necesaria para llevar a cabo su procesamiento. Para cada texto, la petición correspondiente generará una respuesta con los dos valores indicados en nuestra descripción de la API. Ambos se incluirán como atributos en la información de cada *tweet*, con los nombres previstos en el modelo: *sentiment_score* y *sentiment_magnitude*, a fin de ser explotados en la fase de predicción de datos.

5.5. Descripción de los datos obtenidos.

5.5.1. Conjunto inicial.

Al finalizar el periodo de captura, fijado para el mes de abril de 2019, el conjunto de datos alcanzó un total de 135.277 MB (unos 133 GB) de datos capturados. Es necesario señalar que, debido a un error desconocido durante el proceso de captura, se perdieron los datos correspondientes al período entre el 16 y el 25 de abril (incluidos, parcialmente), por lo que podríamos estimar el volumen total de datos para un mes en más de 200 GB. Desde luego, el tamaño de estos datos fundamenta la consideración de este problema como Big Data, especialmente tomando en cuenta que ya durante la captura se está limitando la cantidad de información capturada por medio del uso de la información de tendencias como filtro del flujo. En la Tabla 5.7 se muestran algunas estadísticas adicionales sobre el conjunto de datos capturado.

Datos	Tamaño (MB)	Núm. ficheros	Núm. registros
Tweets	135.249	503	20.692.280
Trending topics	28	2.965	148.250

Tabla 5.7: Estadísticas del conjunto de datos *captured*.

5.5.2. Conjunto tras el pre-procesamiento.

Durante el pre-procesamiento se introducen varios cambios que afectan a la cantidad de datos que se maneja: se descartan todos los mensajes escritos en un idioma distinto al español, se enriquecen los datos mediante los identificadores asignados a los *trending topics* individuales, y se construyen las muestras de *tweets* para la clasificación de las tendencias. Tras estas

operaciones, el volumen total de datos es de 31.324 MB. Las características del conjunto de datos generado se reflejan en la Tabla 5.8.

Datos	Tamaño (MB)	Núm. ficheros	Núm. registros
Tweets	31.062	503	4.707.676
Trending topics	24	22	148.250
Muestras	239	22	148.250

Tabla 5.8: Estadísticas del conjunto de datos *raw*.

Cabe señalar que, a pesar de agregar información a los *trending topics*, el tamaño total de estos se reduce. Esto se debe al modo en que se almacenan los datos, que pasan de estar en formato JSON a presentarse como CSV, más eficiente en términos de espacio de almacenamiento. Además, se recogen todas las tendencias de un día en un único fichero, reduciendo notablemente el número de ficheros utilizados.

5.5.3. Conjunto filtrado.

Tras la clasificación de los *trending topics* y la subsecuente selección de los *tweets* asociados a ellos, el tamaño del conjunto de datos se ha reducido notablemente, a unos 7.257 MB, como puede verse en la Tabla 5.9.

Datos	Tamaño (MB)	Núm. ficheros	Núm. registros
Tweets	7.256	479	713.340
Trending topics	24	21	45.632

Tabla 5.9: Estadísticas del conjunto de datos *filtered*.

5.5.4. Conjunto etiquetado.

Una vez se ha llegado a la forma final de los datos, habremos descartado una gran proporción de *tweets*, además de la mayor parte de los atributos de cada *tweet* individual, alcanzando un tamaño total de 80,8 MB, lo que supone una reducción de 99,94 % sobre el volumen de datos de partida.

Datos	Tamaño (MB)	Núm. ficheros	Núm. registros
Tweets	81	159	157.771

Tabla 5.10: Estadísticas del conjunto de datos *filtered*.

Capítulo 6

Clasificación de tweets.

Twitter es una red social con contenidos muy diversos, y es claro que sus usuarios tienen otros intereses comunes aparte de la política que generan corrientes de opinión y dan lugar a *trending topics* en la web. Por tanto, es imprescindible realizar un filtrado de los conjuntos de datos capturados para trabajar únicamente con aquellos que proporcionen valor a nuestros propósitos. Esta división puede ser difusa en muchos casos, pues en los últimos tiempos la política ha tendido a impregnar otras tantas áreas de la actualidad informativa, y tratarla adecuadamente excedería el alcance del trabajo y, posiblemente, el ámbito de la titulación; por ello, se diseñará un enfoque que logre simplificar esta complejidad para obtener un resultado aproximado, pero funcional.

Inicialmente se planeó llevar a cabo esta operativa con el módulo de clasificación semántica de textos de la Natural Language API del servicio Google Cloud, incluido en su plan gratuito, pero dicho módulo no es compatible, por el momento, con el idioma español. Posteriormente se exploraron otras alternativas: por ejemplo, el clasificador de textos de Spacy, que sí es compatible con el español, o los planes gratuitos de diversas herramientas comerciales que ofrecen una API REST pública con planes gratuitos de uso, como uClassify[1], MeaningCloud[2], TextRazor[3] o Dandelion[4]. No obstante, ninguna de ellas satisfacía nuestros requisitos, bien porque no proporcionaban un volumen de procesamiento suficiente para cubrir las necesidades previstas (quinientos textos al día es un límite habitual), bien porque la taxonomía utilizada para la clasificación no era lo suficientemente detallada

[1] `https://www.uclassify.com` (visitado: 17/06/2019)
[2] `https://www.meaningcloud.com/es` (visitado: 17/06/2019)
[3] `https://www.textrazor.com` (visitado: 17/06/2019)
[4] `https://dandelion.eu` (visitado: 17/06/2019)

(careciendo, por ejemplo, de la temática 'Política').

Por tanto, para discernir entre los *tweets* relevantes y los que no lo son, se diseñará e implementará un clasificador binario que determine la temática de los mismos, separándolos en las categorías "Política" y "No política". A continuación, se describe el proceso de diseño e implementación de este clasificador, así como las diferentes decisiones de desarrollo tomadas durante el mismo.

6.1. Aproximación.

Dado que contamos con un conjunto de clases predefinido, se ha optado por utilizar un enfoque supervisado, habitual en la clasificación de textos, y en contraste con el modelado de tópicos en texto (que suele utilizar técnicas no supervisadas para obtener las categorías en las que podrían segregarse los documentos de un corpus no anotado) [28]. El primer reto que se presenta es obtener un conjunto de datos anotados semánticamente adecuado, que nos permita entrenar un clasificador de las características deseables para el proyecto. El segundo, de mayor envergadura, es que este conjunto esté en español.

Actualmente existen escasos recursos que podrían ser utilizados para este tipo de tareas en nuestro idioma:

- El *workshop* TASS[5], organizado anualmente por la Sociedad Española para el Procesamiento de Lenguaje Natural, proporciona conjuntos de datos anotados de tweets orientados a su uso en clasificación de sentimiento en mensajes cortos. En particular, el dataset STOMPOL [29] incluye metadatos relativos a la semántica de los tweets, indicando la categoría semántica general a la que pertenecen (política, entretenimiento, etcétera). No obstante, este conjunto se orienta al minado de opinión de un partido político respecto a un tema concreto, por lo que todos los tweets están relacionados con política y no sirven a nuestro propósito.
- El conjunto de datos "*Twitter Dataset - 2015 Spanish General Election*" [30] recoge identificadores de tweets relacionados directamente con la campaña de las elecciones autonómicas de 2015. Además de presentar el mismo problema que el anterior, hay que señalar que no incluye los contenidos del tweet, sino que estos deben ser extraídos por otros medios a partir del ID proporcionado.

[5] `http://www.sepln.org/workshops/tass/` (visitado: 19/06/2019)

En ambos casos, además, se tratan de conjuntos de datos desfasados, que si bien podrían ofrecer un rendimiento aceptable, este sería posiblemente menor que si utilizáramos información actualizada y más cercana a los textos que efectivamente queremos clasificar. Desde su generación hasta la actualidad han surgido o cobrado relevancia nuevas formaciones políticas: por ejemplo, Ciudadanos, que precisamente daba su salto al panorama nacional en 2015, o Vox, que en las elecciones generales del 20 de diciembre de 2015 obtuvo un 0,23 % de los votos a nivel nacional y en la última convocatoria alcanzó el 10,26 %. Además, la evolución de la actualidad ha modificado los nombres propios en la primera línea de la política, así como los idearios y consignas de cada formación política, por lo que los elementos que caracterizan a la política hoy pueden no ser los que lo hacían entonces.

Una alternativa para contar con un conjunto de datos actualizado y adecuado para los datos sobre los que se trabajará en el proyecto sería anotar, manualmente, una muestra suficientemente grande de los *tweets* capturados. Por razón de tiempo y recursos, esta opción tampoco es viable a efectos prácticos.

Finalmente, se ha optado por una solución innovadora, automatizable y con una implementación asequible: utilizar comentarios de usuarios en noticias de medios digitales. Este tipo de mensajes presentan las siguientes características favorables a nuestro propósito:

- Suelen tener una corta longitud, análoga a la que pueden presentar los *tweets* con suficiente longitud como para exponer una opinión (es decir, que constan de más de unas pocas palabras).
- Presentan la misma naturaleza espontánea e informal que los *tweets*, ya que son generados por usuarios lectores del medio en cuestión, y no por redactores especializados.
- Están asociados a un período temporal y una temática definidas, determinados ambos aspectos por la noticia en la que se depositan, por lo que nos permite obtener un conjunto de datos semánticamente semejante al conjunto de textos a clasificar.

Por otro lado, debemos considerar otras características que no son deseables, y que deben ser tenidas en cuenta a la hora de utilizar los datos:

- Los medios escritos cuentan con una línea editorial definida, y por tanto existe un sesgo importante en el público que publicará comentarios en sus noticias. Para paliar este hecho, recogeremos comentarios de medios de diferente sensibilidad ideológica.
- A pesar de tener la misma naturaleza informal, los comentarios en medios digitales suelen tener una mejor calidad de redacción, debido

probablemente al público más adulto que los genera. Sin embargo, idealmente se debería corregir los errores gramaticales y ortográficos de los *tweets* haciendo uso de un procesamiento de lenguaje natural más avanzado que el planteado en este proyecto, lo que los acercaría a los datos obtenidos de los periódicos.

En este caso, se ha optado por recolectar comentarios correspondientes a noticias del mes de abril de 2018 publicadas en las ediciones digitales de tres periódicos de tirada nacional: El País, El Mundo y El Diario. En los tres casos se trata de periódicos con un buen nivel de difusión, lo que nos asegura tanto una buena cobertura de la actualidad (que no ofrecerían medios de menor tamaño) como un público suficientemente grande como para que su participación (comentarios) sea significativa.

Ninguno de los medios seleccionados pone a disposición del público sus contenidos para su descarga en forma de conjunto de datos ordenado, al menos dentro de nuestro conocimiento. Es decir, se requieren técnicas alternativas para obtener esta información de manera que pueda ser usada de forma automatizada, como es nuestro propósito. Para ello, haremos uso de la técnica de extracción de información conocida como *Web Scraping* o "raspado de páginas web" (en adelante, nos referiremos a ella por el vocablo inglés por ser el término más extendido). El *Web Scraping*[6] consiste en descargar u obtener el código fuente de una página o sitio web (parcial o completamente), con el fin de extraer información localizada en el mismo a través de distintas técnicas (exploración del árbol DOM, búsqueda de patrones, interpretación de tablas, copiado manual, etcétera). Este proceso puede ser llevado a cabo de forma manual o automatizada.

Cabe señalar un factor importante a la hora de llevar a cabo este tipo de operaciones, y es la naturaleza del sitio web que se desea "raspar". En este sentido, podemos distinguir dos tipos de página web:

- Estática: La página web se compone únicamente de código fuente estático; es decir la interfaz está generada únicamente mediante HTML y CSS. Puede implementar componentes de Javascript, pero estos se utilizan para controlar el comportamiento de la página web, y no para construir su estructura.

- Dinámica: En este tipo de páginas, parte de la estructura del documento web se genera dinámicamente en el lado del cliente: este recibe un documento HTML con la estructura básica, que después se enriquece mediante la ejecución de *scripts* Javascript para generar nuevo código HTML. Este enfoque es útil, por ejemplo, cuando para completar una

[6] `https://en.wikipedia.org/wiki/Web_scraping` (visitado: 20/06/2019)

página web es necesario consultar una base de datos, pues permite implementar un comportamiento asíncrono: se carga en cliente la parte estática de la interfaz mientras se recibe la respuesta, y cuando llega esta se modifica la web mostrada para reflejarla.

Los mecanismos tradicionales de *Web Scraping* realizan peticiones a los servidores y capturan el código HTML que este envía como respuesta. Mientras que en webs estáticas este código está completo y refleja el estado final de la web, en una página dinámica solo se recibirá el "esqueleto", probablemente sin la información que es de nuestro interés. Por ello, en estos casos es necesario recurrir a otros mecanismos. Por ejemplo, utilizar un *driver web*: este es un software que simula el funcionamiento de un navegador web, con capacidades que exceden la mera interpretación de código HTML estático, como la ejecución de código Javascript o la interacción con los elementos de la interfaz como lo haría un usuario humano. De esta manera, es posible interactuar con la página hasta que esta cargue por completo la información deseada y entonces, ya sí, capturar el código HTML que conforma la página y extraer dicha información.

6.1.1. Consideraciones legales.

Antes de entrar a detallar el proceso de extracción implementado, es conveniente indicar alguna de las implicaciones de este tipo de procesos automatizados sobre contenidos de autoría ajena. En general, el *Web Scraping* no constituye una actividad ilícita de por sí [31], aunque hay casos de uso en los que sí podría serlo. A pesar de su accesibilidad, las páginas web no se consideren fuentes públicas de datos (Informe 0342/2008 de la Agencia Española de Protección de Datos)[32][33], y por tanto los datos que contienen no pueden ser utilizados con cualquier finalidad.

Hay principalmente dos factores que podrían introducir el carácter de ilegalidad en esta actividad:

a. *Naturaleza de los datos capturados.* No es habitual, pero es posible que en determinadas circunstancias se acceda a datos personales protegidos por el Reglamento de Protección de Datos vigente. El hecho de que estos datos puedan ser públicamente accesibles no obsta para que deba cumplirse con la legislación vigente, que establece una serie de preceptos independientemente de la publicidad del dato [34], al no considerarse esta una forma de consentimiento tácito para el tratamiento.

   En este caso, nos encontraríamos con que se han obtenido estos datos sin consentimiento expreso de sus titulares, y, probablemente, sin el menor conocimiento por parte de estos, lo que daña los derechos de

acceso, rectificación, cancelación o limitación del tratamiento. Además, se estarían utilizando para un fin diferente para el que se accedió a su cesión (suponiendo siempre que el sitio web del que han sido extraídos no esté infringiendo también el Reglamento).

b. *Quebrantamiento de la propiedad intelectual.* Incluso si los datos recabados no son datos personales sujetos a una protección especial, los contenidos de una página web pueden estar registrados como propiedad intelectual de la empresa administradora del medio y/o de sus respectivos autores (periodistas, expertos o editores, por ejemplo). La publicidad de estos contenidos tampoco habilita a su uso libre por parte de terceros, como quedó demostrado tras el cierre de Google News en España [35].

   Google News es un servicio de indexación automatizada de noticias que recoge los enlaces a los contenidos de múltiples medios de información, clasificándolos y agrupándolos para facilitar el acceso a la información de sus usuarios. Los diferentes medios emisores de noticias reclamaron sus derechos de propiedad intelectual, al considerarlos violentados por Google al recabar y mostrar en su propio servicio partes de los contenidos enlazados. Como respuesta, se reformó la Ley de Propiedad Intelectual (Real Decreto Legislativo 1/1996, de 12 de abril), estableciendo la obligatoriedad de una compensación económica por el uso de dichos contenidos a los medios creadores de los mismos, popularmente conocida como "tasa Google".

En el caso que nos ocupa, estaremos accediendo exclusivamente a información publicada en medios de comunicación, reconocida como fuente de acceso público, por lo que no debemos preocuparnos por el acceso y utilización de estos contenidos en el ámbito de nuestro proyecto.

En todos los casos se ha consultado el fichero *robots.txt* [7] de cada sitio. Este fichero establece los permisos de automatización de procesos sobre la estructura del sitio web a los distintos agentes que pudieran emprenderla, siendo el caso más frecuente la indexación de las páginas en el buscador de Google. Este fichero suele ser accesible en el *home* de cada sitio web, agregando '/robots.txt' a la URL de acceso.

Los tres periódicos disponen de este fichero, y en ellos puede apreciarse tres características principales:

- Bloquean el acceso a todos sus contenidos a las herramientas de *scraping* masivo más habituales, como MSIECrawler, Zealbot o Webcopier.

[7] `https://www.humanlevel.com/diccionario-marketing-online/robots-txt` (visitado: 24/06/2019)

- Bloquean de forma generalizada el acceso automatizado a una lista de URL concretas. Generalmente, este bloqueo se produce como consecuencia de una sentencia judicial que impide, por ejemplo, la difusión o indexación de una noticia por suponer un atentado al honor y la propia imagen de los mencionados en la misma, o en cumplimiento del conocido como derecho al olvido.
- Restricción de acceso a determinados subdominios.

Por tanto, al tratarse de un *scraping* focalizado y puntual, podemos considerarnos habilitados para acceder, aunque sea de forma automatizada, a los contenidos de los medios consignados, siempre y cuando no obtengamos ninguna de las noticias cuyo acceso se encuentra bloqueado.

6.2. Construcción de datasets.

El proceso de extracción de información de cada uno de los periódicos consta de tres fases idénticas, que se describen a alto nivel a continuación. Los detalles de implementación, que no se cubrirán en esta memoria, se describen en los cuadernos Jupyter asociados en los que se ha llevado a cabo el desarrollo de los scripts de extracción de información.

6.2.1. Diseño del almacenamiento de datos de periódicos.

Dentro del directorio dedicado a los datos de artículos de prensa se creará un subdirectorio para cada uno de los periódicos consultados. Dentro de cada uno de ellos se definirán, a su vez, dos subdirectorios (tres en el caso de El Mundo), que describimos a continuación:

- Directorio de artículos: *articulos*. Contiene los directorios que agrupan los ficheros fuente correspondientes a los artículos publicados durante el día, a razón de un directorio por día. Cada directorio se nombrará acorde al día capturado, con el nombre *AAAA-MM-DD*.
- Directorio de comentarios: *comentarios*. Tiene una estructura interna idéntica a la de *articulos*.
- Directorio de portadas: *portadas*. Exclusivo de El Mundo. Contiene los ficheros HTML de las portadas de las diferentes ediciones (matutina, vespertina y nocturna) del período de tiempo considerado.

El nombre asignado a los diferentes ficheros permitirán la identificación unívoca de sus contenidos:

- Los ficheros de artículos se nombrarán con el identificador único del artículo asignado por el periódico de procedencia, extraído de la URL de acceso. Los ficheros tendrán la extensión *.html*.
- Los ficheros de comentarios se nombrarán con el identificador del artículo en el que se publicaron. Los comentarios se almacenan como objetos dentro de un fichero JSON, a razón de un fichero por cada artículo.
- Los ficheros de portadas se nombrarán con el siguiente criterio: *portada_AAAA-MM-DD[m/t/n].html*, indicando mediante el carácter correspondiente la edición que contiene.

6.2.2. Recopilación de enlaces.

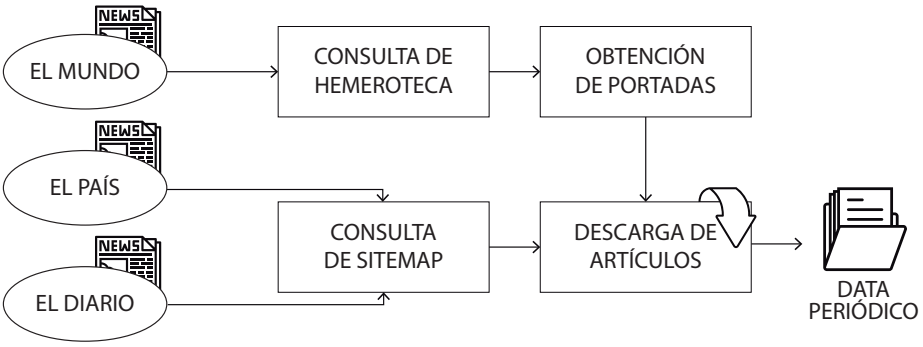


Figura 6.1: Descripción del proceso de extracción de artículos.

Una de las condiciones descritas en la Sección 6.1 incluye una correspondencia "temporal" para asegurar que la realidad de los tópicos abordados es similar, y no ha cambiado fruto de la evolución que sufre la actualidad con el tiempo. Para ello, es necesario elaborar un mecanismo de filtrado de la información que vamos a obtener, en lugar de utilizar técnicas habituales en *web scraping* como el paseo aleatorio o la construcción del mapa del sitio (*sitemap*) en base a la estructura de hiperenlaces entre las páginas que lo conforman. De esta manera, podremos extraer artículos publicados en un rango de fechas de interés que nos asegure la pertinencia de los datos capturados.

En este sentido, tanto El Diario como El País proporcionan listados de enlaces a todas las noticias publicadas durante un mes concreto. Estos listados se obtienen en formato XML, y están ubicados dentro del dominio de los periódicos:

- En el caso de El País, el fichero XML se encuentra localizado en una ubicación que codifica el período con que se corresponde mediante una estructura de directorios. Por ejemplo, para el mes de abril se utilizaría `https://elpais.com/sitemaps/2019/4/sitemap_1.xml`.

- El Diario ubica sus *sitemaps* en el directorio raíz, codificando el nombre del mismo de acuerdo al período que abarca. Las páginas correspondientes al mes de abril se localizan en `https://www.eldiario.es/sitemap_contents_2019_04.xml`.

En el caso de El Mundo, no hemos logrado obtener un recurso similar, por lo que la lista de enlaces deberá ser construida por otros medios. Para ello, se recurrirá al servicio de hemeroteca digital que proporciona el diario, y que da acceso a las portadas de la versión digital del periódico para cada una de las ediciones de la mañana, la tarde y la noche para cada uno de los días. Aplicando de nuevo *web scraping*, podemos realizar un sencillo proceso en dos fases que nos proporcione el resultado deseado:

1. Obtención de las portadas: Accedemos al servicio de hemeroteca de El Mundo y localizamos los enlaces a las portadas correspondientes a cada uno de los días que se incluyen en el período temporal objetivo.

2. Obtención de los artículos: Extraemos los enlaces a los artículos cuyo titular se encuentre en la portada en alguna de las ediciones diarias del periódico para los días recogidos.

El resultado final obtenido es análogo al que conseguimos extrayendo la información de los ficheros XML que proporcionan el resto de diarios considerados.

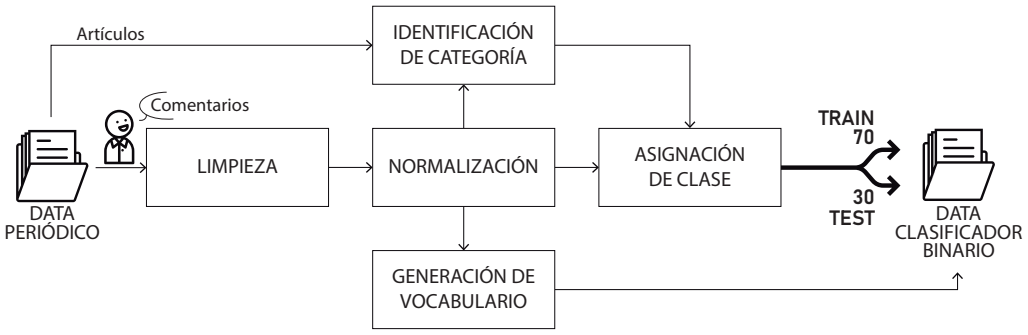


Figura 6.2: Descripción del proceso de recuperación de información de artículos.

6.2.3. Extracción de datos de los artículos.

Una vez contemos con los enlaces a los artículos, podremos automatizar la descarga de los mismos y la posterior extracción de la información de interés que contienen. El objetivo será obtener información de artículos y comentarios:

1. De los artículos se obtendrán tanto el texto de la noticia como metadatos de interés para la caracterización de los comentarios asociados a la noticia.

2. De los comentarios se recogerá el texto, la fecha y las reacciones cosechadas entre el resto de usuarios, tanto positivas como negativas.

Aunque se planteó el número de reacciones como una medida de la relevancia de cada comentario, la captura de esta información no ha sido posible en el caso de El Mundo: para visualizar la información de los votos positivos y negativos de los comentarios, es necesario haber votado antes en cada uno de ellos. Por tanto, no se utilizará esta información para el procesamiento posterior.

En las Tablas 6.1 y 6.2 se describe en detalle la información recabada. Cabe señalar que parte de la información objetivo es generada de forma dinámica durante la carga del sitio web, por lo que deberán utilizarse técnicas adaptadas a este tipo de sitios web. En nuestro caso, utilizaremos la librería Selenium desplegada sobre un *web driver* basado en el navegador Google Chrome, que nos permitirá navegar las páginas de los artículos de noticias para cargar con éxito los contenidos generados dinámicamente, como los comentarios de algunos de los periódicos. Apoyaremos la interpretación del código HTML generado mediante la librería BeautifulSoup4, habitual en entornos Python para este tipo de operaciones.

El proceso de extracción debe realizarse de forma respetuosa con el sitio web de origen, asegurando que no se produzca una sobrecarga de peticiones o se bloquee la conexión por ser identificada como *spam* o peligrosa. Para ello, se impondrán tiempos de espera adecuados que espacien las peticiones, aunque esto incrementará inevitablemente el tiempo necesario para completar el proceso.

6.2.4. Normalización y vectorización de textos.

Se utilizará el mismo proceso de normalización de textos descrito en el Capítulo 5, Subsubsección 5.4.3.3. Tras este proceso, dispondremos de textos que presentan, en su conjunto, una menor variabilidad que nos permitirá caracterizar mejor la información que contienen. No obstante, este resultado

Atributo	Descripción
Título	El título del artículo, disponible tanto a través de los metadatos del artículo como del árbol DOM (elemento h1)
Descripción	Entradilla del artículo, disponible en los metadatos del artículo o a continuación del título en el árbol DOM
Palabras clave	Términos relevantes para el contexto del artículo. Obtenidas de los metadatos del artículo, o bien del elemento correspondiente de la web
Categorías	Categoría en la que se clasifica el artículo. Obtenida de los metadatos del artículo, aunque más adelante se complementa con el nombre del subdominio extraído de la URL
Texto	Texto íntegro de la noticia, eliminando saltos de línea y enlaces
Fecha	Fecha de publicación del artículo, tal y como se proporciona en los metadatos
URL	Dirección de acceso al artículo

Tabla 6.1: Atributos de los artículos.

Atributo	Descripción
Texto	Texto íntegro del comentario
Fecha	Fecha de publicación del comentario
Reacciones positivas	Número de reacciones positivas (sistema de voto)
Reacciones negativas	Número de reacciones negativas (sistema de voto)

Tabla 6.2: Atributos de los artículos.

no es compatible con los algoritmos de aprendizaje que se utilizarán, por lo que es necesario un nuevo proceso de transformación: la *vectorización del texto*.

La vectorización de un texto consiste en la transformación de un conjunto de palabras en un vector dentro de un espacio vectorial de palabras. Consideramos un espacio vectorial n-dimensional, cuyas n dimensiones están determinadas por un vocabulario de n palabras diferentes. Cada texto considerado será equivalente a un vector dentro de este espacio vectorial. Este vector tendrá un valor no nulo en cada dimensión cuya palabra correspondiente se encuentre en el texto, y ceros en el resto de dimensiones. Las dimensiones no nulas adoptan un valor diferente en función del planteamiento elegido:

- Si únicamente es relevante la presencia o no de la palabra en el texto, el valor no nulo será 1, independientemente del número de apariciones

de dicha palabra en el texto.

- Si el número de apariciones es un dato de interés, puede utilizarse este dato como valor no nulo, de manera que cada componente del vector almacena el número de apariciones de cada palabra del vocabulario (cero si no aparecen).

La primera decisión de diseño que debemos adoptar se corresponde con la creación del vocabulario. Para ello, sería posible reunir todas las palabras distintas que aparecen en nuestro corpus y codificar cada texto del mismo en este espacio vectorial. No obstante, esto plantea varios problemas:

- El espacio resultante tendría una dimensionalidad excesiva. Los textos considerados son espontáneos y de diversa naturaleza, por lo que se debe asumir que se utilizarán gran variedad de términos y expresiones. Este factor se acentúa por la posibilidad de errores.

- Los vectores generados serían muy dispersos, ya que se van a procesar textos cortos de apenas unas decenas de palabras. En general, las librerías tienen mecanismos para reducir la huella de memoria de estos vectores optimizando su representación, pero en todo caso tendrán un cierto impacto.

- Todas las dimensiones tendrán la misma importancia. No obstante, dado que estamos estudiando un ámbito concreto, cabe esperar que algunos términos sean más relevantes o representativos que otros, por lo que sería deseable poder contemplar este hecho en el proceso.

Existen soluciones para abordar los dos primeros inconvenientes, como la utilización de *hashing* para no almacenar el vocabulario completo o almacenar vectores de gran tamaño. Otra opción, que también aborda el tercer problema, es reducir la dimensionalidad utilizando como vocabulario únicamente los términos más relevantes para el corpus considerado.

Un enfoque posible para abordar esta última solución es el análisis TF-IDF del corpus documental. El valor TF-IDF (*term frequency–inverse document frequency*) normaliza las frecuencias absolutas de las palabras en el corpus con respecto a todas las palabras del mismo. De esta manera, se reduce el peso que tienen las palabras que aparecen muy frecuentemente, y por tanto proporcionan menos información que otros términos presentes únicamente en subconjuntos más pequeños del corpus completo.

El cálculo de este valor tiene múltiples variantes [36], pero en este caso

nos decantaremos por la implementada en la librería *scikit-learn*[8]:

$$tfidf(t, d, D) = tf(t, d) * idf(t, D) \tag{6.1}$$

$$tf(t, d) = f(t, d) \tag{6.2}$$

$$idf(t, D) = \log \frac{|D|}{df(t, D)} + 1 \tag{6.3}$$

De acuerdo con la Ecuación 6.2, para calcular el TF-IDF de una palabra en un documento se utiliza la frecuencia absoluta del término en el mismo, pudiendo suavizarse mediante la aplicación del logaritmo fijando los parámetros adecuados. Por otro lado, en la Ecuación 6.3 se ve que la frecuencia inversa de documento es el logaritmo de la cardinalidad del corpus, o el número total de documentos, dividido por el número de documentos en que aparece la palabra t. La adición de 1 a su valor es una decisión de implementación, de manera que los términos con TFIDF igual a cero no sean ignorados. También existe una versión suavizada de esta expresión, que suma 1 a numerador y denominador para evitar divisiones entre cero en casos particulares.

Junto con los valores de puntuación TF-IDF, se establece un tamaño máximo de vocabulario con el que podemos controlar la dimensionalidad del espacio vectorial resultante. De esta manera, solo las n palabras con un mayor valor TF-IDF (es decir, las más representativas) se contemplarán en el vocabulario y la vectorización resultantes. Como nuestro objetivo es abarcar un ámbito restringido de la realidad – la política –, pero que a su vez comprende múltiples dimensiones, no podemos fijar un tamaño de vocabulario demasiado pequeño. Dada la variedad de términos encontrados en el corpus (unos 135.000), se ha fijado un valor arbitrario de 15.000 dimensiones para capturar suficiente detalle sin incrementar la dimensionalidad en exceso, puesto que a partir de este valor las pruebas realizadas no mostraban un incremento apreciable en el rendimiento de los clasificadores, y sí en el tiempo de entrenamiento de los modelos. Se han evaluado diferentes tamaños de vocabulario (en concreto, 300, 500, 2.500 5.000 y 10.000) para tratar de trabajar únicamente con los términos más relevantes, pero la diversidad de los textos utilizados impedía que vocabularios más pequeños pudieran caracterizar correctamente los textos.

Adicionalmente, se impondrán filtros de frecuencia de términos: los términos muy poco frecuentes en el corpus documental (frecuencia absoluta de 5 o menos) y los términos que aparezcan muy frecuentemente (90 % o

[8] `https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfTransformer.html` (visitado: 03/07/19)

más) se descartarán automáticamente para aliviar la presencia de *outliers* y ruido.

6.2.5. Asignación de clases.

Tras la vectorización, contamos con un vector de características que describe cada uno de los ejemplos recabados, pero no se les ha asignado aún una clase. Para ello, utilizaremos la información de metadatos que hemos extraído de cada noticia para identificar su temática (bien política, bien cualquier otro tema). En concreto, se explorarán tanto la categoría o categorías capturadas, como las palabras clave asociadas al artículo, entre las que buscaremos el término 'política'. Este enfoque es perfectamente válido para El Diario y El País: el primero archiva esta clase de artículos bajo la sección 'Política', mientras que el segundo utiliza un sistema de etiquetas o *tags* bien definido para caracterizar sus artículos. Además, se agregará como categoría el identificador de sección ubicado en la URL del artículo.

El caso de El Mundo es menos inmediato: carece de sección 'Política', ya que organiza este tipo de noticias atendiendo a su localización geográfica ('España', 'Madrid' o 'Castilla y León' son ejemplos de secciones del periódico), y no tiene un conjunto de etiquetas consolidado, sino que utiliza palabras relevantes del cuerpo del artículo. Dado que es la principal fuente de comentarios, se ha implementado un mecanismo de búsqueda más exhaustivo en las palabras clave del artículo, donde podemos encontrar etiquetas como 'política - partidos'.

Una vez identificados los artículos que tratan de política, propagaremos esta consideración a todos los comentarios publicados en el mismo. Aunque es posible que algunos de los comentarios no guarden relación ni con el contenido de la noticia ni con la temática política en general (por ejemplo, *spam* o publicaciones de *bots*), el tamaño de la muestra de textos debería limitar su posible impacto en el rendimiento final del clasificador.

6.2.6. Descripción del corpus generado.

El conjunto de datos está formado por un total de 26.201 artículos de noticias (tras la exclusión de aquellas páginas web capturadas que no se corresponden con noticias) y 232.909 comentarios, distribuidos como se reporta en la Tabla 6.3. Estos artículos son todos aquellos publicados en la edición digital de los periódicos elegidos a lo largo del mes de abril de 2019, con el fin de utilizar información a priori pertinente para la clasificación de los textos objetivo. Los comentarios recogidos son los existentes en las

noticias a fecha de 26 de julio de 2019, momento en que se llevó a cabo la captura.

	Artículos			Comentarios		
Periódico	El Mundo	El País	El Diario	El Mundo	El País	El Diario
Número	1.853	6.045	18.303	127.749	77.139	27.746
Peso (MB)	8,37	31,00	60,70	44,00	26,30	13,60

Tabla 6.3: Estadísticas del corpus de artículos y comentarios.

El Diario dispone de una cantidad mucho mayor de artículos, debido probablemente a que publican de forma automática numerosas notas de prensa de las agencias de información que utilizan, como Efe. Estas son noticias muy breves, que en ocasiones motivan un artículo posterior con un mayor nivel de desarrollo, pero que quedan en cualquier caso en la hemeroteca. Sin embargo, esta gran actividad no se traduce en una gran interacción de los usuarios, acumulando apenas 28.000 comentarios en todo el mes. En este sentido, El Mundo es el diario digital con más participación de sus usuarios, a pesar de ser el que menos noticias publica.

Siguiendo el criterio establecido, se han asignado las clases *'política'* y *'no política'* con el resultado indicado en la Tabla 6.4.

Periódico	Artículos		Comentarios	
	Pol	No pol	Pol	No pol
El Mundo	444	1.409	65.976	62.000
El País	1.668	4.377	35.332	41.807
El Diario	7.910	10.393	11.388	16.406
Total	10.022	16.179	112.696	120.213

Tabla 6.4: Distribución de clases para los diferentes periódicos.

La distribución de clases resultante es bastante equilibrada en lo que se refiere a los comentarios. Los artículos etiquetados como política, por otra parte, son minoría ante el resto de temáticas; esto nos indica que, en general, los artículos relacionados con política despiertan en mayor medida el interés de los lectores, pues el número medio de comentarios por artículo es superior.

Finalmente, se ha dividido aleatoriamente el conjunto global de datos en dos conjuntos, con una proporción de 70 %-30 %, para utilizarlos en las tareas de entrenamiento y evaluación de los modelos, respectivamente.

6.3. Desarrollo del clasificador.

En esta Sección se discute el proceso de generación del modelo de clasificación binaria, que abarca la selección de un algoritmo de aprendizaje, la optimización de sus parámetros principales y el entrenamiento y evaluación del modelo final sobre el conjunto de evaluación separado anteriormente.

6.3.1. Selección del algoritmo.

Se han probado un total de cuatro clasificadores, configurados con las opciones por defecto, utilizando dos subconjuntos del conjunto de entrenamiento definido previamente. En este caso, se ha dividido dicho conjunto en una proporción de 70 %-30 %, con el fin de formar un conjunto de entrenamiento y otro de validación. El objetivo es evaluar el comportamiento general de los clasificadores con nuestros datos de validación, de manera que podamos elegir el mejor clasificador a priori para, después, llevar a cabo un proceso de ajuste o *tuning* sobre la configuración de parámetros del clasificador. Los resultados obtenidos, reflejados en la Tabla 6.5, se ofrecen en términos de porcentaje de acierto (*accuracy*) y área de la curva (AUC, *Area Under Curve*) ROC[9], una métrica adecuada para problemas de clasificación binaria cuando la distribución de clases está balanceada [37][38].

Clasificador	Acc (%)	AUC
Bayes Multinomial	69,23	69,12
Gradient Boosting	64,51	63,74
Random Forest	68,35	68,20
SVC (lineal)	67,52	67,43

Tabla 6.5: Valores de precisión y AUC obtenidos en la prueba preliminar.

Salvando los ofrecidos por el algoritmo GBT, todos los resultados obtenidos presentan valores muy similares para las métricas elegidas. En la Tabla 6.5 se aprecia que los diferentes modelos rondan el 68 % tanto de tasa de acierto como de área bajo la curva ROC, lo que, combinado con los datos de la Tabla 6.6, nos permite evaluar el número de asignaciones correctas e incorrectas para cada una de las clases.

[9]La curva ROC, o *Receiver Operating Characteristic curve*, es una representación de la proporción de falsos positivos frente a la proporción de verdaderos positivos, representado como una curva en un eje bidimensional. Cuanto más cercano sea su valor a 1 (clasificación correcta en todos los casos), mejor será el clasificador.

Bayes	0	18.282	7.089	Random	0	18.263	7.108
Multinomial	1	7.959	15.581	Forest	1	8.372	15.168
Gradient	0	21.429	3.942	SVC	0	17.759	7.612
Boosting	1	13.413	10.127	(lineal)	1	8.271	15.269

Tabla 6.6: Matrices de confusión de la prueba preliminar.

Clasificador	Clase	*Precision*	*Recall*	F1
Bayes Multinomial	0	0,70	0,72	0,71
	1	0,69	0,66	0,67
Gradient Boosting	0	0,62	0,84	0,71
	1	0,72	0,43	0,54
Random Forest	0	0,69	0,72	0,70
	1	0,68	0,64	0,66
SVC (lineal)	0	0,68	0,70	0,69
	1	0,67	0,65	0,66

Tabla 6.7: Resultados obtenidos de *precision*, *recall* y F1.

Todos los clasificadores son ligeramente más efectivos en la predicción de ejemplos de clase negativa, principalmente debido a su valor de *recall* (ver Tabla 6.7). Este implica que interpretan una mayor proporción de ejemplos negativos como tales, que los de clase positiva como positivos. Este hecho es especialmente acusado en el GBT, que demuestra un cierto sesgo hacia la clase negativa: acierta un número mucho mayor de ejemplos negativos que el resto, pero el número de ejemplos negativos que yerra es también muy superior. Dado que nuestro interés se encuentra, precisamente, en los ejemplos de clase positiva (pues los negativos se descartarán tras la clasificación cuando apliquemos el modelo), un valor tan bajo de *recall* nos fuerza a rechazar este algoritmo.

Del resto, Naive Bayes se presenta como el más efectivo, superando en un 1 % al siguiente modelo en prácticamente todas las métricas. No obstante, este es un clasificador muy sencillo sobre el que apenas podremos aplicar operaciones de optimización. Además, la diferencia es tan pequeña que es probable que se vea superado por otros modelos más complejos, una vez se hayan ajustado.

Finalmente, *Random Forest* y SVC (*Support Vector Classifier*) presentan pequeñas ventajas en *precision* y *recall*, respectivamente, por lo que procederemos a ajustar ambos para comprobar su capacidad de mejora al evaluar diferentes parámetros a los de por defecto.

6.3.2. Configuración y entrenamiento.

Una vez elegido los algoritmos que utilizaremos, es necesario tratar de optimizar su rendimiento por medio de los parámetros de configuración que proporcionan. Esta optimización o *tuning* se llevará a cabo mediante una exploración de rejilla o *grid search*, una técnica habitual para la optimización de modelos. Consiste en la evaluación del modelo fijando diferentes valores para cada parámetro objetivo, de manera que se verifiquen todas las posibles combinaciones para los valores dados. La combinación que alcance el mejor resultado de acuerdo al criterio de evaluación indicado será la elegida como el modelo óptimo. En este caso, se utilizará el área bajo la curva ROC. La implementación de la validación cruzada se hará mediante el módulo `model_selection.GridSearchCV` de la librería Scikit-Learn.

Evidentemente, esta exploración no es exhaustiva, y no ofrece la mejor configuración posible del algoritmo para el conjunto de datos utilizado, puesto que no evalúa combinaciones fuera de las suministradas manualmente. Además, se trata de un proceso costoso que supone entrenar y evaluar cada modelo generado, por lo que es imprescindible elegir valores adecuados para cada parámetro. Cabe señalar que en total se realizarán $\sum_{i=0}^{m}\sum_{j=0}^{n_i} p_i^j$ entrenamientos de modelo, donde m es el número de parámetros y n_i el número de valores que adopta cada uno de los parámetros.

Para mejorar la fiabilidad de los resultados, se utilizará una validación cruzada (*cross validation*) sobre el conjunto de los datos de entrenamiento: la validación de cada modelo se llevará a cabo sobre una partición en n subconjuntos, de manera que se obtenga el resultado promedio de evaluar el modelo sobre cada uno de los subconjuntos utilizados, entrenándolo sobre los subconjuntos restantes. De esta manera, se reduce la influencia de un posible sobreajuste del modelo a una parte de los datos, algo posible si usáramos dos subconjuntos únicos para entrenar y validar los modelos generados. El SVC se ha evaluado utilizando una partición de tamaño 10, mientras que el RF, atendiendo a su mayor complejidad y tiempo de entrenamiento, se hará con 5 particiones.

Para el algoritmo *Random Forest*, los parámetros configurables más relevantes son los siguientes:

- Número de estimadores (*n_estimators*): Número de árboles con que se construirá el *ensemble*. Valores bajos generan modelos más simples y rápidos, pero menos estables al enfrentarse a errores.
- Profundidad máxima (*max_depth*): Número de niveles máximo que pueden tener los árboles construidos para el *ensemble*. Cuando se alcanza este valor, los nodos no pueden dividirse.

- Número mínimo de ejemplos para dividir un nodo (*min_samples_split*): Número mínimo de ejemplos alojados en un nodo para que este pueda dividirse en dos nodos.

Para el SVC lineal, la lista es más reducida:

- C: Es un parámetro de penalización al error de clasificación para cada uno de los ejemplos. Si fijamos un valor alto de C, el algoritmo buscará un hiperplano con un margen de error más pequeño (será menos tolerante al error), aumentando la complejidad y el riesgo de sobreajuste al conjunto de entrenamiento; en cambio, valores pequeños de C relajan esta condición y penalizan menos el error en la clasificación, por lo que existe el riesgo de no poder capturar la complejidad subyacente en los datos.
- Número máximo de iteraciones (*max_iter*): Número de iteraciones que podrá realizar el algoritmo de optimización para hallar el hiperplano adecuado. No es tanto un parámetro del SVC como del método de cálculo del plano divisor, pero puede tener una gran importancia en conjuntos de datos complejos.

En la Tabla 6.8 se listan los valores evaluados para los parámetros indicados, así como el valor óptimo encontrado por el mecanismo de validación cruzada.

Clasificador	Parámetro	Valores	Valor óptimo
Random Forest	*n_estimators*	25, 125, 200	200
	max_depth	50, 100, 200	200
	min_samples_split	5, 25	25
SVC Lineal	*C*	0,001, 0,01, 0,1, 1, 10	0,1
	max_iter	1000, 2500	1000

Tabla 6.8: Parámetros utilizados para la validación cruzada.

Los modelos arrojan unos valores medios de AUC ROC de 69,22 % y 67 %, respectivamente, para las configuraciones de parámetros señaladas como óptimas, lo que sitúa el *Random Forest* ligeramente por encima del SVC y demuestra una mayor capacidad de mejora con el *tuning*. Sería interesante llevar a cabo pruebas en una horquilla mayor de valores para caracterizar mejor el comportamiento de los clasificadores, así como realizar una fase de *fine-tuning* en torno a los parámetros, variando estos en un intervalo muy pequeño alrededor del máximo localizado, pero probablemente la ganancia sería leve.

6.3.3. Evaluación.

Finalmente, se evaluará cada modelo obtenido sobre el conjunto de evaluación separado al comienzo del proceso, para caracterizar su comportamiento frente a un conjunto de ejemplos no vistos hasta el momento. En este caso, analizaremos su rendimiento en términos de los parámetros que hemos utilizado hasta el momento. Con la configuración elegida, los modelos generados obtienen los resultados reseñados en las Tablas 6.9 y 6.10.

Clasificador	Acc (%)	AUC
Random Forest	69,79	69,55
SVC (lineal)	69,34	69,20

Tabla 6.9: *Accuracy* y AUC obtenidos tras la evaluación.

Clasificador	Clase	*Precision*	*Recall*	F1
Random Forest	0	0,68	0,78	0,73
	1	0,72	0,62	0,66
SVC (lineal)	0	0,69	0,74	0,71
	1	0,70	0,64	0,67

Tabla 6.10: Resultados obtenidos de *precision*, *recall* y F1 tras la evaluación.

A la luz de estos resultados, vemos que *Random Forest* es efectivamente superior en *accuracy* y *AUC*, así como en *precision* para ambas clases. Sin embargo, el tiempo de entrenamiento del modelo es muy superior (aunque este no es un factor muy relevante, dado que se lleva a cabo una vez), y las diferencias son pequeñas con respecto a SVC. Este último alcanza mejores resultados (de nuevo, por una ventaja muy pequeña) en términos de *recall* y F1 para la clase positiva. Por esta razón, nos decantaremos por este modelo para su aplicación en el proyecto: al obtener un mayor nivel de *recall* para la clase positiva, maximizará el número de verdaderos positivos para esta clase; la disminución de precisión, además de poco significativa, puede ser abordada en futuras fases del proyecto (por ejemplo, el filtrado manual eliminará muchos falsos positivos), mientras que la pérdida de datos en esta fase supondría la pérdida total de dicha información.

En cualquier caso, los valores obtenidos, no siendo excesivamente bajos, tampoco proporcionan unos resultados verdaderamente fiables. Esta fase sería una de las principales candidatas a ser revisada en el futuro, pues es, probablemente, la más crítica en todo el flujo de trabajo, pues un clasificador

de gran calidad proporcionaría siempre resultados pertinentes, reduciendo la importancia de las operaciones de filtrado posteriores. En la Capítulo 8 abordaremos algunas cuestiones en relación a este aspecto.

Capítulo 7

Predicción de resultados.

En este Capítulo se describe la fase de predicción de datos, para la que se explotarán los conjuntos de datos generados tras el proceso de extracción, limpieza y transformación que se ha desarrollado a lo largo de los capítulos anteriores.

7.1. Enfoque.

Dado el carácter exploratorio de esta propuesta, se planteará un mecanismo sencillo de predicción con vistas a evaluar el interés de nuestro flujo de trabajo de cara a realizar predicciones en el caso de uso expuesto. En el desarrollo futuro de esta prueba de concepto, sería interesante explorar formas más complejas de explotar los datos; por ejemplo, incorporando a las predicciones parte de la información que descartamos en el planteamiento actual, como la localización geográfica.

Este componente debería ser capaz de predecir una distribución del voto de la población completa de votantes en una fecha futura, observando la distribución de "voto" de los usuarios capturados a lo largo de un determinado período de tiempo. Para ello, utilizaremos mecanismos de regresión para analizar las tendencias para cada una de las categorías identificadas – los partidos políticos –, y así obtener un valor predicho que se corresponderá con la proyección de dichas tendencias hacia el futuro. Tal y como se estableció en la Subsección 5.4.4, se analizarán las cinco primeras fuerzas políticas y se agrupará el resto en una única categoría.

Como en este punto del procesamiento contamos con los *tweets* capturados y el sentimiento que portan, es necesario transformar esta información para que nos permita caracterizar a cada uno de los usuarios de los cuales hemos

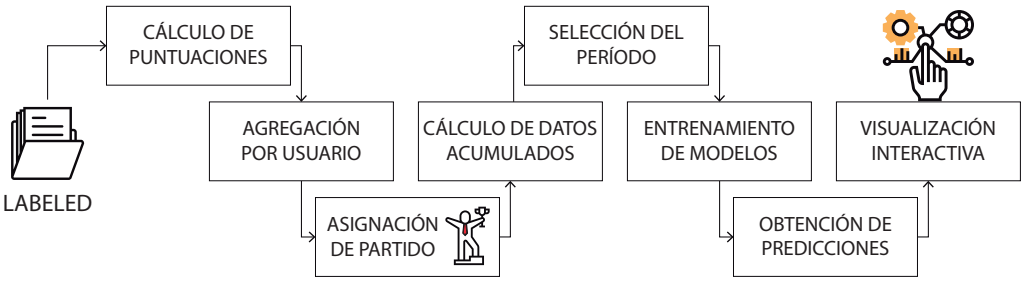


Figura 7.1: Descripción del proceso de obtención de predicciones.

capturado algún mensaje, y así calcular la distribución del voto sobre la muestra de usuarios con que contamos. De esta manera, podrá apreciarse la evolución de cada una de las opciones políticas a lo largo del tiempo. Para conseguir esto, definimos la siguiente secuencia de pasos:

1. Se calcula, para cada *tweet*, un vector de puntuaciones.
2. Se agrupan los *tweets* por usuario, y se calcula el vector de puntuaciones del usuario como la suma de los vectores de los *tweets* que ha publicado.
3. Se determina la puntuación máxima, asumiendo que se corresponde con el partido por el que el usuario siente mayor simpatía, y se anota el usuario con esta información.
4. Se calcula la distribución normalizada (en tanto por uno) del "voto" de los usuarios a los diferentes partidos.
5. Se genera esta distribución para cada uno de los días de los que disponemos de datos.

Aunque a grandes rasgos este es el proceso seguido, es necesario detenerse en algunos de los puntos para discutir algunas decisiones tomadas para su implementación.

En primer lugar, el vector de puntuaciones de cada *tweet* se calcula atendiendo a los *trending topics* que tiene asociados y la información del sentimiento que contiene. Recordemos que disponemos de una lista de *trending topics* manualmente seleccionados como relevantes, y distribuidos entre los diferentes partidos políticos bajo el criterio de que "los *trending topics* asociados a un partido son positivos para el mismo", de manera que si un *tweet* positivo se adhiere a una tendencia asociada a un partido, se incremente el valor de puntuación del partido, y si el *tweet* es negativo, se reduzca. Si un *tweet* tiene múltiples *trending topics* asociados, por cada uno de ellos aportará la misma puntuación a cada uno de los partidos que se asocien a dichas tendencias.

Así, para cada posición del vector (para cada partido) se calcula un valor siguiendo la siguiente fórmula:

$$puntuación_{partido} = \#tts_{partido} * sentiment\_score * sentiment\_magnitude \tag{7.1}$$

Donde *sentiment_score* y *sentiment_magnitude* son la polaridad del sentimiento del texto y su intensidad, respectivamente. El que un *tweet* sume o reste puntuación a un partido dependerá exclusivamente de la polaridad: los mensajes negativos tienen una polaridad menor que cero, y mayor que cero en el caso de los mensajes positivos. Los otros dos valores implicados son siempre positivos o cero.

En segundo lugar, se ha decidido trabajar con los datos acumulados hasta la fecha considerada, en lugar de con los datos individuales de cada día. Esto es, la estimación medida para el día 1 comprende los datos para el día 1; los datos del día 2 son los datos agregados del día 1 y el día 2; y así sucesivamente. De esta manera abordamos dos cuestiones importantes en las circunstancias del desarrollo:

- Observando los datos disponibles, hemos apreciado que existen días en los que no hay datos sobre algunos de los partidos, lo que provocaba fluctuaciones irreales en la evolución de la intención de voto. Utilizando datos acumulados, las variaciones diarias nos permiten ir refinando progresivamente una tendencia global en lugar de observar instantáneas del voto.

- Esta forma de proceder es más realista, pues permite contemplar, para cada usuario, su tendencia ideológica en el pasado y tenerla en cuenta para caracterizarlo en el presente. Es posible que un votante acérrimo de un partido pueda expresarse en buenos términos acerca de alguna propuesta de un partido rival, pero eso no implica que, ese día, esta persona haya cambiado de filiación política.

Para poder prever posibles cambios de opinión (algo inhabitual en el votante, pero no tanto en los políticos, visto el transfuguismo que hemos visto n las últimas fechas), se implementará un *mecanismo de olvido* que permita tener en cuenta, para un día dado, únicamente los n días anteriores, en lugar de todos los días previos a dicha fecha de los que se dispone de datos.

Finalmente, es importante explicar el criterio utilizado para la caracterización del usuario. En este caso, manejamos dos posibles aproximaciones:

- Sumar los vectores de puntuación de todos los *tweets* emitidos por el usuario, y escoger la opción política con mayor puntuación.

- Identificar el partido con mayor puntuación para cada *tweet*, contar el número de mensajes del usuario que se han asignado a cada partido, y escoger el partido al que más mensajes, y más positivos, haya dedicado.

Como se ha indicado, implementaremos el primer enfoque. El segundo daría lugar a situaciones en que un usuario puede haber emitido, por ejemplo, tres mensajes ligeramente positivos sobre un partido y uno muy positivo sobre otro, de tal manera que la puntuación del positivo fuera mucho mayor que la suma de los tres primeros en conjunto, y aún así elegir el partido con los tres mensajes.

Ambos presentan un problema derivado de la gestión de los "ceros": si un usuario tiene un mensaje negativo sobre un partido, todos los demás serán cero, y por tanto máximo. Como no se ha pronunciado respecto a los mismos, no es correcto asumir que vaya a votar a uno de ellos en concreto (de hecho, se escogería aquel que se postule primero como máximo). Tampoco sería adecuado buscar la máxima puntuación distinta de cero, pues podría darse el caso de que un usuario haya emitido exclusivamente mensajes críticos contra un partido, por lo que este sería el único que tendría un valor distinto de cero, aunque negativo, y por tanto sería el elegido como destinatario del voto del usuario. Por tanto, descartaremos aquellos votos cuyo máximo sea 0 o menor. Podría modelarse el concepto de "votar al mal menor" relajando la condición de voto estrictamente positivo en caso de que un número mínimo de posiciones del vector sean distintas de cero (por ejemplo, si se ha expresado en malos términos de todas las formaciones políticas, sería posible asignarle el de la puntuación "menos mala"), pero además de incrementar la complejidad, no se corresponde con la realidad.

Así, para cada día dispondremos de una distribución de clases que toma en cuenta todos los días del período anterior previsto (de toda la serie histórica, o limitado por el "tiempo de memoria" establecido). Por tanto, podemos modelarlo en forma de serie temporal y realizar predicciones sobre ella. En la siguiente sección se discutirán brevemente los mecanismos de regresión que se utilizarán.

7.2. Modelos de predicción.

Se llevarán a cabo pruebas con tres algoritmos de regresión diferentes para disponer de múltiples enfoques que nos permitan caracterizar las tendencias y analizar la validez de los resultados. A continuación se describen los algoritmos utilizados:

Ajuste lineal.

El algoritmo más básico trata por determinar la recta que minimiza el error cuadrátrico medio con respecto a cada uno de los ejemplos suministrados. Esto nos dará una tendencia lineal, que si bien no nos proporcionará una predicción ajustada de la distribución de intención de voto, sí debería permitirnos identificar la tendencia de la percepción de cada uno de los partidos en cada momento. En este caso, el mecanismo de olvido resultará de gran utilidad, puesto que nos permitirá apreciar una tendencia más instantánea, al ritmo de los acontecimientos, si utilizamos un valor de memoria bajo; o más a largo plazo, más orientada al *big picture*, fijando este parámetro en un valor más alto.

La implementación del algoritmo utiliza la clase `LinearRegression` de la librería Scikit-Learn.

Ajuste polinomial.

Este enfoque es similar al anterior, solo que en lugar de utilizar una recta se intenta aproximar a un polinomio de un grado determinado. De hecho, el ajuste lineal es un caso particular de ajuste polinomial, en el quel grado del polinomio es 1. La dificultad de este regresor es dar con el grado de polinomio adecuado: debe tenerse en cuenta las limitaciones de representación que ofrece un polinomio. Un polinomio $p(x)$ de grado n presentará n extremos relativos, es decir, n cambios de carácter (creciente o decreciente). Para valores superiores o inferiores de x, una vez se han superado todos los valores de x en los que se alcanza un extremo relativo, la curva crecerá o decrecerá infinitamente sin cambiar su carácter. Pocas tendencias en la realidad se ajustan a esta característica, por lo que en todo caso la distancia a la que podremos predecir con esperanza de acierto (con respecto a los ejemplos sobre los que ajustamos el polinomio) será limitada.

No obstante, su poder de caracterización es mayor que el modelo lineal, y será tanto mayor cuanto mayor sea el grado del polinomio (corriendo el riesgo de sobre-ajustar el modelo a los ejemplos suministrados). Aún así, deben juzgarse las tendencias generadas con cierto criterio.

La implementación del algoritmo también utiliza la clase `LinearRegression` de la librería Scikit-Learn, aunque previamente se transforman las características de los ejemplos proporcionados por medio de la clase `PolynomialFeatures`.

Random Forests. El algoritmo *Random Forests* divide el espacio en que se ubican los ejemplos en subespacios por medio de una combinación (*ensemble*) de clasificadores binarios en forma de árbol binario, agrupando los ejemplos similares. De esta manera, cada nueva instancia descenderá a

través del árbol siendo clasificada de forma sucesiva por los clasificadores binarios situados en los nodos hasta llegar un nodo hoja, que proporcionará el valor predicho para la instancia introducida.

Su implementación es similar a los anteriores, pero utilizando la clase `RandomForestRegressor` de la librería Scikit-Learn. Utilizaremos un número arbitrario de estimadores (*estimators*) o clasificadores binarios de 100, ajustándonos al valor por defecto establecido por la librería.

7.3. Representación visual de resultados de las predicciones.

A diferencia del resto del flujo de trabajo, se llevará a cabo la implementación de este componente en un cuaderno o *notebook* de Jupyter, que proporciona una interfaz basada en HTML apta para la correcta visualización y exploración de los datos a través de la interactividad que permiten por medio de sus *widgets*.

Los resultados se presentarán en forma de gráfica de líneas, a razón de una línea por opción política (ver Figura 7.2). Aunque que los valores de los ejemplos están normalizados en valores entre 0 y 1, esto puede no ser así para los valores predichos. Por tanto, aunque para representar los ejemplos sería correcto acotar los valores del eje Y a este rango para facilitar la visualización de las líneas, en este caso no impondremos un límite al eje, puesto que nos permite caracterizar la fuerza de la tendencia, por un lado, y determinar cuándo una predicción no tiene sentido (para polinomios altos, es habitual obtener valores de predicción del orden de cientos o miles). Los valores del eje X sí estarán limitados a los días de los que tenemos datos o estamos realizando predicciones, utilizando como etiquetas el número de día al que se refiere cada registro. Como todos los días incluidos se encuentran en el mes de abril de 2019, no se ha considerado necesario utilizar la fecha de cada registro como etiquetas 'DD/MM/AAAA', indicando únicamente el número del día.

El estilo de las líneas se adaptará para facilitar la legibilidad del gráfico, particularmente en dos aspectos:

- El color de cada línea se asignará de acuerdo al color típicamente asociado a cada una de las formaciones políticas (a saber: rojo para el PSOE, azul para el PP, naranja para Ciudadanos, morado para Unidas Podemos y verde para Vox). Aunque los tonos de color no son exactamente los mismos que los corporativos, la asociación es suficientemente clara como para que esto no dificulte en modo alguno

su interpretación. La categoría que agrupa al resto de formaciones políticas recibe un color sin un significado particular que asegure que se distingue adecuadamente del resto.

- La forma de la línea cambiará cuando los datos reflejados se correspondan con un dato consolidado o con una predicción:
  - Los datos calculados en base a los registros con que se cuentan se muestran como una línea continua y ligeramente más gruesa, dando a entender que se trata de datos consolidados.
  - Los valores predichos se representan como puntos individuales del color del partido.

  Para facilitar la visualización de las tendencias, se unirán los valores predichos mediante una línea adicional punteada y más fina, que servirá de soporte visual para seguir el recorrido de los puntos para cada partido. Si no, solo se vería una nube de puntos de colores y sería mucho más difícil interpretar el gráfico.

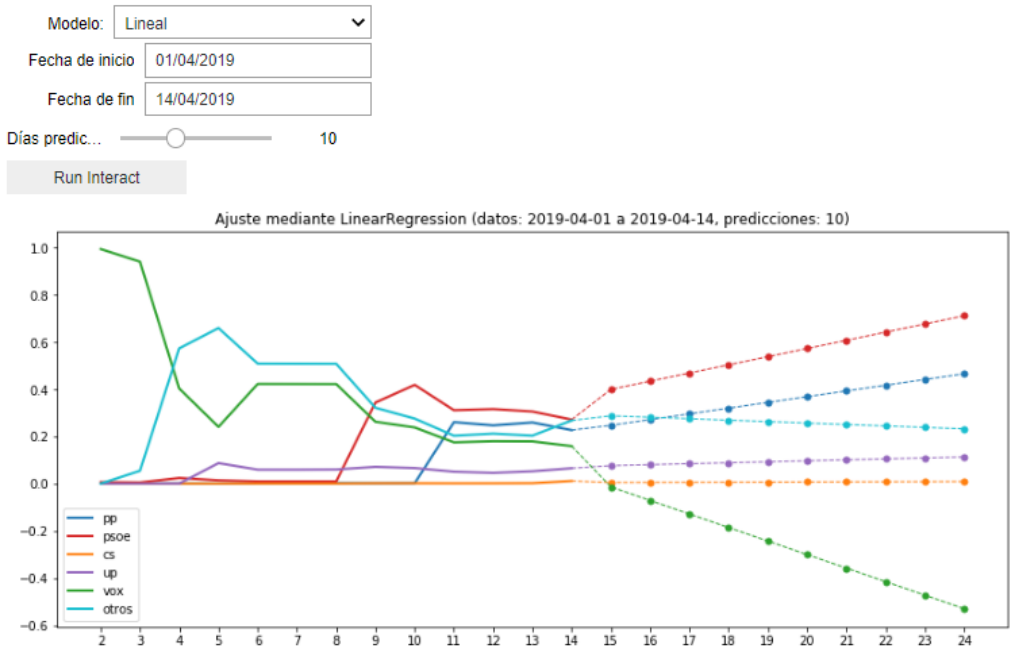


Figura 7.2: Captura del gráfico interactivo en el cuaderno Jupyter.

Dado que las operaciones de cálculo de las distribuciones de voto son considerablemente pesadas dada la cantidad de datos, se precalcularán los datos correspondientes a cada día, y se permitirá la exploración de los datos

por medio de un gráfico interactivo. Este gráfico permitirá al usuario filtrar los datos para ajustar la visualización a sus intereses, para lo cual se le darán las siguientes opciones:

- Seleccionar el período de datos utilizado para el entrenamiento de los modelos.
- Seleccionar el algoritmo de regresión aplicado.
- Seleccionar el número de días a predecir a partir del día siguiente al último del que se disponen datos.

El mecanismo de olvido determina la manera en que se realizan los cálculos de los datos diarios, por lo que no será configurable por medio de la consola interactiva de los gráficos. Se proporcionará un mecanismo de carga y salvado de los datos precalculados, junto con un volcado de los datos completos del mes utilizando un valor de tiempo de olvido de 5 días, y otro sin factor de olvido.

7.4. Resumen de resultados.

En esta sección se explicarán los resultados obtenidos tras las operaciones descritas en el capítulo, así como las conclusiones que podemos extraer tanto de los datos como del enfoque con que se ha abordado el componente de predicción. Utilizaremos el sistema de visualización desarrollado para ilustrar nuestras explicaciones.

7.4.1. Análisis de las predicciones.

Como este desarrollo se ha llevado a cabo sobre los datos de un evento electoral cuyo desenlace ya conocemos, podemos utilizar los resultados reales para evaluar el grado de acierto obtenido por los diferentes algoritmos. En la Tabla 7.1 se pueden ver los porcentajes de voto alcanzados por cada una de las fuerzas políticas incluidas en el estudio[1].

De la misma manera, podemos utilizar otras encuestas realizadas de manera tradicional como referencia para comprobar si las tendencias identificadas en los datos se corresponden con las que se han dado de forma efectiva en la sociedad. Por ejemplo, el informe de seguimiento que realizó el periódico *El Español* el último día en que es legal podían publicar encues-

[1]Fuente: `https://resultados.elpais.com/elecciones/2019/generales/congreso/` (visitado: 04/09/2019)

Partido	% de voto
PSOE	28,68
PP	16,70
Ciudadanos	15,86
Unidas Podemos	14,31
Vox	10,26
Otros	14,19

Tabla 7.1: Resultados de las elecciones generales del 28 de abril de 2019.

tas, una semana antes de la fecha electoral[2]. En particular, utilizaremos el diagrama de evolución de la intención de voto incluido en la Figura 7.3.

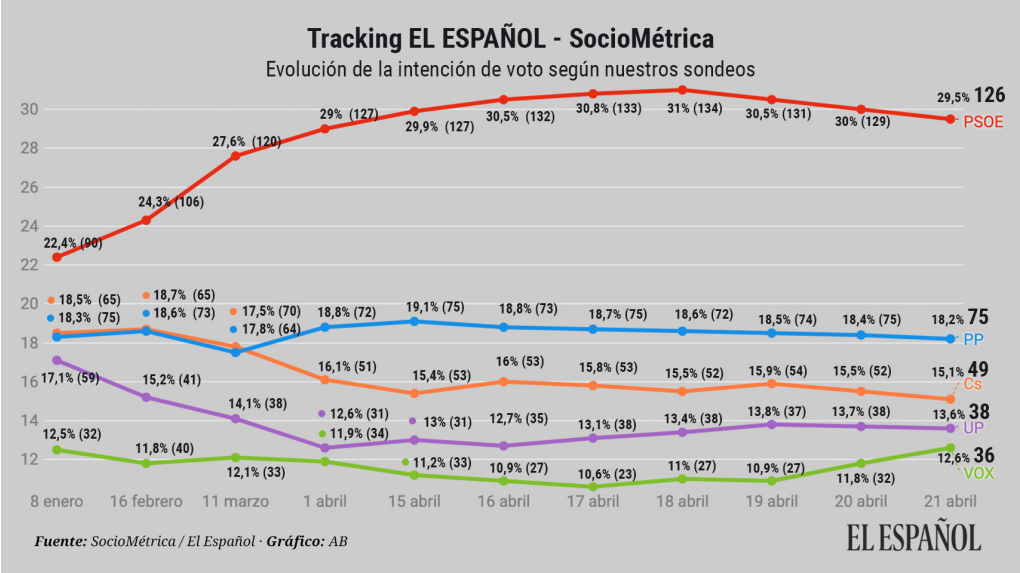


Figura 7.3: Diagrama de seguimiento de la intención de voto en las semanas previas a las elecciones. Fuente: El Español.

En la Figuras 7.4 se aprecia que ninguno de los mecanismos de regresión proporciona valores realmente válidos para proponer una distribución de voto acorde con los resultados ocurridos. Las razones para este hecho son múltiples, y las discutiremos en la siguiente subsección, pues se derivan del enfoque seguido y las condiciones del análisis. Sin embargo, podemos observar que sí están presentes algunas de las tendencias que efectivamente se dieron a lo largo del período pre-electoral.

[2] `https://www.elespanol.com/espana/20190421/bajada-psoe-vox-tracking-permite-publicar-espanol/392711034_0.html` (visitado: 09/07/19)

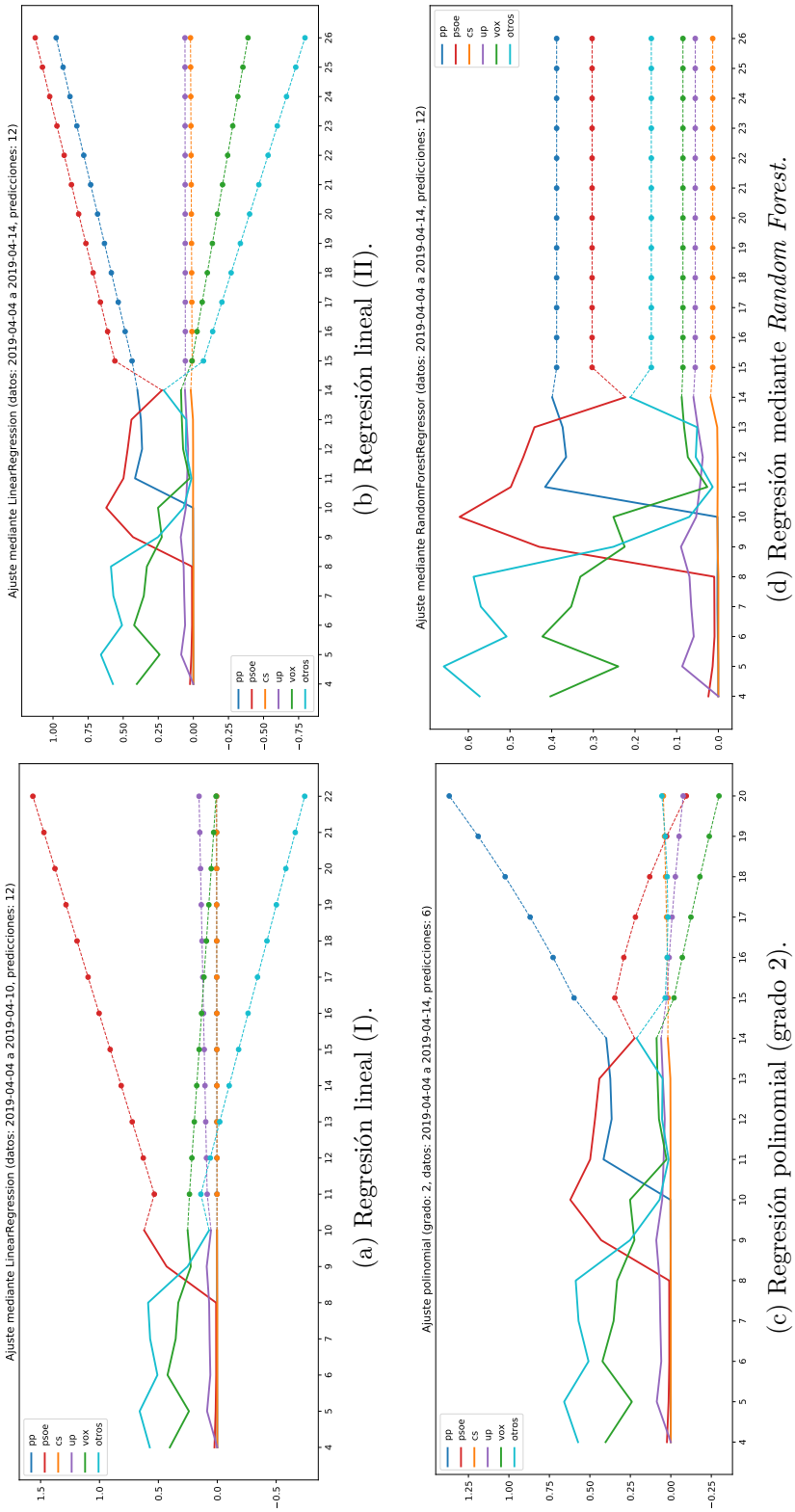
En la Subfigura 7.4a descubrimos unos de los problemas persistentes que se ha observado a la hora de desarrollar el componente de predicción: la falta de datos. Entre los días 4 y 9 no disponemos de información de las tres primeras fuerzas políticas, y en todo el período utilizado para calcular la tendencia inicial, dos de ellas no hacen acto de presencia (PP y Ciudadanos), por lo que sus predicciones son constantes y con valor 0. El PSOE presenta una clara tendencia alcista, pero se debe más al hecho de que no hubiera datos disponibles hasta el día 9 que por una tendencia verdadera, como se refleja en la Subfigura 7.4b, en la que se incluyen los datos de los días posteriores.

Al aumentar el período de datos conocidos, disponemos de más información para determinar las tendencias existentes. A medida que van entrando nuevos partidos al "reparto de votos", los valores encontrados van tornándose menos extremos, y se puede apreciar que los valores comienzan a fluctuar en la mitad inferior del intervalo [0,1]. Previsiblemente, cuanta más información podamos utilizar, más ajustada será la estimación que obtengamos cada día de la intención de voto, y con mayor fundamento podremos hacer predicciones para el futuro. De hecho, el resultado final predicho en esta figura coincide, si no en el valor, sí en la posición de los partidos, con el resultado que se dio en las elecciones. Con la excepción de Ciudadanos, de quien no disponemos de datos.

En la Subfigura 7.4c, correspondiente al ajuste polinomial, las predicciones divergen rápidamente y exceden el intervalo [0,1] el primer día predicho. Resulta evidente que un polinomio de grado 2 es insuficiente para capturar la complejidad de la curva definida por los datos conocidos. Sin embargo, diversas pruebas con grados mayores presentan comportamientos similares o más extremos, y resultan muy sensibles a las oscilaciones en los últimos días de la serie utilizada para entrenar los modelos. Probablemente, el ajuste polinomial no sea adecuado para representar curvas con tanta variación en su carácter; al menos, no utilizando el mismo grado de polinomio para todas las curvas. Implementando un grado diferente para cada curva, seguramente podríamos obtener representaciones mucho más ajustadas, aunque con toda probabilidad igual de incapaces de predecir a medio y largo plazo.

Finalmente, el resultado que arroja el *Random Forest* Subfigura 7.4d resulta sorprendente, puesto que realiza predicciones constantes para cada una de las curvas definidas a partir de los datos conocidos. Este extraño comportamiento se debe a la naturaleza del algoritmo de regresión, como referiremos más adelante, y nos llevará a excluirlo de las siguientes pruebas.

A continuación analizaremos el resultado de utilizar todos los datos hasta la víspera de las elecciones (ver Figura 7.5). En primer lugar, podemos



(a) Regresión lineal (I).

(b) Regresión lineal (II).

(c) Regresión polinomial (grado 2).

(d) Regresión mediante *Random Forest*.

Figura 7.4: Resultados de regresión sin olvido. Predicción con datos parciales.

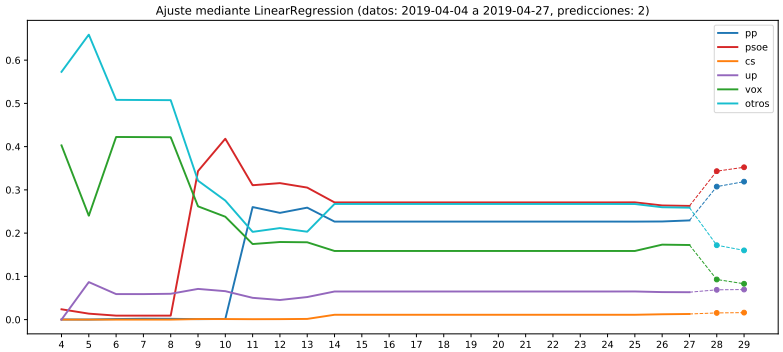
observar que el modelo de regresión lineal genera predicciones con pendientes menos pronunciadas, probablemente debido a la meseta correspondiente a los datos perdidos. Tras este parón, apenas se observan fluctuaciones en las líneas de datos consolidados: esto puede significar que, o bien no hay apenas datos en esos días previos a la fecha electoral, o bien que el impacto de los datos de un día apenas es capaz de modificar la tendencia impuesta por la información histórica. Aunque esto último podría considerarse positivo (en el sentido de que los datos tienden a estabilizarse, lo que nos permite obtener información consolidada de la muestra reunida), también indica que nuestro enfoque es poco adaptable a los cambios, especialmente cuando procesamos períodos de tiempo muy largos.

Por otro lado, el polinomio de grado 2 también proporciona predicciones más matizadas, pues ya no presenta la misma tendencia inmediata al infinito. De nuevo, esto puede ser causado por la meseta: la curva de ajuste será una parábola muy abierta, con crecimiento o decrecimiento suave en los valores del eje Y. El mismo comportamiento se aprecia en el polinomio de grado 4, aunque las predicciones realizadas siguen sin resultar factibles, ya que difieren de las representadas en la Figura 7.3.

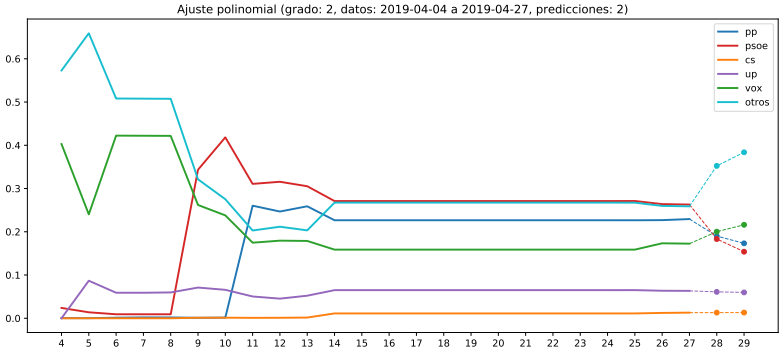
Analicemos ahora los resultados obtenidos de utilizar el mecanismo de olvido. En este caso abordaremos únicamente las predicciones en base a los primeros días del mes, pues la meseta en los datos entre el 16 y el 25 anula cualquier prueba que podamos llevar a cabo con el factor de olvido activado: al llegar los primeros datos tras el hueco, el sistema ha olvidado toda la información histórica. Salvo que se utilice un factor de olvido de valor alto, por ejemplo de 15 días, el caso “con olvido” degenera prácticamente al caso de “días individuales”, cuyos resultados son muy pobres. Usando un valor de olvido muy grande, obtendremos unos resultados similares a los reflejados en el caso acumulado “sin olvido”.

Los resultados obtenidos de la predicción sobre los primeros días se muestran en la Figura 7.6, utilizando un factor de olvido de cuatro días. Se puede apreciar una mayor fluctuación de las curvas, cuya variación ya no depende solo de los datos que se les agrega, sino también de aquellos que se olvidan. No obstante, la información remanente parece ser suficiente para caracterizar la tendencia de los datos: las rectas de predicción son muy similares a las obtenidas por el modelo de regresión lineal sin uso de olvido.

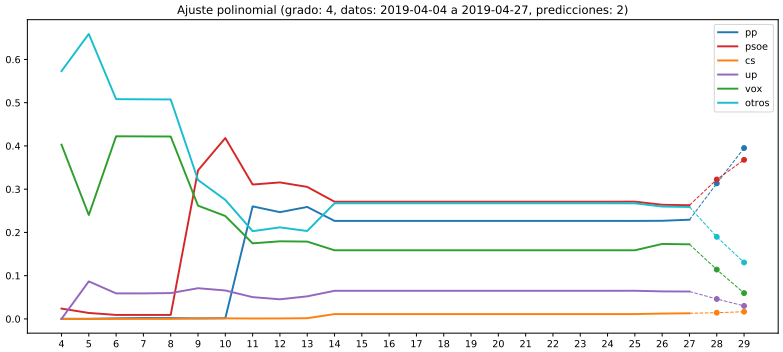
En este caso, observamos de nuevo que el modelo predice, con carácter cualitativo, la posición final de todos los partidos. De hecho, pronostica una caída de Vox que no existe en la Figura 7.3 , y que sin embargo sí se produjo en la realidad (los resultados electorales reales otorgaron a Vox un resultado mucho más bajo que el que preveían las encuestas hasta el momento).



(a) Regresión lineal.



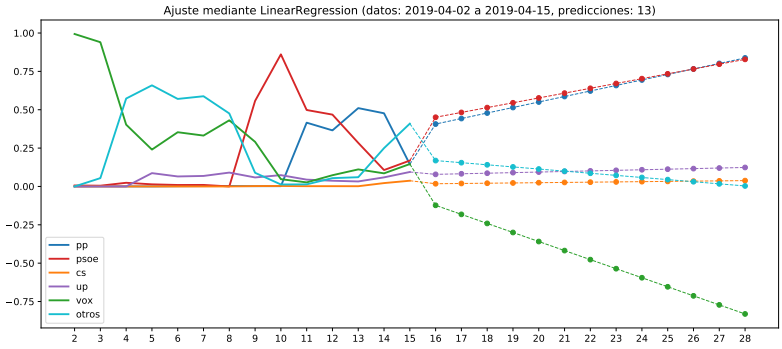
(b) Regresión polinomial (grado 2).



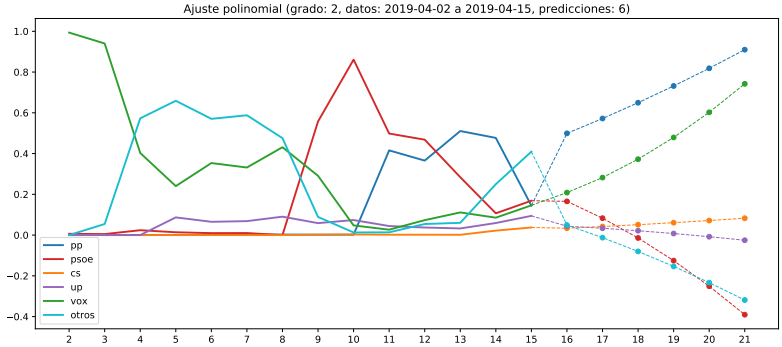
(c) Regresión polinomial (grado 4).

Figura 7.5: Resultados de regresión sin olvido. Predicción con datos 'completos'.

Por otro lado, el algoritmo de ajuste polinomial presenta un crecimiento moderado en el extremo, lejos de la rápida deriva hacia el infinito al usar las curvas completas.



(a) Regresión lineal.



(b) Regresión polinomial (grado 2).

Figura 7.6: Resultados de regresión con olvido tras 4 días. Predicción con datos parciales.

7.4.2. Evaluación del enfoque.

Tras las pruebas realizadas, podemos concluir que los algoritmos utilizados no permiten alcanzar el objetivo de predecir una distribución concreta de votos en la implementación propuesta. No obstante, el ajuste lineal ha demostrado ser capaz de proporcionar un resultado cualitativo bastante ajustado a la situación real al final de las elecciones, una vez cuenta con suficientes datos para obtener un análisis significativo.

Tal y como se preveía, el modelo lineal es incapaz de proporcionar valores ajustados de predicción, ya que asume un comportamiento monótono y constante que, evidentemente, no se corresponde con una situación que pudiéramos suponer cierta en la realidad. Además, a partir de cierta distancia de predicción arroja resultados absurdos (como porcentajes de voto negativos, o superiores al 100 %). Sin embargo, en combinación con el mecanismo de olvido puede permitirnos caracterizar el *momentum* de un partido en cada momento, dando información, por ejemplo, sobre el impacto que un evento

podría tener en la percepción del partido.

El algoritmo *Random Forest* presentó un comportamiento extraño en los primeros experimentos. Tras una breve investigación, descubrimos que se debía a la propia naturaleza del algoritmo [39]: dado que, en el fondo, el clasificador divide el espacio en el que se disponen los ejemplos de entrenamiento mediante sucesivas divisiones binarias del mismo, esta partición presentará un mayor grado de división en el espacio que media entre los diferentes ejemplos proporcionados. No obstante, en el espacio que les circunda el espacio se dividirá en grandes "zonas", por lo que cualquier ejemplo nuevo que aparezca en este espacio exterior en una determinada dirección obtendrá siempre la misma predicción.

Las series temporales es un caso claro de esta situación. El parámetro de entrada para la predicción es un número, equivalente a una fecha, que forma parte de una escala lineal de una dirección. Por tanto, todas las posibles instancias para las que se solicita la predicción estarán en la misma "zona" exterior. Es decir, no puede utilizarse este algoritmo para la predicción de valores *en el futuro*, aunque sí se demuestra muy efectivo en la predicción de valores *dentro* del espacio delimitado por los ejemplos del entrenamiento. Por ejemplo, podríamos utilizarlo para "reconstruir", de forma aproximada, los datos perdidos entre el día 16 y el día 25, pero no para predecir los valores del día 28 antes de que se alcance este.

Algo parecido hemos observado para el algoritmo de interpolación polinomial. La curva con la que se aproxima la tendencia de los datos podrá predecir valores de forma ajustada entre los ejemplos de entrenamiento, pero al intentar proyectar la curva hacia el futuro (o hacia el pasado), lo más probable es que se encuentre ya en un estado monotónico y crezca o decrezca hacia el infinito con mayor o menor rapidez. Incrementar el grado del polinomio no surtirá efecto, pues probablemente se dedicará ese extra de complejidad a ajustar más la curva a los ejemplos proporcionados, provocando un fuerte sobreajuste del modelo de regresión a los datos de entrenamiento, y llevando a una pobre capacidad de generalización. Sin embargo, sería interesante explorar dos posibles líneas de mejora: el uso de diferentes grados de polinomio para las distintas curvas, y la normalización de las predicciones en el rango objetivo. Este segundo cambio podría asegurar la convergencia de las predicciones a un valor concreto dentro del rango: a medida que las predicciones van haciéndose cada vez mayores, también lo será el rango que las contiene; por tanto, escalar este rango al intervalo [0,1] podría generar resultados interesantes.

Por otro lado, la utilización de los datos acumulados en lugar de analizar cada día por separado resulta prometedora. A medida que se dispone de más

datos, las curvas parecen tender a estabilizarse en torno a una proporción de "voto", con lo que con suficientes datos de usuarios podríamos llegar a obtener curvas poco fluctuantes, en las que el ajuste lineal pudiera proporcionar resultados mucho más fiables, al menos en el corto plazo.

En esa situación, el mecanismo de olvido adquiriría una relevancia mucho mayor; en particular, en el aspecto de la carga computacional del proceso. Recordemos que cada día debemos procesar los datos correspondientes a todos los demás días, por lo que el factor de olvido actuaría como limitante ante la recursividad de este cálculo. Por otro lado, si cada día contamos con una gran cantidad de datos para construir una muestra significativa, resulta innecesario remontarnos excesivamente hacia atrás para recopilar suficientes datos. Además, permitiría caracterizar mejor los cambios que se producen día a día, pues sin el factor de olvido, como hemos visto, es posible que la información histórica tenga tanto peso que los nuevos datos apenas logren modificarla. En este sentido, sería conveniente estudiar otros mecanismos de olvido, como un factor de decaimiento (*decay*) para dar una mayor relevancia a los nuevos registros frente a los antiguos, sin que estos pierdan toda su importancia.

Por lo tanto, una de las prioridades para mejorar este componente pasaría por conseguir más y mejores datos; es decir, mejorar todas las fases anteriores del flujo de trabajo. En este caso, tanto la pérdida inesperada de buena parte de los datos como la poca cobertura que hemos logrado capturar sobre los diferentes partidos políticos, nos han impedido profundizar en las posibilidades de este componente de predicción. Sin embargo, se han conseguido identificar numerosas oportunidades de mejora, que referiremos en el siguiente capítulo junto con nuestras conclusiones sobre el proyecto en su conjunto.

Parte III

Conclusiones finales.

Capítulo 8

Conclusiones y trabajo futuro.

8.1. Conclusiones.

A lo largo de esta memoria se ha presentado la documentación técnica del proyecto desarrollado como Trabajo Fin de Máster, consistente en un sistema de predicción de intención de voto para un evento electoral determinado en base a información recogida en tiempo real de Twitter. El proyecto planteaba diversos retos, para los que se ha tratado de desarrollar propuestas que condujeran a una solución satisfactoria de los mismos. A continuación, repasaremos las actividades realizadas y evaluaremos si se han conseguido alcanzar los objetivos que enunciamos al comienzo del proyecto.

En primer lugar, se diseñó e implementó un componente de captura de información de la red social Twitter, aprovechando la API pública de que dispone para acceder a la información que se publica en ella en tiempo real. Se planteó un proceso de captura que incluía la definición de un filtro, con un doble objetivo: tratar de reducir el caudal del flujo de datos, manteniendo solo la información de mayor relevancia, con el fin de mejorar la escalabilidad de nuestra solución; y dar pie a una posible generalización de nuestro enfoque gracias a la automatización de la selección de información, en una fase posterior del proceso, aprovechando los *trending topics* capturados. Sin embargo, esta aproximación se demostró ineficaz en su implementación actual. Si bien el segundo filtrado, que se hace por medio de la clasificación de los *trending topics* a partir de los *tweets* que les pertenecen, ha demostrado ser útil a la hora de extraer tendencias relevantes de entre la lista completa obtenida, el filtro que imponemos al *stream* de Twitter podría ser excesivamente restrictivo y eliminar una gran cantidad de información útil.

Esto se ha puesto de manifiesto, en especial, en las últimas fases del

proyecto: en muchas ocasiones encontrábamos que, sencillamente, no disponíamos de suficientes datos para asegurar un buen funcionamiento de los componentes que utilizábamos. Por tanto, una de las principales acciones de mejora que plantearemos irá en la dirección de evaluar criterios alternativos de acceso a la API para mejorar la calidad de la información capturada.

Otras decisiones relativas al proceso de captura han presentado un mayor grado de éxito. Por ejemplo, la captura periódica de *trending topics* con la frecuencia establecida en 10 minutos nos ha permitido caracterizar con detalle la evolución de tendencias en nuestro país. El criterio de almacenamiento inicial de los datos ha permitido el procesamiento de los datos sin problemas incluso en equipos con capacidades reducidas, aunque seguramente debamos revisarlo si introducimos cambios en los criterios de captura del *stream* de datos Finalmente, cabe señalar que el componente de captura desarrollado se ha mostrado robusto a lo largo del período en que se ha utilizado, especialmente en comparación a la primera iteración de este módulo. Al principio, los errores de reconexión y de sobrepasado de los límites de la API eran frecuentes. Tras una revisión de estos límites e imponer en el código restricciones para asegurar que estos se respetaban, la captura transcurrió sin ningún problema hasta el día 16 de abril. La información de log no recogió ningún evento relacionado con la conexión a la API ni ningún error del componente, por lo que asumimos que se trató de un error puntual de la máquina de captura, como un apagado de mantenimiento o un reinicio de emergencia, coincidiendo con el período vacacional de Semana Santa. Esta hipótesis queda respaldada por el hecho de que se mantuvo el componente capturando, de forma ininterrumpida, durante todo el mes de mayo sin que se produjera ningún corte o error.

Como parte del proyecto también se desarrolló un clasificador binario de textos, con el fin de suplir a la herramienta que habíamos previsto para llevar a cabo la labor de caracterización semántica de los *tweets*. Al constatar que los recursos disponibles para la creación del clasificador eran muy limitados, decidimos explorar una nueva vía para conseguir este objetivo, y que pasaba por extrapolar la información que podíamos extraer de un tipo de texto que comparte muchas características con los textos que queremos clasificar, los *tweets*, como se ha indicado en el capítulo correspondiente. No obstante, esta solución podría haber resultado perfectamente inútil, dado que estamos utilizando textos de dominios diferentes para el entrenamiento y la aplicación del modelo, por lo que los resultados obtenidos durante la evaluación del modelo no tienen por qué trasladarse al uso efectivo del clasificador. En este sentido, queda pendiente la evaluación de este enfoque desde un punto de vista cuantitativo, por ejemplo, aplicando el clasificador entrenado sobre

un *dataset* de *tweets* que nos permita determinar la eficacia del mismo en el entorno en que se aplicará. Desde un punto de vista cualitativo, los resultados han sido, como mínimo, aptos para nuestros propósitos, pues una gran parte de los *trending topics* declarados como relevantes lo eran en realidad, reduciendo en gran medida el esfuerzo de la anotación manual de los *trending topics* de acuerdo a la tendencia política a la que aluden.

El flujo de trabajo definido para el procesamiento de datos de Twitter se ha comprobado eficaz para los propósitos planteados. La política de almacenamiento de datos que hemos definido nos ha permitido disponer de todos los resultados intermedios generados por la ejecución del sistema, de manera que podíamos ejecutar componentes aislados en cualquier momento y sin necesidad de ejecutar todas las operaciones anteriores, lo que ha facilitado el desarrollo, el ajuste y la evaluación de cada uno de los módulos del proyecto al eliminar dependencias en la ejecución. Además, ha permitido la trazabilidad de las operaciones, de manera que la localización de los errores pudiera ser descubierta sin más que revisar los productos intermedios almacenados.

Sin embargo, hay dos elementos que probablemente tengan un margen de mejora considerable: el clasificador utilizado para clasificar los *tweets* y así caracterizar los *trending topics*, y el tratamiento que se hace del texto de acuerdo a las operaciones habituales en el Procesamiento de Lenguaje Natural. Cubriremos ambos en la sección de Trabajo futuro.

El componente de predicción ya ha sido discutido en el capítulo anterior, por lo que no abundaremos en él. Sí es importante señalar que, dentro de las líneas de trabajo futuro, posiblemente debería tener la prioridad más baja. La limitación que todo el procesamiento anterior puede imponer al mecanismo de predicción es tan grande, que lo más recomendable sería asegurar primero un buen resultado en cada una de las fases que preceden a la predicción, para poder abordar esta cuestión con garantías de éxito. En particular, el poder contar con una mayor riqueza de datos supondrá, con toda seguridad, un enorme impulso para la efectividad de este componente, ya que, como se ha explicado en la memoria, en múltiples ocasiones carecíamos de información suficiente, o directamente de ninguna información, para tener en cuenta todos los factores que afectan a la calidad de las predicciones. Por ejemplo, la ausencia en nuestros conjuntos de datos sobre Ciudadanos o la escasez de registros sobre Unidas Podemos. Habiendo seguido la actualidad, consideramos imposible que no se haya hablado de ellos tras determinados eventos ocurridos durante el mes de abril, por lo que lo más probable es que esa información haya sido desechada en alguna de las fases del proceso.

Es en este punto en el que se demostraron las carencias del proceso

de captura. Al centrarnos únicamente en los *trending topics*, nos estamos enfocando en parcelas de información muy concretas, que abarcan fenómenos muy específicos y localizados. De esta manera, eventos más difusos o de mayor amplitud pueden pasar desapercibidos. Por ejemplo, la crisis interna que atraviesa Ciudadanos: son tantos los nombres implicados y los enfoques que se le puede dar, que es difícil que uno de ellos cobre fuerza suficiente como para llegar a ser información de tendencias. Especialmente cuando deben competir con tendencias tan robustas como la información deportiva o ciertos programas de televisión.

Finalmente, se proporcionó el acceso a los datos relativos a las predicciones a través de una herramienta de visualización interactiva implementada como cuaderno de Jupyter, una herramienta muy utilizada en entornos de *Data Science* y *Business Intelligence* para la elaboración de aplicaciones de este tipo. La visualización implementada, aunque muy sencilla, trata de asegurar los principios básicos de claridad, utilidad, y rapidez. Pendiente queda la elaboración de una visualización más elaborada, que compensará con creces el esfuerzo dedicado a mejorarla en el momento en que se consiga mejorar los resultados de predicción.

Como resumen de estas conclusiones, consideramos el trabajo como culminado con éxito, habiendo cumplido los cuatro objetivos que nos marcamos al comienzo del proyecto, y satisfaciendo los requisitos académicos y personales que no se reflejan explícitamente en esta memoria. Así las cosas, volveremos en este punto sobre la pregunta de investigación que ha guiado el desarrollo de este proyecto:

> ¿Es posible predecir, dentro de un margen de error prudente, los resultados de un evento electoral atendiendo a la información más prominente disponible en redes sociales (por ejemplo, la incluida en las tendencias del momento), en un período de tiempo previo a dicho evento?

Sin poder responder a esta cuestión con un sí rotundo, consideramos que los resultados obtenidos son alentadores, en especial en el descubrimiento de tendencias a partir de los datos capturados. A pesar de la carencia de datos en algunos aspectos, la evolución de las distribuciones de voto que mostraban los datos se mostraban similares a las reales, representadas por la encuesta enlazada al comienzo del análisis. Esto nos hace pensar que la información que buscamos se encuentra efectivamente en los datos que podemos obtener de Twitter; es decir, que podemos explotar esa fuente para el caso de uso que hemos propuesto. Lo que sí es claro es que, previamente,

será necesario mejorar el proceso global tratando de optimizar cada uno de sus componentes.

8.2. Trabajo futuro.

Como se ha podido ver a lo largo de la memoria, y especialmente en las conclusiones, las oportunidades de mejora que ofrece el proyecto son muy diversas, pues prácticamente en cada fase del proyecto hemos aprendido lo suficiente como para ver nuevas maneras en las que abordar los retos que supone, y las soluciones que podemos proponer para resolverlos. En esta sección enunciaremos las más relevantes de cara a futuras iteraciones del sistema.

8.2.1. Mejora en los componentes del sistema.

El proceso de captura debe revisarse para solventar las carencias identificadas a lo largo del proyecto. Debe evaluarse la posibilidad de descartar cualquier tipo de filtro, de manera que capturemos toda la información proporcionada en tiempo real por Twitter. Con ello, ampliaríamos el alcance de la información con que contamos, pero también se incrementaría la dificultad para manejar un volumen de datos de esta magnitud, lo que nos obligaría a tomar decisiones más allá de la eliminación del filtro.

En primer lugar, sería necesario incrementar la potencia de cómputo y la capacidad de almacenamiento del soporte hardware utilizado, pues la escala de los datos excedería rápidamente las capacidades de los equipos utilizados en el marco de este proyecto. En segundo lugar, sería necesario una profunda revisión de las operaciones de procesamiento llevadas a cabo sobre los datos capturados, de cara a estudiar su viabilidad para el tratamiento de un gran volumen de datos. Finalmente, sería imprescindible buscar métodos alternativos de agrupación de *tweets*, con el fin de suplir la función que prestaban los *trending topics* en este proyecto: un mecanismo sencillo para separar los *tweets* semánticamente relevantes con un enfoque orientado a maximizar el *recall* de la operación.

Otro de los focos de interés de mejorar el desempeño de las fases de filtrado de información es la eliminación del componente manual. Para ello, debemos asegurar que los mecanismos de filtrado o selección de información son fiables, precisos y no desechan la información. Una vez asegurada una mayor eficacia por parte de los componentes de obtención y transformación de datos, será mucho más sencillo consolidar un proceso de cálculo de predicciones que pueda proporcionar conclusiones fiables y fácilmente interpretables. Como

hemos visto, la falta de datos introduce anomalías en el descubrimiento de tendencias, impidiendo la generación de nueva información aunque esta se trate de presentar de forma adecuada.

8.2.2. Consolidación de la arquitectura.

Por cuestiones de alcance, el presente proyecto estaba limitado a ser una prueba de concepto, un primer acercamiento a un flujo de trabajo que permitiera identificar puntos de falla graves en el planteamiento que hemos propuesto. Sin embargo, como ya indicamos en la introducción, el fin último sería implementar este mismo *workflow* en una plataforma más adecuada para el procesamiento de grandes volúmenes de datos. Por ejemplo, el salto a una infraestructura Hadoop, con todo lo que ello conlleva: computación y almacenamiento distribuidos y escalables, capacidad de paralelización de procesos para aumentar la velocidad de completado del flujo completo, uso de herramientas específicas y optimizadas para varias de las operaciones que hemos implementado dentro de este proyecto, como Apache Flume para la captura de datos de la API de Streaming, Apache Kafka para poder trabajar con los datos en tiempo real, en lugar de en *batch*, o Apache Hive para llevar a cabo, de forma distribuida, la transformación y limpieza que aquí aplicamos de forma secuencial en local... Es decir, dar paso a un sistema capaz de trabajar con buen rendimiento en un entorno de Big Data real.

8.2.3. Otros incrementos.

Aunque el caso de uso elegido ha sido la predicción de resultados electorales, la variedad de contenidos de Twitter y su escala podrían permitir obtener suficientes datos sobre cualquier tema como para reunir una muestra significativa que someter a un estudio de cualquier tipo. Si en las futuras ampliaciones del sistema se potencia su diseño modular, podría alcanzarse una solución aplicable en múltiples dominios, simplemente cambiando las piezas adecuadas del flujo de trabajo. Este ha sido uno de los principales intereses durante la elaboración de este trabajo, estimando que podría aplicarse a casos de uso diferentes modificando los componentes de clasificación de *trending topics*, o definiendo unos criterios distintos para la clasificación manual.

Finalmente, nos gustaría hacer mención de un planteamiento que se propuso al comienzo de la vida del proyecto, pero que fue rápidamente descartado para simplificar el flujo de trabajo. Aunque su interés desaparecería en caso de capturar el *stream* de datos íntegro, sí sería útil si se implementara una mejora del sistema actual, basado en *trending topics*, o se introdujera

algún tipo de filtro adaptable basado en palabras clave relevantes obtenidas por otros medios. Según el planteamiento actual, comenzamos a capturar información sobre una tendencia cuando esta aparece, y dejamos de hacerlo cuando deja de estar en la lista de tendencias. Es decir, estamos perdiendo de forma sistemática el conjunto de mensajes que provocaron la generación de la tendencia, así como los mensajes que permitirían caracterizar la fase de debilitamiento de la tendencia (después de que desaparezca de la lista, posiblemente siga generándose cierta información residual que podría ser de interés). Por ello, en un primer momento se planteó combinar la API de búsqueda de Twitter (*Search* API) con la API de *Streaming*, de forma que pudiéramos capturar mediante la primera, de forma asíncrona, la información que actualmente perdemos, y seguir utilizando la segunda para obtener . Esto nos permitiría comprender mejor la manera en que las tendencias "nacen y mueren", lo que seguramente ayudaría a la hora de, por ejemplo, definir nuestro propio mecanismo de detección de tendencias, con criterios diseñados de acuerdo a nuestro intereses, independientemente de las listas de *trending topics* generadas por Twitter.

Parte IV

Apéndices.

Apéndice A

Herramientas utilizadas.

En esta sección se recogen las principales herramientas software o librerías utilizadas para la implementación de este proyecto, estando listadas las versiones utilizadas en la Tabla A.1. Todas las librerías indicadas deberán estar instaladas en la distribución Python para que los diferentes *scripts* se ejecuten de forma correcta. Asimismo, el sistema es compatible para la versión 3.5 y posteriores de Python, pudiendo no funcionar correctamente en versiones anteriores de la rama 3.X.

Herramienta	Versión instalada
Anaconda	3
BeautifulSoup4	4.8.0
Chrome Web Driver	76
Google Cloud Language	1.3.0
Jupyter Notebooks	6.0.1
Matplotlib	3.11
Pandas	0.24.2
Python	3.5.3/3.7.3
Requests	2.22.0
Scikit-learn	0.21.3
Selenium	3.141.0
Spacy	2.1.8
Tweepy	3.7.0

Tabla A.1: Herramientas y librerías utilizadas

- *BeautifulSoup*. BeautifulSoup[1] es una librería para la interpretación de código HTML y XML. Además de la ingestión, permite una navegación rápida por los elementos que conforman la estructura de los documentos HTML. En el ámbito de este proyecto se utilizará para el procesamiento de las páginas web estáticas obtenidas mediante la librería `requests` y el código HTML de páginas web dinámicas tras generarlo mediante Selenium.

  Existe una librería alternativa, tradicionalmente más potente y rápida, llamada `lxml`. No obstante, es una opción menos flexible, que requiere de código bien formado para su correcto funcionamiento. BeautifulSoup, además, incluyó durante su desarrollo el *parser* propio de lxml, por lo que la diferencia de rendimiento entre ambas no es significativa.

- *Chrome Web Driver*. Esta herramienta incluye la funcionalidad básica de un navegador para su uso en entornos de automatización de pruebas sobre páginas web. Permite la carga, interacción y transformación dinámica de páginas web mediante una gran variedad de acciones. El *driver* utilizado se corresponde con Chromium[2] (motor de código abierto en el que se basa Google Chrome), aunque existen alternativas como el *driver* de Mozilla Firefox (*geckodriver*).

- *Google Cloud Language*. Es la librería cliente suministrada por Google para la interacción con la API de Google Cloud, facilitando tanto el uso como el acceso a través de las credenciales requeridas. Está disponible en múltiples lenguajes, entre ellos Python[3].

- *Jupyter Notebooks*. Jupyter Notebook es una aplicación web de código abierto de gran utilidad en cualquier proceso de trabajo con datos: permite ejecutar código en múltiples lenguajes (Python, Julia, R), realizar visualizaciones, documentar procesos mediante texto enriquecido (Markdown, LaTeX), compartir documentos, etcétera. La versión utilizada es la incluida en la distribución Anaconda.

- *Matplotlib*. Es una librería de generación de gráficos con una alta integración con la librería científica Numpy, además de ser compatible con Jupyter para la realización de visualizaciones tanto estáticas como interactivas.

- *Pandas*. Pandas es una librería de procesamiento de datos que proporciona estructuras de datos y herramientas de análisis de datos con un alto rendimiento (ya que está basado en la librería `numpy`). Incluye

[1] `https://www.crummy.com/software/BeautifulSoup/`
[2] `https://chromedriver.chromium.org`
[3] `https://pypi.org/project/google-cloud-language/`

funcionalidades útiles en el ámbito del proyecto como la integración de múltiples formatos de ficheros de datos (CSV, JSON); diferentes operaciones de agregación sobre los datos; y operaciones de transformación masivas sobre conjuntos enteros de datos.

- *Requests.* `requests`[4] es una pequeña librería de Python cuya principal funcionalidad es realizar peticiones HTTP, en particular peticiones GET para obtener el código HTML de un sitio web.
- *Scikit-learn.* Scikit-learn[5] es una librería en Python que implementa una gran variedad de recursos para el desarrollo y evaluación de modelos de aprendizaje automático. Abarca tanto aprendizaje supervisado (clasificación, regresión, etcétera) como no supervisado (*clustering*...), además de selección automatizada de modelos, representación y tratamiento de datos, transformación de dimensionalidad...
- *Selenium.* `selenium`[6] es una librería que integra el conocido *web-driver* Selenium[7] para la interacción con sitios web en tiempo real y la automatización de acciones en entornos web. Esta librería permite cargar páginas web completas e interactuar con ellas mediante eventos (*clicks*, *scrolls*, entradas de texto) o ejecutando código Javascript de forma dinámica. Su funcionamiento se basa en simular el comportamiento normal de un navegador web (generalmente suministrado a través de un complemento o *driver*) a través de estas acciones.

  Dentro del proyecto, se utilizará para interactuar con las páginas web dinámicas hasta alcanzar el estado de las mismas que se desea capturar, extrayendo entonces su código HTML y procesarlo mediante otras herramientas.
- *Spacy.* Spacy[8] es una librería *open-source* que implementa diversas funcionalidades necesarias para el procesamiento de lenguaje natural en múltiples idiomas. Se ha utilizado en contraposición a la clásica librería NLTK por la claridad de su documentación y su orientación a un entorno de producción.
- *Tweepy.* Tweepy[9] es una herramienta que integra la funcionalidad de diferentes APIs REST de Twitter para su utilización en entornos Python. Además de la funcionalidad análoga a la de estas APIs, ofrece

[4] `https://2.python-requests.org/en/master/`
[5] `https://scikit-learn.org/stable/`
[6] `https://selenium-python.readthedocs.io`
[7] `https://www.seleniumhq.org`
[8] `https://spacy.io`
[9] `https://www.tweepy.org`

distintas facilidades para la gestión de credenciales y conexiones, como la reconexión en caso de caída o el control automático de tasas de tráfico de datos (gestión de errores por exceso de volumen, etcétera).

Apéndice B

Atributos de objetos en Twitter.

Campo	Descripción
id	Identificador único del tweet (64 bits signed integer)
id_str	Equivalente en string al anterior
created_at	Tiempo UTC de creación del tweet
timestamp_ms	Exclusivo de streaming. Momento de captura del tweet en milisegundos
text	Texto del tweet en UTF-8 (posiblemente truncado)
truncated	Indica si el texto del tweet está truncado (ver Extended tweets)
user	Objeto JSON que describe al usuario autor del tweet
coordinates	Objeto GEO-JSON con las coordenadas de geolocalización
place	Objeto JSON correspondiente a una localización a la que se ha asociado el tweet (ver Tabla B.7)
in_reply_to_status_id	ID del tweet al que responde
in_reply_to_status_id_str	Equivalente en string al anterior
in_reply_to_screen_name	Nombre del usuario autor del tweet al que responde
in_reply_to_user_id	ID del usuario autor del tweet al que responde (no está necesariamente mencionado en el texto)
in_reply_to_user_id_str	Equivalente en string al anterior
is_quote_status	Indica si el tweet es un quote tweet
quoted_status_id	ID del tweet que se ha citado
quoted_status_id_str	Equivalente en string al anterior
quoted_status	Objeto JSON que describe el tweet que se ha citado
retweeted_status	Objeto JSON que describe el tweet que se ha retweeteado
retweet_count	Número de veces que se ha retweeteado un tweet
favorite_count	Número de veces que se ha marcado como favorito un tweet
reply_count	Número de veces que se ha respondido a un tweet
entities	Array de objetos JSON con las entidades asociadas al tweet

Tabla B.1: Estructura de un *tweet* (I).

Campo	Descripción
extended_entities	Array de objetos JSON con información adicional
lang	Idioma automáticamente detectado para el tweet
possibly_sensitive	Indica si tiene contenido potencialmente sensible
source	Servicio de origen del tweet (aplicación web, app Android...)
favorited	Indica si un tweet ha sido marcado como favorito por el usuario autenticado
retweeted	Indica si un tweet ha sido retweeteado por el usuario autenticado
contributors	Array de usuarios que han contribuido a la autoría de un tweet
quote_count	Indica el número de veces que el tweet ha sido citado (solo Premium/Enterprise)
filter_level	Indica el máximo nivel de filtrado del stream por el que un tweet puede ser capturado
matching_rules	Refleja las reglas de filtrado utilizadas para capturar el tweet (API de búsqueda)
geo	*Obsoleto*

Tabla B.2: Estructura de un *tweet* (II).

Campo	Descripción
id	Identificador único del usuario (64 bits signed integer)
id_str	Equivalente en string al anterior
name	Nombre del usuario (hasta 20 caracteres)
screen_name	Nombre identificativo del usuario (hasta 15 car.)
url	URL proporcionada por el usuario en su perfil
description	Texto descriptivo del usuario en el perfil
location	Cadena con la localización establecida por el usuario, no estandarizada
derived	Array de metadatos de enriquecimiento para la geolocalización del usuario
verified	Indica si se trata de una cuenta verificada
protected	Indica si el usuario tiene el perfil cerrado
followers_count	Número de seguidores
friends_count	Número de usuarios a los que sigue
listed_count	Número de listas de las que es miembro
favourites_count	Número de tweets que ha marcado como favoritos
statuses_count	Número de tweets que ha publicado
created_at	Fecha UTC de creación de la cuenta
geo_enabled	Indica si el usuario tiene activada la información de geolocalización
lang	Lenguaje configurado de la interfaz del usuario
default_profile	Indica si el aspecto del perfil de usuario es por defecto
default_profile_image	Indica si la imagen del perfil de usuario es por defecto

Tabla B.3: Estructura del usuario autor del *tweet*.

Campo	Descripción
contributors_enabled	Obsoleto
utc_offset	Siempre nulo
time_zone	Siempre nulo
default_profile	Indica si el aspecto del perfil de usuario es por defecto
default_profile_image	Indica si la imagen del perfil de usuario es por defecto
profile_image_url	Estético
profile_background_color	Estético
profile_background_image_url	Estético
profile_background_image_url_https	Estético
profile_background_tile	Estético
profile_banner_url	Estético
profile_image_url_https	Estético
profile_link_color	Estético
profile_sidebar_border_color	Estético
profile_sidebar_fill_color	Estético
profile_use_background_image	Estético
profile_text_color	Estético
withheld_in_countries	Lista de países en que un contenido está restringido
withheld_scope	Indica la naturaleza del contenido restringido (en este caso, un usuario)
follow_request_sent	Indica si el usuario autenticado ha mandado una solicitud de seguimiento al usuario
following	Obsoleto
is_translator	Obsoleto
translator_type	Obsoleto
notifications	Obsoleto

Tabla B.4: Campos descartables del *user*.

Campo	Descripción
user_mentions	Array de objetos *mention*
hashtags	Array de objetos *hashtag*
urls	Array de objetos *url*
symbols	Array de objetos *symbol*
media	Array de objetos *media*
media_size	Array de objetos *media_size*
polls	Objeto *poll*

Tabla B.5: Entidades/entidades extendidas de un *tweet*.

Campo	Descripción
id	Identificador único del usuario mencionado
id_str	Equivalente al anterior, como cadena de caracteres
indices	Posición de la mención en el texto
name	Nombre del usuario mencionado
screen_name	Nombre identificativo del usuario mencionado

Tabla B.6: Estructura de las menciones a usuarios.

Campo	Descripción
full_name	Nombre completo del lugar
id	Identificador alfanumérico del lugar
place_type	Tipo de lugar (ciudad, país...)
country_code	Código del país donde se ubica
country	Nombre del país donde se ubica
url	URL de la información de metadatos del lugar
name	Nombre corto del lugar
bounding_box	Contiene un objeto *Coordinates* (pares long/lat) y un tipo

Tabla B.7: Estructura de *places*.

Apéndice C

Trending Topics seleccionados al final del proceso.

A continuación se muestra la lista completa de *trending topics* resultantes del proceso de filtrado basado en su clasificación automática como 'política' o 'no política'. Se corresponden con los contenidos del fichero 'lista_tts_-export.txt', ubicado en el directorio *scriptsSeleccion*. Como puede apreciarse, el número total de registros es notablemente menor que el inicial, de cerca de 150.000, y los seleccionados muestran en general un buen nivel de relevancia. En resumen, podemos considerarlo un buen punto de partida que podrá ser refinado en futuras iteraciones del sistema mediante la optimización de cada una de las fases del proceso y el ajuste del flujo de trabajo por medio de los parámetros de configuración propuestos.

Una vez seleccionados de forma manual entre la lista anterior, la distribución de *trending topics* por partidos queda como se lista en las Tabla C.5.

	Trending topic		*Trending topic*
1	#AZEmovistarF1	46	#cloacasARV
2	#AryaStark	47	#jaimelannister
3	#AtletiRealValladolid	48	#malbalá
4	#BranStark	49	#mastercisarv
5	#CloacasARV	50	#okupadeluxe
6	#DecretazOtegi	51	#podemoslimpiarlascloacas
7	#ElDebateDeVerdad	52	26-M
8	#ElecccionesGenerales28A	53	Abascal
9	#Elecciones2019	54	Abdelaziz Bouteflika
10	#EleccionesA3N	55	Abdelaziz Buteflika
11	#EleccionesGenerales28A	56	Abel Ruiz
12	#EleccionesL6	57	Abrams
13	#EleccionesRTVE	58	Afrojack
14	#FalconViajes	59	Al Rojo Vivo
15	#FelizDomingo	60	Alcalá con Santiago Abascal
16	#FelizFinde	61	Amazon Prime Video
17	#FelizMartes	62	Anoeta
18	#FelizMiércoles	63	Antonio Ferreras
19	#Ferreras	64	April Fool's Day
20	#GHDÚOFinal1	65	April's Fools
21	#JaimeLannister	66	Argelia
22	#JonSnow	67	Arona
23	#JornadaDeReflexion	68	Arrate
24	#L6Neldebate	69	Arrimadas
25	#L6Nencampaña	70	Atresmedia
26	#LibertadEnCataluña	71	Audi de Touriño
27	#MasterCISARV	72	Autónoma de Barcelona
28	#MiCasaElecciones	73	Benito Villamarín
29	#MásPPMenosImpuestos	74	Bertín
30	#NBASundays	75	Bildu
31	#NuestroMapaDelAlma	76	Bildu 5
32	#NuestroMapadelAlma	77	Borja Valle
33	#OkupaDeluxe	78	Burbuja
34	#PodemosLimpiarLasCloacas	79	Bustos
35	#RealSociedadGetafe	80	Buteflika
36	#RealSociedadVillarreal	81	CEDADE
37	#SV2019Gala1	82	CEOE
38	#SalvadosPapa	83	Caballo
39	#TeamPobre	84	Campana
40	#TodoEsMentira66	85	Canaletas
41	#TyrionLannister	86	Carlos Castro
42	#Ultra2019	87	Casquero
43	#VamosRZ	88	Cayetana
44	#VotaPSOE	89	Cazorla
45	#WeColorLaLiga	90	Celta 3-1 Real Sociedad

Tabla C.1: Caption

	Trending topic		*Trending topic*
91	Chani	136	Feliz Domingo de Ramos
92	Cifu	137	Femen
93	Ciudadanos 42-51	138	Ferran López
94	Ciudadanos y Vox	139	Ferraz
95	Claveles	140	Ferreras
96	Clippers	141	Ferrán López
97	Coalición	142	Festmaster
98	Cobos	143	Fiesta de la Democracia
99	Colegio Electoral	144	Finanzas
100	Colón	145	Finlandia
101	Colón de Barcelona	146	Fnatic
102	Compromís y Podemos	147	Force
103	Con 24	148	Forn
104	Constitucional	149	Fornals
105	Corbyn	150	Forta
106	Corrupción Institucional	151	Frente Popular
107	Covadonga	152	Galilea
108	Cs 55	153	Gaming
109	Cs y Vox	154	Gantz
110	Curva	155	Gernika
111	D'Hont	156	Gil Manzano
112	Decretos	157	Gipuzkoa
113	Dice Abascal	158	Gobierno de Pedro Sánchez
114	Dosrius	159	Gorbachov
115	DÍA DE PARTIDO	160	Gracias España
116	Díaz Ayuso	161	Gracias VOX
117	ERC 15	162	Gracias Vox
118	Edurne Uriarte	163	Grapo
119	El Banco de España	164	Génova
120	El CIS	165	Génova 13
121	El Consejo de Gobierno	166	HBO España
122	El Falcon de Sánchez	167	Hernández Hernández
123	El PNV	168	Hugo Carvajal
124	El Papa	169	ID A VOTAR
125	El Parlament	170	Iglesias a Ferreras
126	En Euskadi	171	Iglesias y Montero
127	Errenteria	172	Inda
128	España e Italia	173	Industria 4.0
129	Espinosa	174	Inés Arrimadas
130	Esquerra	175	Irene
131	Este CIS	176	Irene Montero
132	Euskadi	177	Iván Espinosa
133	FA Cup	178	Iván Rodríguez
134	Fairy	179	Jn 7
135	Feijoo	180	Joffrey

Tabla C.2: Caption

	Trending topic		*Trending topic*
181	Jordi	226	Netanyahu
182	Jordi Pina	227	Nuevas Generaciones del PP
183	José Couso	228	Nuremberg
184	José Luis Martí	229	Nyon
185	José Manuel Soto	230	Ortega Lara
186	José María Aznar	231	Ortega Smith
187	José Ramón Bauzá	232	Osborne
188	Juan Pablo Lázaro	233	PACMA
189	Julen Arzuaga	234	PP 65
190	Jusapol	235	PP 70
191	Juve	236	PP y Podemos
192	JxCat	237	PSOE 116-121
193	Kiko Méndez Monasterio	238	PSOE 128
194	La Caperucita Roja	239	PSOE 132
195	La Romareda	240	PSOE y Ciudadanos
196	Laffer	241	PSOE y Cs
197	Lara	242	PSOE y Unidas Podemos
198	Leopoldo López	243	PSOE-Cs
199	Ley de Amnistía de 1977	244	PSPV
200	Librilla	245	Pablo FraCasado
201	Lisa	246	Pablo Hasel
202	Lo del PP	247	Pablo Iglesias e Irene Montero
203	Loquillo	248	Pacma
204	Los Warriors	249	Papa Francisco
205	Los de VOX	250	Parlamento Vasco
206	Luisle	251	Pasadena
207	Lunes Santo	252	País Vasco
208	M.Rajoy	253	Pedraza
209	Make It Right	254	Pedro Jota
210	Manresa	255	Pina
211	Manuel Moix	256	Pinchazo del PP
212	Mariano	257	Plaza de Colón
213	Marlaska	258	Podemos al Gobierno
214	Maroto	259	Por España
215	Marquesa	260	Prieto Iglesias
216	Mañana 10	261	REVENTADOS
217	Melilla	262	Rafael Sánchez Ferlosio
218	Mistol	263	Rajoy 3
219	Montero	264	Real Madrid TV
220	Monteros	265	Redondo
221	Montserrat	266	Rentería
222	Muñiz	267	Revilla
223	NOS VAMOS A VOTAR	268	Revolución
224	Natxo	269	Rey Mysterio
225	Navarra y País Vasco	270	Roque Mesa

Tabla C.3: Caption

	Trending topic
271	Rosell
272	Rubén Blanco
273	SMI a 850
274	Santiago Bernabéu
275	Segunda y Primera
276	Sevilla 5-0 Rayo Vallecano
277	Si PSOE
278	Si Podemos
279	Sondeo GAD3
280	Sondeo de GAD3
281	Soraya
282	Soraya Rodríguez
283	Soraya Sáenz de Santamaría
284	Sucesiones
285	Susana Díaz
286	TV3 y Catalunya Ràdio
287	Taula
288	Tezanos
289	Universidad de Barcelona
290	Uriarte
291	VOTAD
292	VOX - Luc Loren Toma La Calle
293	Vaticano
294	Verdaderos Finlandeses
295	Viaja Sánchez
296	Villarroya
297	Viva Vox
298	Volkswagen
299	Votad
300	Vox 21
301	Vox 23
302	Vox 24
303	Vox 36-38
304	Vox Iván Espinosa
305	WhatsApp
306	YA HE VOTADO
307	Yuumi
308	Zoido
309	abascal
310	arrimadas
311	atresmedia
312	bildu
313	borja valle
314	cayetana
315	cobos
316	covadonga
317	errenteria
318	este cis
319	iván espinosa
320	jn 7
321	joffrey
322	jusapol
323	marquesa
324	nuremberg
325	pacma
326	papa francisco
327	pasadena
328	por españa
329	psoe y cs
330	rajoy 3
331	tezanos
332	vox iván espinosa
333	Álvarez de Toledo
334	Ángel Hernández

Tabla C.4: Caption

Partido	
PP	#DecretazOtegi, #FalconViajes, #MásPPMenosImpuestos, Cifu, Coalición, Colón, Ciudadanos y Vox, Cs y Vox, Díaz Ayuso, El Falcon de Sánchez, Feijoo, Génova, Génova 13, José María Aznar, José Ramón Bauzá, Lo del PP, M.Rajoy, Mariano, Maroto, Nuevas Generaciones del PP, Ortega Lara, Plaza de Colón, Soraya, Soraya Sáenz de Santamaría, Viaja Sánchez, cayetana, Álvarez de Toledo
PSOE	#VotaPSOE, El Consejo de Gobierno, Este CIS, Ferraz, Gobierno de Pedro Sánchez, Marlaska, Monteros, Pinchazo del PP, Si PSOE, Soraya Rodríguez, Susana Díaz, Tezanos, este cis, tezanos, psoe y cs
Ciudadanos	#FalconViajes, Arrimadas, Ciudadanos y Vox, Coalición, Colón, Cs y Vox, El Falcon de Sánchez, Inés Arrimadas, Ortega Lara, Plaza de Colón, Viaja Sánchez, psoe y cs, arrimadas
Unidas Podemos	#CloacasARV, #PodemosLimpiarLasCloacas, #cloacasARV, #podemoslimpiarlascloacas, Compromís y Podemos, Corrupción Institucional, Iglesias a Ferreras, Iglesias y Montero, Irene, Irene Montero, Montero, Pablo Iglesias e Irene Montero, Pinchazo del PP, Podemos al Gobierno, Si Podemos
Vox	#FalconViajes, Abascal, Alcalá con Santiago Abascal, Ciudadanos y Vox, Colón, Covadonga, Cs y Vox, Dice Abascal, El Falcon de Sánchez, Gracias España, Gracias VOX, Gracias Vox, Iván Espinosa, Los de VOX, Ortega Smith, Plaza de Colón, Por España, Viaja Sánchez, Viva Vox, Vox Iván Espinosa, abascal, vox iván espinosa, covadonga, por españa
Otros	#LibertadEnCataluña, Bildu, Compromís y Podemos, El PNV, El Parlament, En Euskadi, Errenteria, Esquerra, Forn, JxCat, Navarra y País Vasco, PACMA, Pacma, Parlamento Vasco, Revilla, TV3 y Catalunya Ràdio, bildu, pacma

Tabla C.5: Distribución manual de *trending topics*.

Bibliografía

[1] *Estudio muestra por qué la gente miente en las encuestas.* http://mgcuchile.cl/estudio-muestra-por-que-la-gente-miente-en-las-encuestas. (visitado 23-05-2019).

[2] *El 26-J evidencia el fracaso de las encuestas electorales.* https://www.elconfidencial.com/elecciones-generales/2016-06-27/por-que-fallan-las-encuestas-electorales-en-espana_1223964. (visitado 23-05-2019).

[3] *¿Por qué mentimos en las encuestas de sexo, alcohol o alimentación?*, https://www.bbc.com/mundo/noticias/2013/03/130228_salud_encuestas_mentira_gtg. (visitado 23-05-2019).

[4] *Todos mentimos, a todas horas y a todo el mundo... menos a Google.* https://www.bbc.com/mundo/noticias-47734853. (visitado 23-05-2019).

[5] *Existen cámaras de eco en los medios sociales. ¿qué hacemos con ello?*, https://universoabierto.org/2018/04/24/confirmado-existen-camaras-de-eco-en-los-medios-sociales-que-hacemos-con-ello/. (visitado 25-05-2019).

[6] *La cámara de eco, o cómo la red te muestra solo lo que quieres ver.* https://beersandpolitics.com/la-camara-de-eco-o-como-la-red-te-muestra-solo-lo-que-quieres-ver. (visitado 25-05-2019).

[7] *Tu red social probablemente sea una cámara de eco.* https://www.sophiadigital.es/tu-red-social-probablemente-camara-eco/. (visitado 25-05-2019).

[8] *Twitter Firehose vs. Twitter API*, https://brightplanet.com/2013/06/25/twitter-firehose-vs-twitter-api-whats-the-difference-and-why-should-you-care/. (visitado 27-05-2019).

[9] K. Schwaber y J. Sutherland, «The scrum guide», *Scrum Alliance*, vol. 21, pág. 19, 2011.

[10] *Estimación de Póquer.* `https://www.scrummanager.net/bok/index.php?title=Estimación_de_póquer`. (visitado 22-05-2019).

[11] A. De Mauro, M. Greco y M. Grimaldi, «What is big data? A consensual definition and a review of key research topics», en *AIP conference proceedings*, AIP, vol. 1644, 2015, págs. 97-104.

[12] *Las cinco uves del Big Data*, `https://www.bbva.com/es/las-cinco-uves-del-big-data/`. (visitado 22-05-2019).

[13] *Tweeting Made Easier.* `https://blog.twitter.com/official/en_us/topics/product/2017/tweetingmadeeasier.html`. (visitado 01-06-2019).

[14] *Twitter libraries.* `https://developer.twitter.com/en/docs/developer-utilities/twitter-libraries.html`. (visitado 01-06-2019).

[15] S. Negash y P. Gray, «Business intelligence», en *Handbook on decision support systems 2*, Springer, 2008, págs. 175-193.

[16] *Machine Learning: An introduction*, `https://towardsdatascience.com/machine-learning-an-introduction-23b84d51e6d0`. (visitado 22-05-2019).

[17] D. M. Blei, A. Y. Ng y M. I. Jordan, «Latent dirichlet allocation», *Journal of machine Learning research*, vol. 3, n.º Jan, págs. 993-1022, 2003.

[18] R. Mehrotra, S. Sanner, W. Buntine y L. Xie, «Improving lda topic models for microblogs via tweet pooling and automatic labeling», en *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, ACM, 2013, págs. 889-892.

[19] M. Hajjem y C. Latiri, «Combining IR and LDA topic modeling for filtering microblogs», *Procedia Computer Science*, vol. 112, págs. 761-770, 2017.

[20] S. Martin-Gutierrez, J. C. Losada y R. M. Benito, «Recurrent Patterns of User Behavior in Different Electoral Campaigns: A Twitter Analysis of the Spanish General Elections of 2015 and 2016», *Complexity*, vol. 2018, 2018.

[21] P. Barberá y G. Rivero, «Understanding the political representativeness of Twitter users», *Social Science Computer Review*, vol. 33, n.º 6, págs. 712-729, 2015.

[22] J. Borondo, A. Morales, J. C. Losada y R. M. Benito, «Characterizing and modeling an electoral campaign in the context of Twitter: 2011 Spanish Presidential election as a case study», *Chaos: an interdisciplinary journal of nonlinear science*, vol. 22, n.º 2, pág. 023 138, 2012.

[23] E. M. Said-Hung, R. C. Prati y A. Cancino-Borbón, «La orientación ideológica de los mensajes publicados en Twitter durante el 24M en España», *Palabra Clave*, vol. 20, n.º 1, págs. 213-238, 2017.

[24] U. Kursuncu, M. Gaur, U. Lokala, K. Thirunarayan, A. Sheth e I. B. Arpinar, «Predictive Analysis on Twitter: Techniques and Applications», en *Emerging Research Challenges and Opportunities in Computational Social Network Analysis and Mining*, Springer, 2019, págs. 67-104.

[25] K. Wiegers y J. Beatty, *Software requirements*. Pearson Education, 2013.

[26] *Análisis de sentimiento en Google Cloud API*, `https://cloud.google.com/natural-language/docs/basics#sentiment_analysis`. (visitado 05-07-2019).

[27] *Documentación de Tweepy.* `https://tweepy.readthedocs.io/en/latest/index.html`. (visitado 24-03-2019).

[28] R. Alghamdi y K. Alfalqi, «A survey of topic modeling in text mining», *Int. J. Adv. Comput. Sci. Appl.(IJACSA)*, vol. 6, n.º 1, 2015.

[29] M. Cumbreras, E. Martínez-Cámara, J. Villena-Román y J. Morera, «TASS 2015 - The evolution of the Spanish opinion mining systems», *Procesamiento de Lenguaje Natural*, vol. 56, págs. 33-40, mar. de 2016.

[30] T. Baviera Puig, D. Calvo y G. Llorca-Abad, «Dataset Twitter-Spanish 2015 General Election», 2019. dirección: `%5Csmall%7Bhttp://hdl.handle.net/10251/117194%7D`.

[31] *¿Es legal el web scraping? : De webscraping y legalidad.* `https://diariodeuneletrado.wordpress.com/2017/03/22/es-legal-el-web-scrapping-de-webscrapping-y-legalidad/`. (visitado 25-06-2019).

[32] *Web Scraping: ¿legal o ilegal?*, `https://ecija.com/web-scraping-legal-ilegal/`. (visitado 25-06-2019).

[33] *Que no, que Internet no es una fuente accesible al público.* http://www.gonzalezasturiano.com/que-no-que-internet-no-es-una-fuente-accesible-al-publico/. (visitado 25-06-2019).

[34] *El 'web scraping' y la protección de datos*, https://cincodias.elpais.com/cincodias/2019/06/17/legal/1560752178_395215.html. (visitado 25-06-2019).

[35] *Google News cierra en España.* https://www.elconfidencial.com/tecnologia/2014-12-11/google-cierra-google-news-en-espana_588228/. (visitado 25-06-2019).

[36] *TF-IDF (Wikipedia).* https://en.wikipedia.org/wiki/Tf\T1\textendashidf. (visitado 03-07-2019).

[37] *How and When to Use ROC Curves and Precision-Recall Curves for Classification in Python*, https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python/. (visitado 24-06-2019).

[38] *Advantages of AUC vs standard accuracy*, https://datascience.stackexchange.com/questions/806/advantages-of-auc-vs-standard-accuracy. (visitado 24-06-2019).

[39] *Why Random Forests can't predict trends and how to overcome this problem?*, https://medium.com/datadriveninvestor/why-wont-time-series-data-and-random-forests-work-very-well-together-3c9f7b271631. (visitado 11-07-2019).

[40] D. Zimbra, A. Abbasi, D. Zeng y H. Chen, «The state-of-the-art in Twitter sentiment analysis: A review and benchmark evaluation», *ACM Transactions on Management Information Systems (TMIS)*, vol. 9, n.º 2, pág. 5, 2018.

Los diagramas elaborados para este trabajo contienen iconos descargados gratuitamente de la plataforma www.flaticon.com, y han sido utilizados de acuerdo con los términos y condiciones de la licencia de distribución del sitio. La propiedad intelectual de cada uno de los iconos pertenece a sus respectivos autores.