Trabajo Fin de Grado en Ingeniería Informática

Infraestructura software para el análisis de las interacciones en Twitter: aplicación a la detección de trastornos de conducta alimentaria

Daniel Revillo Rey

Director: Pedro Javier Álvarez Pérez-Aradros

Codirector: Ricardo J. Rodríguez Fernández

Departamento de Informática e Ingeniería de Sistemas
Escuela de Ingeniería y Arquitectura
Universidad de Zaragoza

Diciembre 2020
Curso 2019/2020

Agradecimientos

A mi familia y gente cercana por apoyarme incondicionalmente.
A la Fundación APE por ofrecer esta oportunidad.
A Pedro y Ricardo por aguantarme incluso en verano.

RESUMEN

La red social Twitter permite publicar contenido variado, desde fotos y vídeos hasta texto plano con un límite de 280 caracteres. Este contenido puede ser compartido y respondido por otros usuarios en sus perfiles. Los 340 millones de usuarios activos mensuales subiendo publicaciones, compartiendo y respondiendo crean una red de interacciones inmensa que, gracias al acceso gratuito al público, puede ser explorada casi sin límite.

En este proyecto se ha creado un sistema software que permite la interacción con Twitter y sus diversas entidades, recopilando esta información para representar las distintas interacciones entre los usuarios de manera legible para el ser humano y así extraer conocimiento de las mismas.

Como caso de estudio, este sistema se ha enfocado a los Trastornos de Conducta Alimentaria (TCA). Los TCA son enfermedades graves que afectan al 8 % de la población adolescente entre los 13 y los 19 años, atacando en mayor medida a la población femenina. Los TCA son enfermedades muy desconocidas y de difícil tratamiento pese a ser comunes. Desafortunadamente, en la red hay multitud de contenidos nocivos que dan consejos peligrosos para los más jóvenes. Con el sistema desarrollado se pretende detectar la apología de estos trastornos en la red social Twitter y monitorizar las relaciones de los usuarios que las incentivan. Con una representación legible de estas relaciones se puede saber quién sube estos contenidos, quién participa en ellos, cuánta gente involucrada hay y qué palabras clave se emplean. El estudio de estas métricas ayudará a una mejor prevención y detección precoz de estos trastornos, localizando las fuentes de información que hacen apología de estas conductas y emprendiendo así las denuncias oportunas a los moderadores de la red social para que este tipo de contenido sea eliminado. Se pretende con este sistema desarrollado aportar un granito de arena a una lucha en la que la victoria está aún lejana.

El sistema desarrollado puede ser aplicado a otros casos de estudio, gracias a la generalización realizada. Por ejemplo, se pueden monitorizar los *hashtags* más utilizados durante un período de elecciones u observar las relaciones de usuarios de un lugar durante un conflicto, dándole al sistema un carácter más transversal y ampliando sus potenciales casos de uso.

Palabras clave: Twitter, TCA, Análisis de Sentimientos, Extracción de Conocimiento

ABSTRACT

Twitter is a social network that allows to publish posts whose content can vary from photos and videos to plain text with a limit of 280 characters. These posts can be shared and replied to by other users. The 340 million monthly active users uploading, sharing and replying other content create a vast network of interactions that, thanks to the free public access, can be explored almost without limit.

In this project a software system was created that allows to interact with Twitter and its various entities (*posts*, users, *hashtags*). The system also allows to collect and process the Twitter content in order to represent the different interactions between users in a human-readable way and thus to extract knowledge from them.

In particular, this system is focused on Eating Disorders (ED). ED are serious diseases that affect 8 % of the adolescent population between 13 and 19 years old in Spain, attacking mostly the female group. Despite being common, they are very unknown diseases and difficult to treat medically. Unfortunately there is plenty of harmful content in the net that gives dangerous advice to the youngest. The system developed aims to detect the advocacy of these disorders in Twitter and monitor the relationships between users who encourage them. With a legible representation of these relationships it is possible to know who is uploading these contents, who is participating in them, how many people are involved and many other metrics. The study of these metrics will improve prevention and early detection of these disorders, locating the sources of information that support these behaviors. In this way, it will be possible to report this type of content to the moderators of the social network to delete it, and thus contributing a little bit to a fight in which victory is still long in coming.

Finally, the proposed system has been abstracted to monitor not only these apologies, but any other issue of interest for the user. Monitoring the most used *hashtags* during an election period, watching the relationships between users during a conflict in any place and many other topics can be represented by this system, giving it a more transversal functionality and expanding its potential use cases.

Keywords: Twitter, Eating Disorder, Sentiment Analysis, Knowledge Extraction

Índice general

Índice de tablas

Índice de figuras

Capítulo 1

Introducción

La red social Twitter es una plataforma que permite publicar mensajes (denominados tuits) con diversos contenidos. Estos tuits pueden contener tanto textos limitados a un máximo de 280 caracteres como elementos multimedia (imágenes y vídeos). Los usuarios de esta red social pueden compartir con sus seguidores sus publicaciones. Un usuario puede tener un perfil abierto, permitiendo que le siga quien quiera o un perfil cerrado, decidiendo quién puede seguirle y quién no. Los seguidores de un perfil pueden compartir en su perfil los tuits de los usuarios que siguen y de los usuarios con perfil abierto. En terminología de Twitter, esto se denomina *retweet*. También se puede citar este tuit en uno que se vaya a publicar, lo que se denomina *quote*. Además, pueden compartir un comentario respecto a los mismos, llamándose esto *reply* o respuesta. Finalmente pueden indicar que un tuit les agrada, con una opción de "*Me gusta*". Twitter cuenta con más funcionalidades, como la creación de listas y los mensajes privados, no relevantes para el desarrollo de este trabajo.

Estas formas de interacción hacen que esta plataforma sea muy dinámica e instantánea. Cada vez que se actualiza la pantalla principal de un usuario (denominada *feed*) aparecen aquellos tuits que han publicado, hecho *retweet* o respondido las cuentas que el usuario sigue. Gracias a este dinamismo, esta red permite estar al día de manera instantánea de cualquier tema, noticia, evento, etc. Multitud de entidades, desde personajes públicos hasta grandes empresas y medios de comunicación masivos, emplean esta red para compartir sus publicaciones, viéndose incluso debates enteros en la misma. Todo esto hace de Twitter una de las redes sociales más empleadas a día de hoy, con alrededor de 340 millones de usuarios mensuales activos [14].

Una de las mayores diferencias entre Twitter y sus grandes competidoras como Instagram y Facebook es que se puede acceder a gran parte del contenido, ya que una vez se sube a Twitter pasa a ser público. Esto se ve reflejado en las funciones disponibles para desarrolladores que presenta Twitter, que permiten pedir tuits que citen alguna palabra, expresión clave o *hashtag* o la lista de seguidores de una cuenta, entre muchas otras. La facilidad del acceso a todo este contenido hace posible la realización de muchos experimentos e investigaciones de carácter social, ya que se pueden visualizar las tendencias de cada momento. La recopilación y estudio de todos estos contenidos permite deducir

ciertas pautas y conductas de comportamiento de los usuarios, de las cuales se puede extraer conocimiento valioso.

Estas redes sociales masivas también tienen sus contras, como se muestra en el documental recientemente estrenado "*The Social Dilemma*" [16, 5]. El 60 % de la población mundial ya tiene acceso a Internet y, con la reducción de costes que han permitido los avances tecnológicos, también tiene acceso a dispositivos que pueden conectarse a la red. Esto hace que el acceso a las redes sociales sea posible para buena parte de la población, cada día más joven. Un estudio de GlobalWebIndex indica que la media mundial de tiempo diario usando redes sociales es de 2 horas y 29 minutos [4]. Los avances en marketing y en la psicología del consumidor hacen que las redes sociales sean cada vez más una adicción y no una herramienta de ocio. Varias de las redes sociales más utilizadas emplean una interfaz gráfica cuya interacción predominante es el deslizamiento hacia abajo, lo que genera una adicción bajo el mismo principio que las máquinas tragaperras.

Destaca también que las redes sociales son gratuitas para los usuarios y una de las fuentes de financiación de las mismas es a través de publicidad. Los usuarios de estas redes están sometidos a un bombardeo de publicidad personalizada durante todo el tiempo que invierten en ellas. Entre otras, en esta publicidad se muestra el cánon de belleza actual como una meta a lograr. A esto se suma la constante necesidad de sentirse querido y aceptado que han generado iniciativas como el botón de "*Me gusta*". Cada vez que se sube una publicación o una foto se intenta conseguir el mayor número de "*Me gusta*" posible, lo que puede llegar a provocar ansiedad. Si las personas con más "*Me gusta*" tienen cierto aspecto y se comportan de cierta manera, se intenta reproducir este aspecto o comportamiento para sentirse querido en las redes. Y, en general, cuanto más cerca está el usuario del cánon de belleza, más "*Me gusta*" consiguen sus publicaciones. Esto genera consecuencias nefastas en parte de los usuarios, sobre todo en los más jóvenes, generalmente adolescentes.

Los casos de Trastornos de Conducta Alimentaria (TCA) entre los adolescentes están aumentando exponencialmente, siendo el mal uso de las redes sociales una de sus causas [24]. Estar sometido a los cánones de belleza actuales que fomentan la delgadez a través de las redes puede generar en los adolescentes una necesidad de adaptarse a ellos, lo que llega a generar obsesión y derivar en uno de estos trastornos. Los TCA son enfermedades graves que afectan al 8 % de los adolescentes entre 13 y 19 años, incidiendo de manera más fuerte en la población femenina [7]. Los más comunes son la anorexia nerviosa y la bulimia nerviosa. Estas enfermedades son aún muy desconocidas y su tratamiento es muy complejo y costoso, no siendo seguro el éxito. Debido a esta falta de conocimientos sobre cómo tratar estas enfermedades, una de las mejores maneras de luchar contra ellas es la mejora en la prevención y la detección temprana de las mismas.

En la red existen numerosos contenidos nocivos que hacen apología de este tipo de enfermedades, incluyendo multitud de blogs, cuentas y usuarios en redes sociales que hacen proselitismo de los TCA. A diferencia de otros países, en España todavía no es una conducta delictiva incentivar este tipo de enfermedades, aunque algún gobierno autonómico como Cataluña ha dado algunos pasos recientemente (en 2019) para sancionar a los que difundan este tipo de contenidos [15]. Las cuentas que difunden estos conteni-

dos tienden a considerarse adoradoras de *Ana y/o Mía*, aludiendo a anorexia y bulimia, respectivamente. Entre muchas otras prácticas se realizan las *carreras de kilos*, en las cuales se consigue mejor puntuación cuanto menos se come, más agua se bebe y más ejercicio se hace durante un intervalo de tiempo dado, con el fin de perder el máximo peso posible. Además, se divulgan consejos y prácticas nocivas con el mismo fin.

1.1. Objetivo y motivación

El objetivo de este proyecto es ayudar a mejorar la prevención y la detección precoz de los TCA, además de ampliar el conocimiento sobre los mismos, empleando como fuente la red social Twitter. Para ello, se ha creado un sistema software que integra funcionalidad para interactuar, procesar y analizar distintas entidades encontradas en Twitter para finalmente representarlas de manera legible para el ser humano. Se pretende localizar las cuentas de Twitter que hacen apología de los TCA de manera más activa y visualizar las cuentas que interactúan con ellas, para extraer información de estos grupos de contactos. Esta información permite reportar estas cuentas al equipo de moderación de la red para que adviertan o eliminen este tipo de contenido. Además, el hecho de visualizar las interacciones entre las cuentas permite saber quién las apoya, lo que puede ayudar también en la prevención de estos trastornos o, al menos, en la detección temprana en estos usuarios. El sistema también permite indagar en la psicología de personas que padecen TCA, ya que en Twitter se publican con frecuencia pensamientos y opiniones.

El proyecto se ha realizado en colaboración con la Fundación APE, una organización sin ánimo de lucro que lucha contra estas enfermedades contribuyendo a su prevención y erradicación [7]. La Fundación ayuda a informar acerca de estos trastornos para frenar el intenso crecimiento que experimentan hoy en día. Sus integrantes son voluntarios y voluntarias sensibilizados con los TCA y sus consecuencias, juntados para desarrollar estrategias de prevención y atenuación del sufrimiento de las personas afectadas y sus familias. En este proyecto han servido como usuarios/clientes del sistema desarrollado, detallando qué información sería interesante mostrar.

1.2. Estructura del documento

Esta sección describe la estructura del documento y una breve descripción del contenido de cada capítulo. La información sigue una estructura *Top-down*, comenzando por una visión global del sistema y pasando a describir los componentes.

El capítulo 1 es la introducción al proyecto, explicando tanto el contexto como los objetivos del mismo. El Capítulo 2 describe la fase de análisis de la realidad así como las tecnologías empleadas en la solución. En el Capítulo 3 se encuentra la descripción detallada sobre el diseño de la solución propuesta y cómo se ha desarrollado e implementado cada componente de la misma. Tras esto se explica el caso de estudio en el que se ha aplicado el sistema, el caso de los TCA, en el Capítulo 4. Se muestran así los resultados de esta aplicación y el conocimiento que se ha extraído. Finalmente el Capítulo 5 recoge

las conclusiones obtenidas durante el desarrollo del proyecto y la perspectiva futura y potenciales mejoras del mismo.

Capítulo 2

Análisis y herramientas

En este capítulo se analizan la problemática enfrentada, los requisitos que se han de cumplir y las decisiones sobre las herramientas empleadas para crear la solución desarrollada. La elección de las herramientas a emplear se ha hecho en base a la adecuación al proyecto, al uso previo y a la curva de aprendizaje que requieren.

2.1. Orientación del proyecto

Este sistema será empleado por aquellas personas cuyo interés sea monitorizar las relaciones entre usuarios de Twitter que tengan algún tema en común y participen de manera activa en él. Este público no es necesariamente conocedor de muchas tecnologías ni poseedor de habilidades que le permitan enfrentarse a una API[1] o a una interfaz gráfica compleja. El usuario puede variar, desde un profesor buscando información sobre usuarios que comparten su especialidad, un político buscando qué se dice sobre él o un científico contrastando opiniones acerca de cuál es la mejor vacuna para la COVID-19. En ningún caso tienen que ser usuarios con conocimientos amplios sobre interfaces, ni serán niños o adolescentes, ya que la extracción de conocimiento de las interacciones de los usuarios de Twitter es un tema carente de interés en esa franja de edad. Por tanto, se ha estimado un público objetivo de usuarios entre 18 y 70 años, sin tener necesariamente conocimientos amplios sobre informática pero con un interés en las redes sociales y las interacciones que se realizan en ellas. Dentro de este grupo se pueden encontrar personas que sí tengan conocimientos en informática y les pueda interesar este proyecto para explotarlo de otra manera, lo que ha influido en el diseño del formato final de presentación de la herramienta.

[1] Interfaz de Programación de Aplicaciones: "Conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar el software de las aplicaciones, permitiendo la comunicación entre dos aplicaciones de software a través de un conjunto de reglas." - Yúbal Fernández [20]

2.2. Requisitos

Los requisitos de este sistema, tanto funcionales como no funcionales, se han definido en la tabla 2.1.

Código	Descripción
F1	El sistema debe permitir la búsqueda de tuits según ciertos hashtags
F2	El sistema debe analizar los tuits buscados y guardar las interacciones (*retweet, reply, quote, mention*) entre los usuarios en una base de datos
F3	El sistema debe permitir mostrar mediante un grafo el contenido de la base de datos
F4	El grafo representará la cuantía de las interacciones entre los usuarios
F5	El grafo representará el tipo de relación entre los usuarios y entre los usuarios y los tuits
F6	El sistema debe presentar informes con información descriptiva sobre los datos contenidos en la base de datos
F7	El sistema debe permitir visualizar el estado actual de la base de datos
NF1	El sistema debe tener una base de datos
NF2	El sistema debe tener una interfaz gráfica
NF3	El sistema debe poderse integrar en aplicaciones externas

Tabla 2.1: Requisitos funcionales y no funcionales

Tras analizar los requisitos se puede apreciar que este software se puede ofrecer como una aplicación web que integra la herramienta de extracción de conocimiento con sus correspondientes funcionalidades como la gestión de usuarios y presentación de informes; pero también como un servicio a integrar en otras aplicaciones. Como primera aproximación se ha decidido hacer una aplicación web simple, sin cerrar el camino a su explotación *as a Service*.

2.3. Tecnologías empleadas

2.3.1. Twitter

Twitter es una de las redes sociales más empleadas a día de hoy, con 340 millones de usuarios activos mensuales [14]. Las posibilidades que ofrece son muy amplias, dado el formato de sus mensajes y el acceso abierto a muchos de ellos. Numerosos estudios han empleado esta red social como fuente de información [22, 23] haciendo hincapié en el análisis de sentimientos de los usuarios. Por ejemplo, en [22] fue utilizada para monitorizar los tuits políticos en España durante la emergencia de la COVID-19. La red Twitter también se empleó para analizar la opinión de las masas sobre el Brexit, empleando procesadores de lenguaje natural y observándose un decrecimiento del positivismo hacia el Brexit entre 2017 y 2019 [17]. El análisis de sentimientos de esta plataforma es una herramienta muy utilizada desde su aparición, pudiendo citar innumerables estudios. Acercándose un poco más al objeto de estudio de este proyecto, en [18] se analizaron las interacciones entre los usuarios participando en mensajes políticos. En este estudio se llega a la conclusión de que los *retweets* están altamente polarizados en el ámbito político, mientras que las menciones no, además de que los usuarios que aplican *hashtags* neutrales tienen más probabilidades de entablar comunicación con ideologías opuestas. El análisis de sentimientos y el de interacciones en Twitter se han empleado en numerosas ocasiones siempre con el fin de extraer información descriptiva sobre las masas. Esto hace de Twitter la red social ideal para el análisis de datos de este proyecto, ya que permitirá encontrar las comunidades que apoyan los TCA, ver cómo se relacionan y mostrarlo a los expertos para su posterior estudio.

La API REST de Twitter

La plataforma de desarrolladores de Twitter contiene productos y servicios que garantizan el acceso y, por tanto, la explotación de todos estos datos permitiendo sacarle el máximo partido a esta red social a través de una API. Esta API está actualmente en proceso de migración a la versión 2 [11]. El acceso a esta API tiene diferentes modalidades:

- **API estándar**: La velocidad de descarga de información sobre tuits está limitada, además de las peticiones disponibles en intervalos de 15 minutos. La búsqueda de tuits se remite a un máximo de 7 días atrás, por lo que no emplea el archivo completo. Permite la publicación de tuits, la gestión de listas, de mensajes directos y de la cuenta, además del filtrado de tuits en las peticiones. Se requiere una dirección de correo electrónico para conseguir una cuenta y no puede tener fines comerciales. Es el producto más limitado de todos los ofrecidos por Twitter, pero es gratuito. Esta API es la que se ha empleado en este proyecto.
- **API Premium:** Añade a las características de la API estándar los tuits de un máximo de 30 días atrás y permite buscar en el archivo completo, además de

incrementar las peticiones por hora y los tuits por petición en las búsquedas.

- **API Enterprise:** Se requiere una licencia empresarial para su uso, no tiene límites en la cantidad de tuits por búsqueda ni en las búsquedas por intervalo de tiempo, además de gozar de acceso completo a todo el archivo. Permite la visualización de métricas, funciones de búsqueda y contiene gestores de cuentas y soporte técnico personalizado.

- **API Ads:** Esta API permite gestionar los anuncios de cualquier compañía en la plataforma.

2.3.2. Python

Python es un lenguaje de programación potente y rápido, muy versátil y funcional en la mayoría de equipos. Muchas empresas de servicios de Internet emplean fragmentos de código en Python, como Google y su servicio Youtube. Tiene una sintaxis sencilla y la curva de aprendizaje no es demasiado exigente, además de contar con una extensa documentación y multitud de libros y publicaciones sobre sus distintas características. Otra ventaja es el hecho de ser un lenguaje de código abierto[2] [19].

La versatilidad de este lenguaje y las librerías y externas que permite integrar, como Twython o Neo4j, han sido las razones principales de su elección. Además, la curva de aprendizaje era adecuada para las restricciones temporales del proyecto.

2.3.3. Twython

Sirve de nexo de unión entre la API de Twitter y el software de Python desarrollado. Existen diversas herramientas que cumplen esta función, pero esta es una de las pocas que está mantenida actualmente [12]. Permite la conexión con la API mediante los dos tipos distintos de autenticación:

`OAuth 1` es para llamadas en las que el usuario ha de estar autenticado, como por ejemplo publicar un tuit, mandar un mensaje privado o seguir a otro usuario.

`OAuth 2` es para llamadas de solo lectura en las que no hace falta estar autenticado, como búsquedas o leer el hilo de tuits de un usuario público.

Gracias a esta herramienta se ha podido acceder a los datos de Twitter desde Python. Ha habido un pequeño hándicap con esta librería ya que las funciones que proporciona no devolvían el resultado esperado, por lo que no se ha conseguido integrar del todo. Aún así esta librería contiene una función que permite hacer peticiones directamente con la URL que proporciona la API de Twitter, con lo que se ha logrado el objetivo de manera indirecta.

[2]Software de código abierto: Todo software poseedor de una licencia compatible con la *Open Source Definition*. Más información en [21].

2.3.4. Neo4j

El problema que se quiere tratar en este proyecto es la representación legible para el ser humano de las redes de interacciones que se forman en Twitter. Para ello se tienen que almacenar unos datos que representen estas interacciones en una base de datos. En este caso de estudio, se ha empleado una base de datos de grafos.

Las bases de datos de grafos tienen características peculiares, ya que los únicos datos que hay en ellas son nodos y relaciones. Las bases de datos tradicionales ordenan los datos en filas y columnas, y estas en tablas, calculando las relaciones entre ellos en tiempo de ejecución; las bases de datos de grafos, en cambio, guardan la información con una estructura definida por relaciones entre datos, por lo tanto no la tienen que computar al ejecutar una consulta. Estos datos, denominados nodos, guardan información sobre la entidad deseada así como punteros dirigidos al resto de nodos con los que está conectado. Al ser una estructura simple pero optimizada, se pueden realizar consultas complejas con tiempos de ejecución cortos.

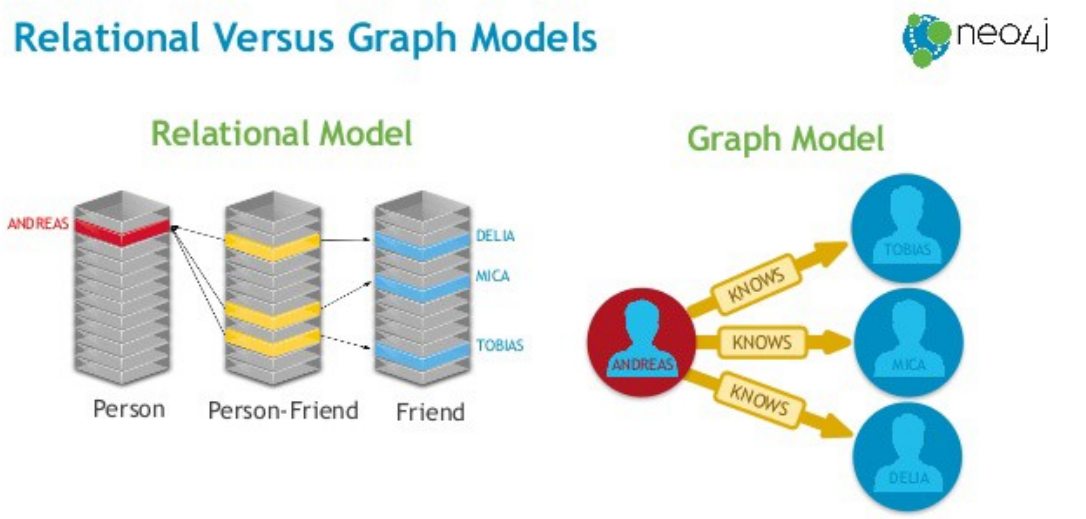

Figura 2.1: Diferencia entre el modelo relacional y el modelo de grafo (extraída de [13])

Otra ventaja de esta estructuración es la representación gráfica. En el caso de las bases de datos de grafos la representación gráfica aporta mucha información y permite a los usuarios extraer conocimiento de un simple vistazo. Los grafos de conocimiento se emplean para albergar y buscar entre enormes cantidades de datos estáticos. Por ejemplo, la NASA ha empleado esta base de datos para buscar qué problemas ha habido en anteriores misiones, cómo se han resuelto y qué han aprendido de ellos, ahorrándose una gran cantidad de tiempo y fondos en posteriores misiones [25].

Existen diversas bases de datos de grafos, lo que ha hecho de la elección un proceso

complejo. Actualmente las más empleadas son ArangoDB, Neo4j, OrientDB, Microsoft Azure CosmosDB, Virtuoso, Amazon Neptune y JanusGraph [2]. Todas ellas poseen características acordes a las necesidades del proyecto, modelo de datos de grafo y conexión a través de Python. Las primeras descartadas han sido aquellas sin licencia gratuita y aquellas que no poseen licencia de código abierto. Finalmente se ha elegido entre el resto según la facilidad de aprendizaje del lenguaje de consultas de cada una, además de su posibilidad de integrar herramientas de representación y por popularidad, para llegar a la conclusión de que Neo4j era la más apropiada [9]. Una tabla resumen mostrando las propiedades evaluadas para cada solución estudiada se muestra en la Tabla 2.2.

Base de datos	**Fuente**	**Modelo**	**Consultas**	**Driver Python**	**Otros**
Neo4j	Código Abierto	Grafo	Cypher	Sí	Bloom
OrientDB	Código Abierto	Multi modelo	Gremlin, OrientSQL	Sí	.....
ArangoDB	Código Abierto	Multi modelo	AQL	Sí	.....
TigerGraph	Código Cerrado	Multi modelo	GSQL	Sí	Versión gratuita limitada
AllegroGraph	Código Abierto	Multi modelo	SPARQL, Prolog	Sí	Gruff
JanusGraph	Código Abierto	Grafo	Gremlin	Sí	.....
Azure CosmosDB	Código Cerrado	Multi modelo	SQL	Sí	.....
Amazon Neptune	Código Cerrado	Multi modelo	Gremlin y SPARQL	Sí	Sin opción gratuita
Virtuoso	Código Abierto	Multi modelo	SPARQL	Sí	.....

Tabla 2.2: Bases de datos de grafos

La plataforma Neo4j no solo es una base de datos de grafos, sino que contiene también herramientas con diversas funciones, una de las cuales es Bloom. Bloom es una herramienta de representación de grafos de Neo4j que ofrece una solución cómoda, personalizable e integrable. Esta herramienta permite crear diferentes perspectivas per-

sonalizables y exportables, cada una representando el grafo (o partes de él) de distinta manera. Además, ofrece un motor de búsqueda para encontrar patrones en el grafo y permite crear y guardar frases de búsqueda, pudiendo ser reutilizadas en todo momento.

Existen muchas bases de datos de grafos, pero Neo4j, que incorpora la herramienta Bloom y ofrece drivers para Python y otros lenguajes, es la que se adecúa en mayor medida a las necesidades de este proyecto. Además, implementa un lenguaje de gestión y consultas propio, denominado Cypher. La edición *community* es de código abierto y permite el uso en proyectos pequeños de manera gratuita. Para el acceso a Bloom se necesita una clave de activación que ofrecen igualmente de manera gratuita.

2.3.5. Flask

Para la aplicación web se comenzó empleando el conjunto de subsistemas MEAN (MongoDB, Express, Angular y Node.js), una estructura de herramientas que permiten la creación de estas aplicaciones web, desde la lógica de la aplicación hasta la interfaz gráfica. El uso de MongoDB no era necesario, ya que se emplea Neo4j como base de datos. Conforme se avanzaba con esta solución se dedujo que había opciones más sencillas que cubrían todas las necesidades del proyecto. En el caso de este conjunto de subsistemas, realizar la parte de la lógica de la aplicación con Express y Node era tedioso, por lo que se buscaron alternativas y se encontró Flask.

Flask es un framework escrito en Python y optimizado para el mismo, muy sencillo y que permite el despliegue de un servidor y la gestión de rutas de manera intuitiva [6]. El uso de esta herramienta ha facilitado en gran medida la construcción de la parte de la lógica de la aplicación.

Capítulo 3

Diseño e implementación del sistema

En este capítulo se explican las decisiones de diseño e implementación del sistema. Se describe primero la arquitectura y sus componentes con la funcionalidad de cada uno, para terminar con su implementación y la integración de todos ellos en una aplicación web.

3.1. Arquitectura y componentes

El sistema desarrollado, bautizado como TwiBloom, va a ejecutarse en el contexto mostrado en la figura 3.1. Se puede observar que el sistema tiene un flujo claramente definido. El sistema está ejecutándose en un servidor y se puede acceder a él mediante una URL desde un navegador, paso (1). Desde el navegador, el usuario especifica los parámetros de la búsqueda de tuits. Acto seguido, usando las credenciales de una cuenta de Twitter con permisos de desarrollador, se realiza una petición de manera adecuada a la API de Twitter, paso (2). Los resultados obtenidos se almacenan en un fichero temporal con formato JSON, paso (3). Un ejemplo de este flujo de interacción con Twitter se muestra en la figura 3.2).

Adicionalmente, puede haber un paso opcional relativo a un clasificador binario. Este clasificador es un componente cuya función es etiquetar los tuits como potencialmente peligrosos o inofensivos. Para ello se hace un modelo de aprendizaje máquina que se entrena pasándole muchos textos ya etiquetados. Al ser entrenado, el modelo saca conclusiones y patrones sobre los textos que emplea como reglas para etiquetar textos futuros. Cuanto mayor es el conjunto de textos de entrenamiento, mejores serán los resultados del clasificador. En caso de que exista la fase del clasificador binario, se pasan los tuits por este, descartando aquellos que el clasificador no considere potencialmente peligrosos (pasos 4 y 5). En caso contrario los tuits pasan directamente al componente de persistencia, que puebla la base de datos Neo4j.

El componente de persistencia de TwiBloom procesa los tuits y obtiene todos los participantes de los mismos, así como las relaciones entre ellos y entre los propios tuits.

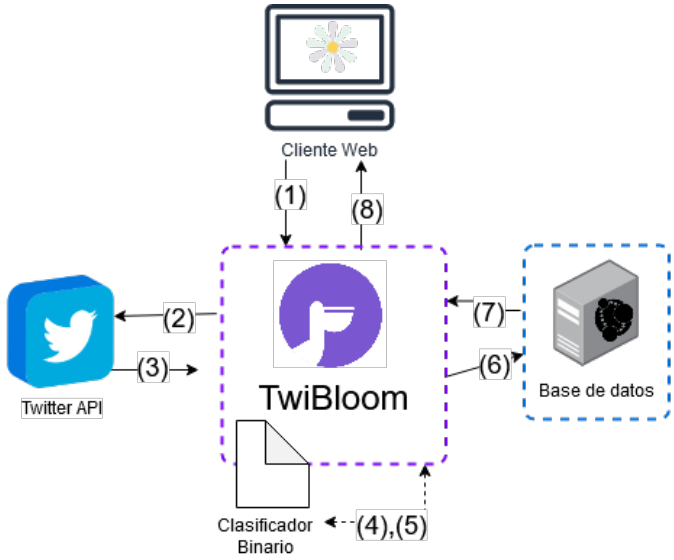

Figura 3.1: Diagrama de contexto de la herramienta

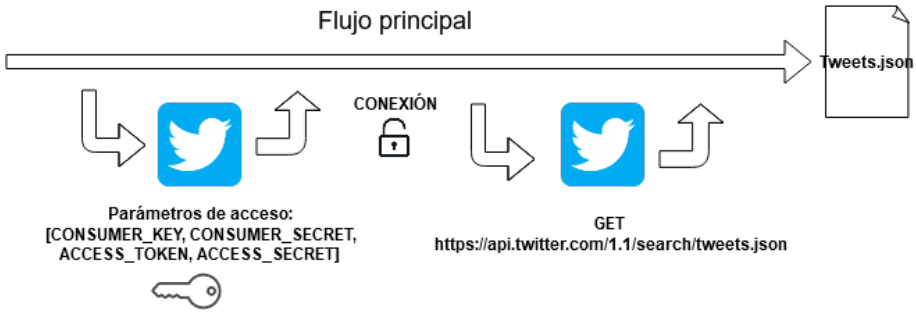

Figura 3.2: Flujo de interacción con Twitter

Esta información es enviada a la base de datos mediante peticiones en el lenguaje nativo de Neo4j, Cypher, paso (6). Una vez poblada la base de datos, se puede realizar una llamada para visualizar con la herramienta Bloom el grafo almacenado en la misma (pasos 7 y 8).

3.1.1. Componentes

El sistema TwiBloom tiene como objetivos la conexión con Twitter, la obtención de tuits y su tratamiento, así como el análisis de los mismos y la población de la base de datos. Por ello el sistema desarrollado comprende una serie de componentes, mostrados en la figura 3.3.

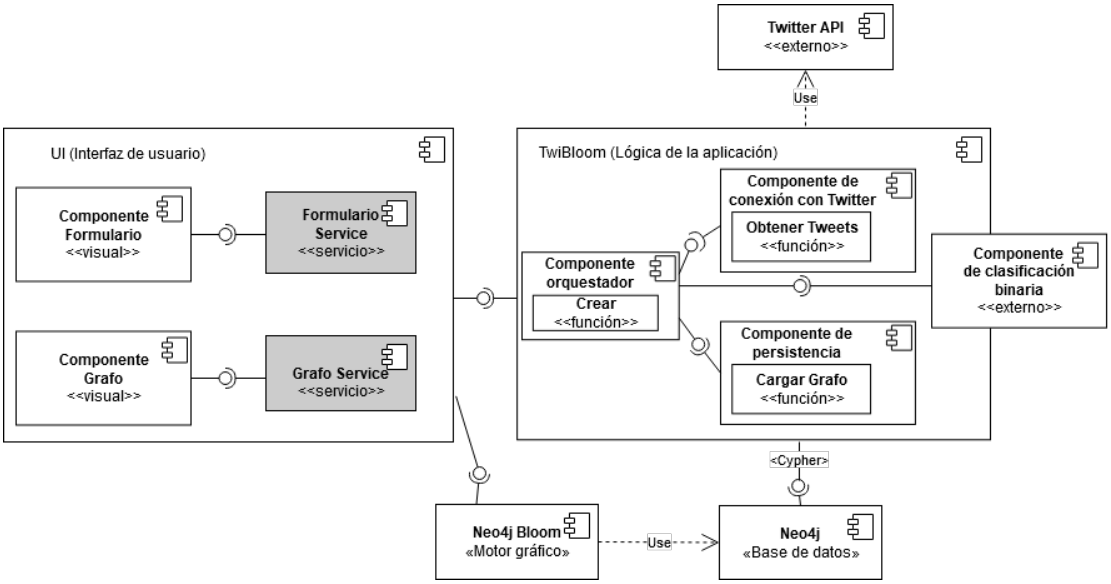

Figura 3.3: Diagrama de componentes del sistema.

El sistema se divide en dos partes; la primera, la parte de lógica de la aplicación y la segunda la interfaz gráfica del usuario. Ambas están conectadas y tienen varios componentes. La interfaz gráfica contiene el componente *Formulario*, que describe la apariencia visual del formulario que permite introducir los parámetros de la petición a Twitter, conectado al componente *Formulario Service*. Este último es el servicio que permite la conexión de esta interfaz gráfica a la parte lógica de la aplicación. El otro componente es el componente *Grafo*. Este se encarga de representar gráficamente el contenido de la base de datos. Para ello se conecta con el componente *Grafo Service*, encargado de conectar a su vez con el motor gráfico Bloom.

TwiBloom es la única herramienta que forma la parte de la lógica de la aplicación. El componente principal es el orquestador, que se encarga de coordinar todo el proceso. Este componente obtiene los parámetros de búsqueda de tuits escogidos por el usuario, y se comunica con los componentes de conexión con Twitter, de clasificación y de persistencia.

La función *Obtener tuits*, dentro del componente de conexión con Twitter, es la encargada de la conexión con la API de Twitter y de enviar las peticiones a la misma con los parámetros especificados por el usuario. Una vez obtenidos todos los tuits, se almacenan en un fichero y se devuelve el nombre del fichero. Posteriormente estos tuits se envian al clasificador binario, que actualiza el fichero de tuits dejando únicamente aquellos que etiqueta como potencialmente peligrosos. Finalmente, la función *Cargar Grafo*, dentro del componente de persistencia, se encarga de leer el fichero modificado por la función anterior, de realizar el análisis de los tuits y sus relaciones y de conectar con la base de datos. Si la conexión es exitosa, comienza a cargar la base de datos con los nodos y las relaciones correspondientes; en caso contrario, se avisa al usuario del error.

3.1.2. Representación gráfica

La representación gráfica es la parte más crucial del sistema. Se requiere una representación fiel a las interacciones de los usuarios en Twitter, además de legible para el ser humano. Varias pautas se han seguido para alcanzar esta representación, siendo la primera la legibilidad: se tiene que entender perfectamente qué está representado y cualquiera ha de ser capaz de comprenderlo. También ha de ser informativo, tiene que poder accederse a toda la información en la base de datos, incluidos todos los atributos de los nodos. Es decir, ha de permitir la búsqueda e indagación en la misma para la obtención de más información. Finalmente se ha abogado por una representación en la que se puedan sacar conclusiones a simple vista.

Para lograr este objetivo se ha tratado de hacer dos vistas con distinta granularidad. La primera y más importante tiene que representar la interacción entre los usuarios de los tuits que ha aportado la API de Twitter. Esta vista tiene que mostrar la cuantía de estas interacciones y de qué manera se han hecho. Además, tiene que mostrar si los usuarios que participan en estos tuits son activos o pasivos, entendiéndose como activo aquel que publica muchos tuits propios y pasivo el que interactúa con frecuencia con los tuits de terceros. Así se pueden visualizar lo que se han denominado *islas de contactos*. Estas islas representan agrupaciones de usuarios que interactúan entre ellos de manera continuada, como se muestra en la figura 3.4. En esta figura se aprecian nodos de usuario de distintos tamaños y colores. El color rojo indica que el usuario tiende a ser activo, mientras que cuanto más azul sea, más pasivo es el usuario. Cuanto más grande es el nodo, más tuits propios ha publicado. En el caso de las relaciones, representadas con flechas, se puede observar que son dirigidas y que el grosor no es igual en todas. Cuanto mayor sea este grosor, más interacción hay entre los usuarios. En la figura se puede observar que el pequeño nodo azul central es un usuario pasivo, que ha participado con los usuarios a su alrededor y sobre todo con el que está debajo, pues el grosor de la flecha es mayor.

La segunda vista desarrollada se aprecia en la figura 3.5 y en ella ya no se muestra la cuantía de la interacción de los usuarios pues el grosor de las flechas es idéntico. Esta vista permite la visualización de los tuits y de los *hashtags* que aparecen en ellos, azules y naranjas respectivamente, así como el tipo de interacción que cada usuario tiene con ese tuit. Al ampliar el zoom en las vistas aparecen etiquetados tanto los nodos como las

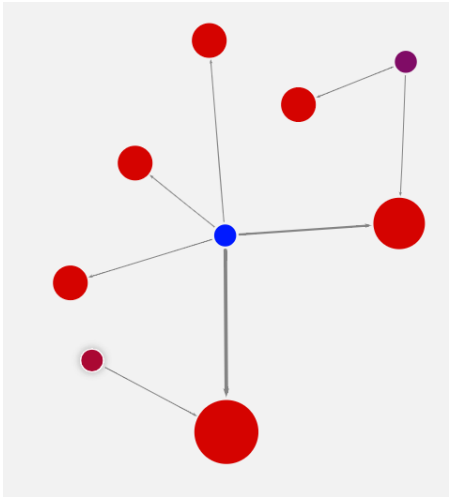

Figura 3.4: Isla de contactos en la perspectiva 1.

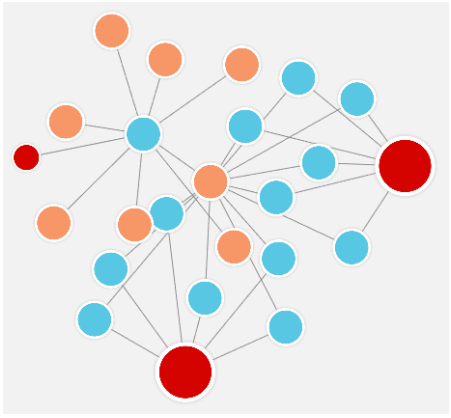

Figura 3.5: Usuarios, tuits y hashtags en la perspectiva 2.

relaciones, estas últimas con el tipo de relación que son y los nodos con los datos de los atributos.

El componente *Grafo* es el encargado de mostrar de manera visual los datos de la base de datos. Para ello hace uso del servicio *Grafo Service*, que permite a su vez la conexión con Bloom. Bloom es el motor gráfico que conecta con la base de datos, obtiene los datos deseados y los muestra en forma de grafo. En el componente *Grafo* se integra la representación gráfica que crea Bloom, permitiendo ser vista por el usuario final.

3.2. Implementación

En esta sección se describen tanto el proceso como las decisiones de implementación tomadas para adecuarse al diseño del sistema planteado. El desarrollo del sistema se ha dividido en varias fases, explicadas detalladamente en las siguientes secciones. Téngase en cuenta que las decisiones de implementación se han basado en las formas de representación final.

3.3. TwiBloom

El objetivo de esta fase ha sido la construcción de una herramienta capaz de conectar con Twitter, analizar los tuits y sus relaciones para finalmente poblar la base de datos Neo4j. En la figura 3.3 se muestra una vista del sistema completo por componentes, en el que se observa el componente TwiBloom, que contiene a su vez toda la lógica dividida en cuatro componentes.

3.3.1. Componente orquestador

Es el encargado de interconectar el resto de componentes y es necesario para la ejecución secuencial de la lógica del sistema, ya que las tareas dependen unas de otras. La clasificación binaria depende de la obtención de tuits y la población de la base de datos depende de la clasificación. Está formado por una única función (*Crear*) que se comunica con el resto de los componentes, empezando por el componente de conexión con Twitter, pasando por el componente de clasificación binaria y terminando con el componente de persistencia.

3.3.2. Componente de conexión con Twitter

Este componente reside dentro de la función *ObtenerTweets*. Para establecer una conexión con Twitter es necesaria la autenticación. Para ello se ha creado una aplicación en la plataforma de desarrollador de Twitter que ha proporcionado las credenciales necesarias para la conexión. Con estas credenciales se puede crear un nexo de conexión que permite realizar las peticiones deseadas a la API, mediante las cuales se obtienen los tuits. El total de tuits devueltos por cada petición es variable (con un máximo), pero la API de Twitter estándar limita la frecuencia de esta petición a 180 veces por cada ventana de 15 minutos. Los tuits son devueltos en formato JSON y tienen una longevidad máxima de 7 días. Las peticiones se pueden personalizar [10] según los siguientes parámetros:

- `q`: Parámetro obligatorio que permite especificar la expresión que guiará la búsqueda. Puede ser una palabra, un hashtag (#thinspo), un usuario (@NASA) e incluso una frase o expresión de 500 caracteres como máximo.
- `geocode`: Parámetro opcional, permite buscar tuits de usuarios ubicados en un radio dado de la latitud y la longitud especificadas. Este parámetro se especifica mediante *'latitud, longitud, radio'*, donde las unidades del radio se especifican como 'mi' (millas) o 'km' (kilómetros), respectivamente.
- `result_type`: Parámetro opcional, especifica el tipo de búsqueda que se desea hacer. Sus posibles valores son `reciente` (devuelve los tuits por orden de más reciente a mas antiguo), `popular` (devuelve los tuits por orden de popularidad), o `mezclado` (seleccionado por defecto, devuelve una mezcla de los dos anteriores).

3.3.3. Componente de clasificación binaria

Este paso está desarrollándose en paralelo en el marco de otro TFG que tiene por objetivo el estudio de los modelos de procesamiento de lenguaje natural que mejor se adapten a este problema. En este proyecto se ha integrado un prototipo inicial, aún incompleto pero funcional. La herramienta de clasificación binaria consiste en un procesador de textos que etiqueta el texto contenido en cada tuit como positivo o negativo, según haga apología de TCA o no. Para ello se aplica un tratamiento extensivo a cada

texto y se pasa a través del clasificador. Este tratamiento pasa desde el paso a minúsculas hasta la eliminación de números, caracteres especiales, signos de puntuación y palabras comunes. Una vez el texto está limpio se pueden calcular unas métricas numéricas como la frecuencia y la frecuencia inversa, que miden el peso de las palabras dentro de un conjunto de textos. Estas métricas se pasan finalmente por un modelo de clasificación binaria de aprendizaje máquina, que indica con cierta probabilidad si hace, o no, apología de estos trastornos. Una vez obtenidos todos los tuits, se hace una llamada a este clasificador y se guardan únicamente los tuits cuya etiqueta es positiva, descartando aquellos que no hacen apología.

3.3.4. Componente de persistencia

Este componente se encarga de poblar la base de datos con los datos obtenidos de Twitter. Toda la lógica de análisis de relaciones y la población de la base de datos se encuentra dentro de este componente. Desde él se lanzan todas las consultas a la base de datos, tanto para añadir datos como para buscarlos. Estas consultas se han realizado mediante Cypher, el lenguaje de consultas y administración de Neo4j optimizado para sus bases de datos [3]. Inspirado en SQL y SPARQL, tiene ciertas semánticas de Python y Haskell y está orientado a parecerse a la prosa inglesa para facilitar su lectura y comprensión. Este lenguaje permite realizar consultas complejas a la base de datos, además de permitir la gestión de usuarios y permisos. Toda la funcionalidad se encuentra dentro de la función *Cargar Grafo* en la figura 3.3.

Recuérdese la diferencia entre dos términos: relaciones e interacciones. Las interacciones son las acciones que se realizan en Twitter entre usuarios; en cambio, las relaciones son creadas entre los nodos de la base de datos, lo que permitirá su representación después. Una vez obtenidos todos los tuits de interés del componente de clasificación, se han de analizar para conocer todos los usuarios que participan en cada uno, así como las relaciones que se establecen entre ellos, para finalmente poblar la base de datos con los datos de interés.

Participantes e interacciones

Se ha realizado un análisis de los distintos tipos de interacciones que se pueden dar entre los usuarios de Twitter. Un usuario puede interactuar con otro de las siguientes maneras:

- **Retweet**: Compartir con los seguidores la publicación realizada por otro usuario o por sí mismo. Esta publicación puede ser una respuesta.
- **Reply o respuesta**: Mandar un comentario al usuario que ha publicado el tuit al que se responde. Este comentario será visible por todos los seguidores del otro usuario.
- **Quote Retweet**: Compartir con los seguidores la publicación realizada por otro usuario o por sí mismo, añadiendo un comentario.

- **Mention o mención**: Compartir con los seguidores una publicación en la que se menciona a otro usuario mediante el esquema @{username}.
- **Like o "Me gusta"**: Mostrar el agrado hacia la publicación de otro usuario o de sí mismo.

Teniendo claros los tipos de interacción entre los usuarios se han tratado de encontrar en el esquema de los tuits. La primera conclusión obtenida es la falta de la interacción "Me gusta" dentro de la información proporcionada por la API. Se indica la cantidad de "Me gusta" de cada tuit pero no qué usuarios los han realizado. Por lo tanto este tipo de interacción queda descartada de la representación final. En el caso del *retweet* existe un campo en el esquema JSON de los tuits denominado '*retweeted_status*'. Este puede estar vacío si no es un *retweet* o representar en su interior la información sobre el tuit que ha sido '*retweeteado*'. Dentro de este campo se engloban los datos del tuit y del usuario que lo ha publicado, lo que permite conocer la interacción entre ambos usuarios. Idéntico es el caso de los *quotes*, existe el campo '*quoted_status*' en cuyo interior se encuentra la información sobre el tuit que ha sido mencionado. El caso de la respuesta también está reflejado en el campo '*in_reply_to_status_id*', encontrándose su identificador y el nombre del usuario que lo ha publicado, pero no el texto ni demás datos, por lo que en la base de datos solo figurarán el usuario y su identificador. Finalmente, para las menciones existe en todos los tuits un campo denominado '*entities*', que contiene a su vez otro campo '*user_mentions*', en el que se encuentran los identificadores y los nombres de todos los usuarios mencionados en el tuit. En '*entities*' también se encuentran los *hashtags* que hay en el tuit, que servirán como nodos para la perspectiva desglosada.

Gracias a estos campos se han podido identificar diversos tipos de interacción. No es un desglose completo pero sí suficiente como para poder tratarlo y poblar la base de datos con los nodos y relaciones deseados.

Modelo de datos: Nodos y relaciones

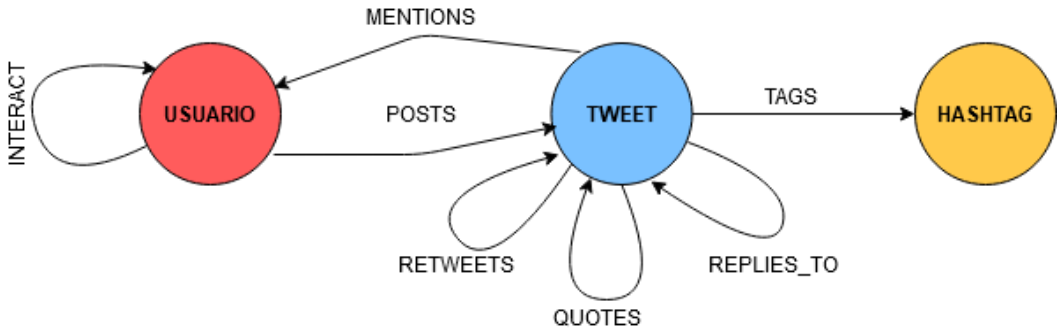

Figura 3.6: Modelo de datos

La figura 3.6 muestra el modelo de datos, mostrando tres tipos de nodos. El primero y más importante es el de *usuario*, en rojo. Un nodo de usuario contiene información

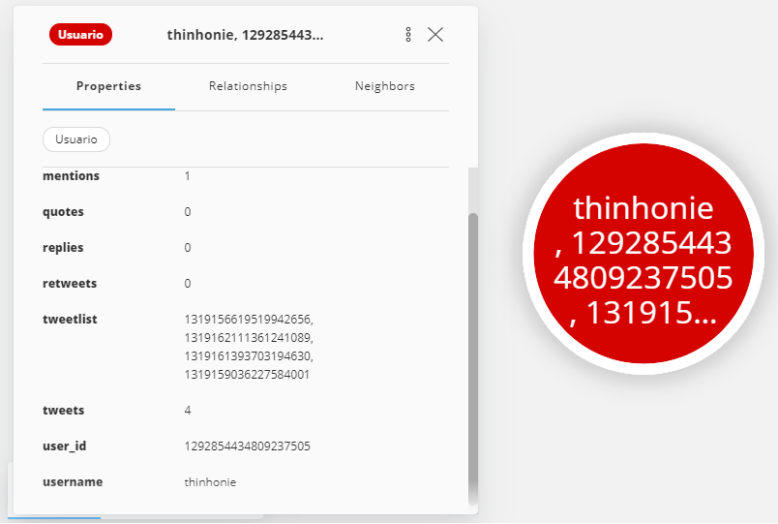

Figura 3.7: Nodo representando un usuario

sobre su identificador dentro de Twitter, su nombre de usuario, la lista de tuits y la cantidad de *retweets, quotes*, respuestas y menciones que se han procesado dentro de la respuesta a la petición (ejemplo mostrado en la figura 3.7). Este nodo es el único necesario para la primera vista. Para la segunda vista de mayor granularidad se han añadido los nodos de tuit, en azul, y los nodos de *hashtag*, en naranja. En los nodos que representan los tuits se alberga el identificador del tuit, además del texto contenido en él y la fecha de publicación (Figura 3.8). En el caso de los *hashtags* solo se encuentra el nombre (Figura 3.9).

En el caso de las relaciones se ha creado la relación de interacción ([INTERACT]) para la primera vista. Para monitorizar la cuantía de la interacción entre dos usuarios, a esta relación se le ha asignado un peso. Este peso depende de la cantidad de interacciones que haya entre ambos usuarios. Asimismo a cada tipo de interacción se le ha asignado otro peso, que depende de a cuántos usuarios vaya destinada esa interacción, véase la Tabla 3.1. La lógica reside en el hecho de que a cuanta más gente esté llegando el resultado de la interacción, más apología se está realizando.

En el caso de la segunda vista se han registrado distintos tipos de relaciones. Al aumentar la granularidad se ha decidido mostrar todas las interacciones entre todos los nodos. En la figura 3.6 se muestran todas las relaciones entre los nodos, que engloban las interacciones entre los usuarios, entre los usuarios y los tuits y entre los tuits y los *hashtags*.

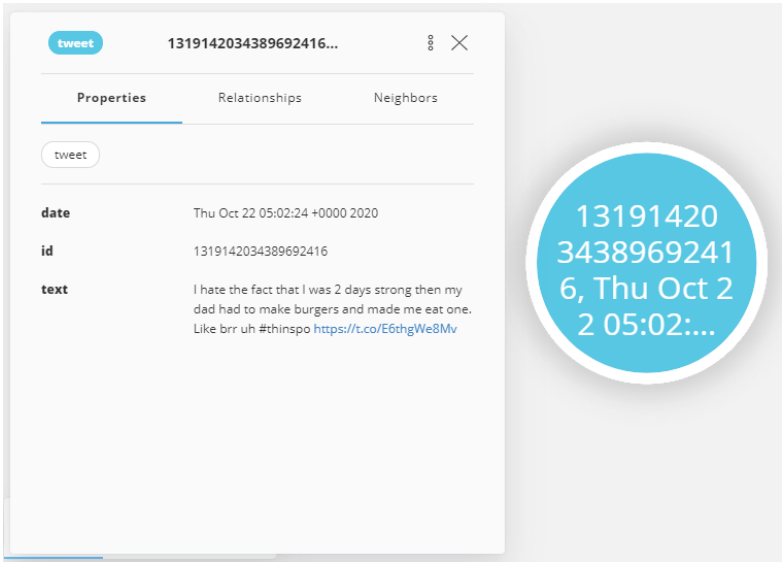

Figura 3.8: Nodo representando un tweet

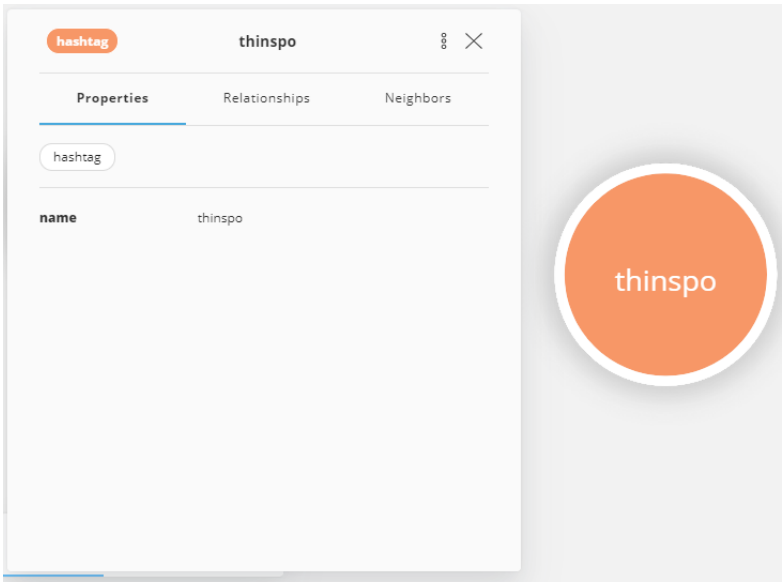

Figura 3.9: Nodo representando un hashtag

Interacción	Destinatarios	Peso
Retweet	Usuarios seguidores	2
Reply	Usuario publicador	1
Quote Retweet	Usuarios seguidores	2
Mención	Usuario mencionado	1

Tabla 3.1: Peso de las distintas interacciones

En la función *Cargar Grafo*, una vez obtenidos los tuits se comprueba qué tipo de tuit es: *Tweet, retweet, reply o quote*. Una vez conocido se procede a la adición de sus participantes a la base de datos, comenzando por los nodos de usuario y las relaciones entre ellos, siguiendo por los nodos de tuit y las relaciones entre ellos y con los usuarios, para llegar finalmente a las menciones y los *hashtags*. Para ello se emplean dos cláusulas sencillas de Cypher, "MATCH" y "MERGE". La primera especifica el patrón a buscar en la base de datos, mientras que la segunda asegura que el patrón existe en el grafo, creándolo en caso contrario. "MERGE" ayuda a comprobar que no se crean nodos ni relaciones duplicadas ya que, en el caso de que existan previamente, no hace nada. Un ejemplo de petición completa e muestra a continuación:

session.run('MERGE (u:Usuario { username: $username })
ON CREATE SET u.user_id = $id, u.tweets = 0, u.retweets = 1, u.quotes = 0, u.replies = 0, u.mentions = 0, u.tweetlist = $lista
ON MATCH SET u.retweets = u.retweets +1',
username=user['screen_name'], id=user['id_str'], lista=[])

Esta petición en concreto busca un nodo del tipo usuario cuyo nombre de usuario está especificado en la variable $*username*. Si no lo encuentra lo crea (ON CREATE SET), indicando que ha hecho un *retweet* y que su lista de tuits está vacía; en caso de que lo encuentre (ON MATCH SET), le suma uno a la cantidad de *retweets* que ha hecho.

3.3.5. Componente de representación

Una vez poblada la base de datos ya está el sistema listo para la representación. Como se ha explicado previamente este es el punto más crítico del sistema, ya que se requiere una representación legible para el ser humano y adecuada para la posterior extracción de conocimiento.

Para esta tarea se ha empleado la herramienta Bloom de Neo4j [1], un motor gráfico optimizado para bases de datos Neo4j, que permite la creación de diversas perspectivas

personalizables. Una perspectiva es una forma personalizada de representar los datos de la base de datos. Estas perspectivas se pueden exportar y compartir entre los usuarios.

Como se ha descrito anteriormente, para el caso de uso de los TCA se han creado dos perspectivas de distinta granularidad. La primera es una perspectiva de alto nivel en la que se representan únicamente los usuarios y las relaciones [INTERACT] entre ellos. En la segunda se representan el resto de relaciones, además de los tuits y *hashtags*. A continuación, se describen ambas perspectivas en más detalle.

Primera perspectiva

En esta primera perspectiva se ha buscado una representación que a simple vista muestre los usuarios que más influencia tienen haciendo apología de los TCA. Estos usuarios son los que publican un tuit que tiene más repercusión, los que han interactuado con mayor actividad con las publicaciones de los demás o los que más publicaciones han compartido. No se ha dado importancia al contenido de los tuits que ha publicado cada uno, a las interacciones con los tuits ni qué *hashtags* han empleado con mayor frecuencia. Recordando que los usuarios se representan como nodos y las interacciones como relaciones, para lograr esta representación se ha experimentado con tres variables: el tamaño y color de los nodos y el grosor de las relaciones.

El grosor de una relación indica cuánto ha interactuado un usuario con otro y esto se ha modelado con el atributo de peso de las mismas. Cuanto mayor es el peso de la relación [INTERACT] entre dos usuarios, más gruesa es la flecha que los une en el gráfico. La herramienta Bloom permite enlazar el grosor de la flecha con el peso de la relación, pero no permite una relación directamente proporcional. Bloom permite 5 grosores (0.25x, 0.5x, 1x, 2x, 4x), por lo que a partir de cierto umbral superior la flecha es del grosor máximo. Gracias a esta característica se puede apreciar la cuantía de interacción entre los usuarios.

En cuanto a los nodos se ha jugado con las variables de su tamaño y color. El tamaño es cuasi-directamente proporcional a la cantidad de tuits propios que ha subido el usuario entre los tuits almacenados. Los tuits propios son aquellos que no hacen *retweet, reply o quote* a otros. La limitación es la misma que en el caso de las relaciones, con sólo 5 grosores distintos. La variable tamaño está ligada al atributo '*tweets*' del usuario en la base de datos, que indica la cantidad de tuits propios.

El caso del color ha sido más complejo. El color está definido por el tipo de participación del usuario, pasiva o activa. La participación pasiva se define como la interacción del usuario con las publicaciones de otros, englobando *retweets, replies y quotes*. La participación activa se define como la creación de contenido propio, en este caso los tuits propios. Cuanto más activa sea la relación el color ha de tender a uno concreto, mientras que si es más pasiva ha de tender hacia otro. La idea inicial era que el color se definiese según una función en tiempo de representación, pero Bloom no tiene esta característica. Por lo que se ha tenido que añadir el color como un atributo más de los nodos (usuario) y calcularse mientras se puebla la base de datos. La función que define el color es la siguiente:

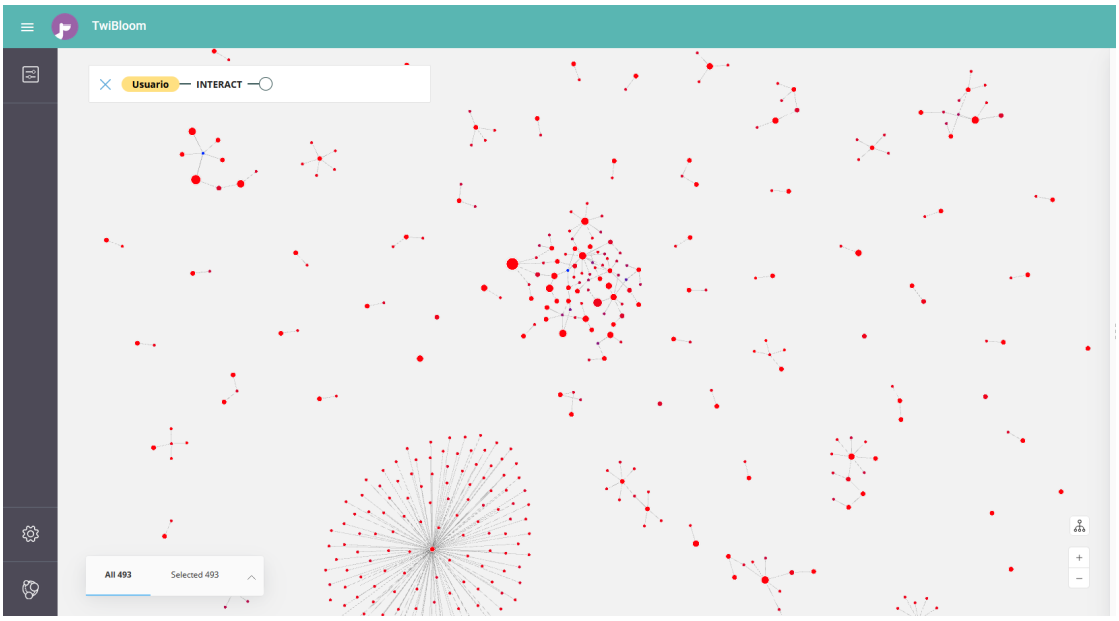

Figura 3.10: Vista de la primera perspectiva.

$$color = \frac{num.retweets + num.respuestas + num.quotes}{num.tweets + 1}$$

El denominador suma uno para eliminar la posibilidad de una división entre cero. Con esta función se ha conseguido que si el valor de *color* es alto se representa una participación pasiva, siendo activa en caso contrario. El color se calcula en tiempo de ejecución, por lo que a las peticiones Cypher que crean o actualizan usuarios se les ha añadido el cálculo/actualización del color. En este caso, el degradado de color oscila entre el rojo y el azul; color rojo indica una participación activa y el azul, pasiva (véase la figura 3.10).

Segunda perspectiva

Esta perspectiva tiene un nivel mayor de granularidad. El objetivo de esta perspectiva es mostrar de manera más desglosada las relaciones entre los usuarios y el resto de entidades. En este caso se muestran los tuits de cada usuario y las relaciones que distintos usuarios tienen con ese tuit, además de qué *hashtags* ha mencionado. Esta visión permite otro tipo de conclusiones, como saber qué *hashtags* son más empleados por estos usuarios que hacen apología. Se ha tratado de modelar el tamaño y color de los nodos según el número de relaciones entrantes, lo que permitiría la obtención de más información a simple vista, pero Bloom no cuenta con esta característica. Para lograr esto se podría seguir el ejemplo del color: en cada petición Cypher que cree una relación nueva

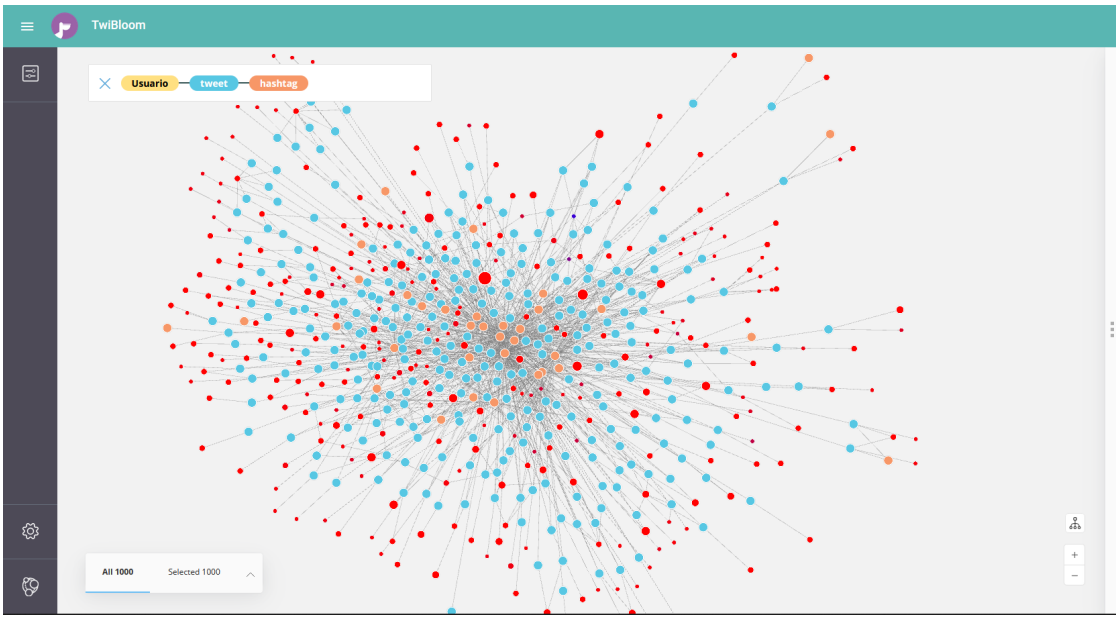

Figura 3.11: Vista de la segunda perspectiva.

se podría actualizar el nodo que recibe la flecha entrante y sumarle uno a un atributo *entrantes*. Un ejemplo de esta perspectiva se muestra en la figura 3.11.

Otras funcionalidades de Bloom

La creación de perspectivas no es el único propósito de Bloom. Esta herramienta ofrece otras funcionalidades que pueden ayudar a la extracción de información y conocimiento a los expertos. La primera de ellas es la consulta al grafo a través de patrones. Se puede hacer que Bloom represente solo subgrafos del grafo original con las restricciones deseadas. Por ejemplo se le puede pedir que represente solo aquellas relaciones [INTERACT] con más de un peso determinado, o aquellos tuits que nombren cierto *hashtag*. Además permite la creación de búsquedas predeterminadas y guardarlas.

3.4. Aplicación web e integración de la herramienta

Una vez terminado el proceso de extracción de conocimiento con la herramienta TwiBloom, se ha procedido a integrarla en una aplicación web como solución a la presentación ante los futuros usuarios. Esto ha requerido ciertos cambios y adaptaciones de la herramienta aunque el comportamiento es el mismo. La aplicación web está pensada para que, una vez escogidos los parámetros de una petición a Twitter, ejecutarla de manera periódica (cada día) para ir almacenando los tuits correspondientes, hasta que el usuario decida cambiar estos parámetros. Una vez el usuario cambia los parámetros, el contenido de la base de datos se borra y se vuelve a empezar el proceso. La aplicación

web se ha dividido en dos partes, la interfaz de usuario y la lógica de la aplicación. La interfaz de usuario se encarga de la interacción del usuario con la aplicación y se comunica con la parte de la lógica de la aplicación, es decir, de toda la funcionalidad. La herramienta TwiBloom está integrada en la parte de lógica de la aplicación.

La interfaz de usuario se ha hecho con Angular. Este marco de trabajo permite la separación de los elementos de la vista en componentes, cada uno con su funcionalidad. En la aplicación hay varias vistas. En la vista inicial hay una breve descripción del sistema y se muestra el estado actual del mismo, con información sobre el caso de uso de la aplicación y el estado de la base de datos. De esta primera vista se puede acceder a una segunda, que permite cambiar los parámetros de la petición a la API de Twitter. Para ello se ha hecho un formulario con los campos necesarios y se ha incluido un pequeño mapa con *Leaflet* [8] que muestra las coordenadas del punto en el que se pulsa, lo que permite rellenar los campos de *latitud y longitud* sin tener que buscar en el navegador. El único campo obligatorio es el primer *hashtag*. El tipo de resultado (*mezclado, reciente o popular*) se elige mediante un selector con *mezclado* por defecto. También existe otra vista, conectada con Bloom, que muestra el grafo con el contenido de la base de datos y con toda la funcionalidad de Bloom, para que el usuario pueda crear sus perspectivas y hacer las consultas pertinentes. Se ha añadido una vista final con preguntas frecuentes. Se puede navegar entre las vistas gracias a un menú desplegable accesible desde la esquina superior izquierda, véase la figrua 3.12, pudiendo también volver a la vista inicial siempre pulsando el título *TwiBloom* de la parte superior izquierda o desde la opción *Home* del menú desplegable. La comunicación entre Angular y la lógica de la aplicación se realiza mediante JSON.

La parte de la lógica de la aplicación se ha hecho con Python y Flask, un marco de trabajo que permite la creación de un servidor de manera sencilla. En este caso se ha adaptado la herramienta TwiBloom para lanzarla dentro de este servidor Flask. Este servidor contiene varias funciones, una para obtener datos de la propia base de datos (como el número total de tuits, el número total de nodos, etc.) y otra para realizar toda la lógica de TwiBloom, desde la petición de tuits al volcado en la base de datos. Asimismo aquí está configurada la periodicidad de ejecución de esta última función.

3.5. Despliegue del sistema

El sistema requiere desplegar tanto la base de datos Neo4j, como el servidor de la interfaz gráfica de Angular, como la lógica de la aplicación en el servidor Flask. Por defecto, la base de datos se lanza en el puerto 7474 para peticiones HTTP y 7687 para peticiones mediante el protocolo BOLT, el servidor Angular se lanza en el puerto 4200 y el servidor Flask en el puerto 5000. El cliente se conecta mediante una URL al servidor de Angular, obteniendo la página de inicio. Este servidor de Angular se conectará al servidor de Flask siempre que se alcance la página de inicio y cuando se modifiquen los parámetros de la petición a Twitter. A su vez, el servidor Flask se conectará con la base de datos cada vez que se ejecute una petición a Twitter y haya que añadir los datos. Finalmente, cuando se requiera la representación gráfica el cliente se conectará al

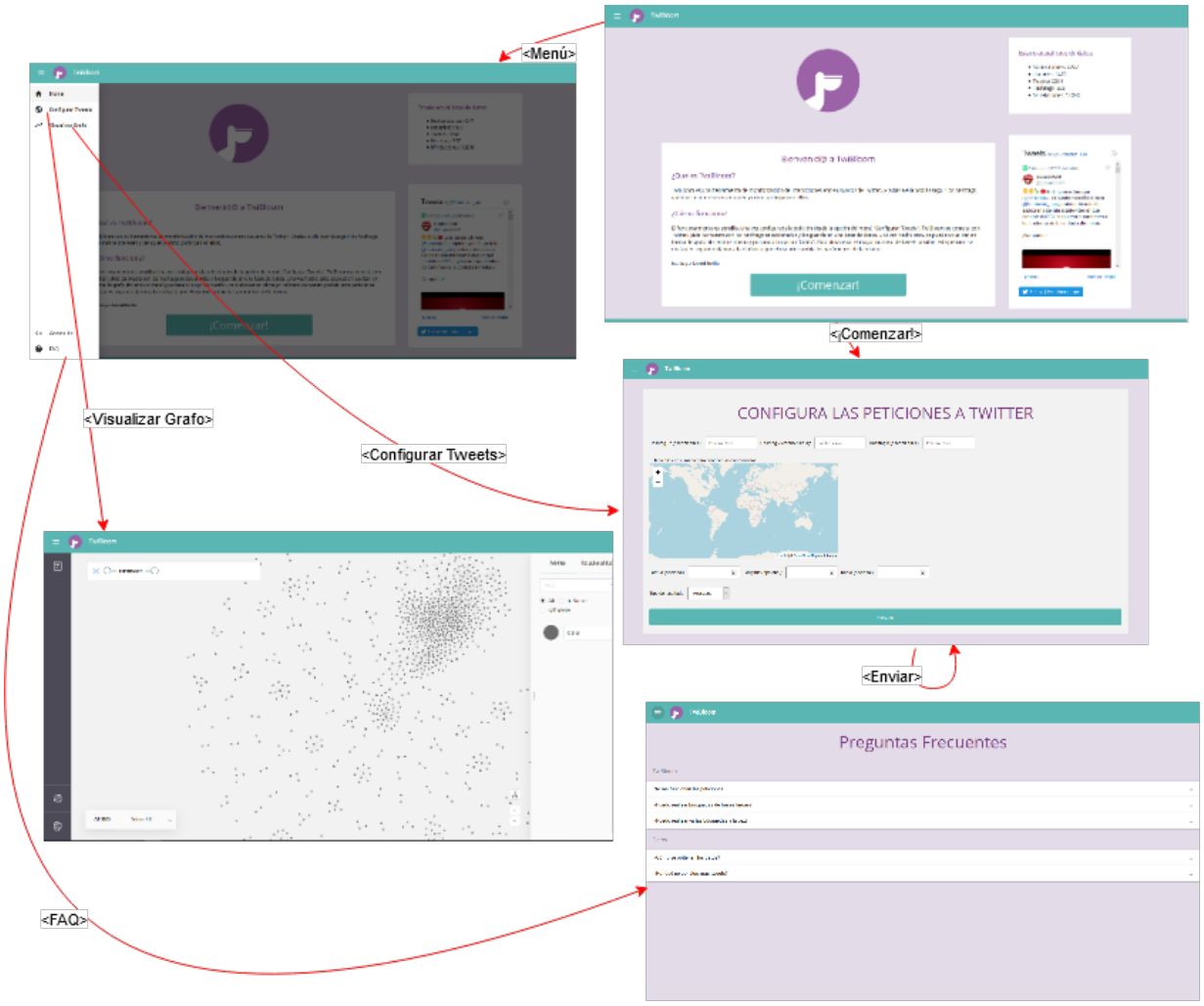

Figura 3.12: Navegación entre las diferentes vistas.

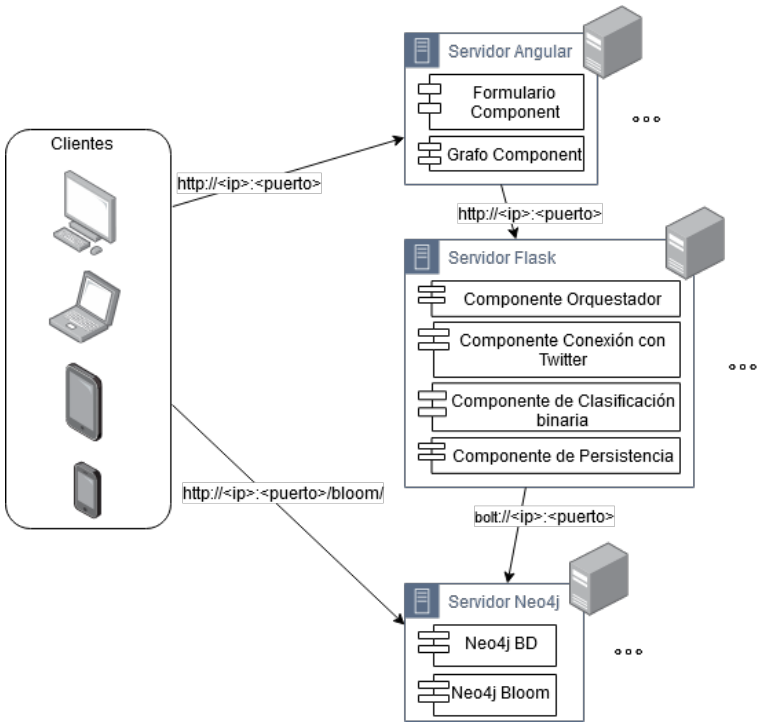

Figura 3.13: Esquema del despliegue general de la aplicación.

servidor de la base de datos directamente para obtener las perspectivas de Bloom.

Este despliegue es personalizable, se puede lanzar todo en el mismo terminal o se puede separar según convenga. Un despliegue razonable consistiría en separar la base de datos, la aplicación Angular y la aplicación Flask en distintas instancias. Para ello, si no se dispone de las máquinas suficientes, se pueden emplear los servicios de Cloud de Neo4j (Aura) para la base de datos y cualquier otro servicio, como Amazon Web Services para desplegar los otros dos servidores. Esto permite aprovechar mejor los recursos de cada una y hacer estas replicables de manera más sencilla, mejorando su escalabilidad. Si en un futuro las peticiones son muy numerosas se podrían redundar estas instancias y equilibrar el número de peticiones que se dirigen a cada una mediante un balanceador de carga. Si el acceso no va a ser concurrente ni de manera continua, se puede desplegar todo en la misma instancia.

Para probar la aplicación se ha lanzado todo desde el terminal local, lanzando cada servidor en su puerto por defecto. Además, se ha accedido desde los navegadores Chrome y Firefox con la URL *http://localhost:4200* y probado la responsividad de las vistas en distintos dispositivos, siendo exitosas todas las pruebas realizadas.

Capítulo 4

Lecciones aprendidas con el caso de estudio

En este capítulo se describen las lecciones aprendidas de la aplicación de este sistema a la detección de apología de los TCA. Este proyecto propone una herramienta para que los expertos en la materia puedan llegar a conclusiones relativas a los TCA. Durante los tests de la herramienta se ha hecho un pequeño estudio que ha permitido descubrir ciertos aspectos en común de las cuentas que realizan este tipo de apologías. Más concretamente, qué tipo de *hashtags* emplean con mayor frecuencia y qué patrones siguen estas cuentas.

Para este caso, se ha lanzado la herramienta con una tasa de una petición cada día durante 6 días. Esta petición no ha constado de geolocalización, el tipo de resultado mezclado (*mixed*) y los *hashtags* de búsqueda *#proana*, *#thinspo*, *#ed*. Estos tres *hashtags* se encontraron tras hacer una petición sencilla en la que el *hashtag* era *#ana&mia*, haciendo alusión a anorexia y bulimia respectivamente. El total de tuits obtenidos fueron 2264, entre ellos los *hashtags* más empleados fueron precisamente los tres escogidos (con ligeras variaciones):

- `#thinspo`: Este *hashtag* es una abreviatura de *thinspiration* en inglés, cuya traducción es 'la inspiración en la delgadez'. Las cuentas que realizan esta apología tienden a idolatrar a otras personas y cuentas que suben fotos y vídeos en las que se muestra una extrema delgadez. Encuentran la inspiración en otras personas que padecen este tipo de trastornos y las emplean como ejemplo para seguir con sus rutinas, generalmente nocivas para la salud.

- `#ed`: Siglas de las palabras *eating disorder* (equivalente a TCA: trastorno de conducta alimentaria). Se emplea con mucha frecuencia y se utiliza como forma de identificación. Está muy generalizado y se emplean diversas variaciones del mismo, como '*#edtwt*' haciendo alusión a los trastornos en Twitter.

- `#proana` : Este punto engloba también *#promia*, *#ana&mia* y otros *hashtags* derivados. En estas comunidades *ana* es una referencia a la anorexia, y *mia* a la bulimia. Muchas cuentas entienden estos conceptos como una forma de vida que seguir y respetar, incluso adorar.

Estos *hastags* son los más empleados junto con sus variantes, como añadiéndoles el sufijo "-twt"para indicar que es Twitter, variar entre mayúsculas y minúsculas o mezclarlos. Haciendo búquedas con TwiBloom de estos *hashtags* se encuentran multitud de cuentas haciendo apología. Asimismo se ha encontrado algún patrón en estas cuentas, como incluir ciertos parámetros en las descripciones de las mismas. Por ejemplo 'cw: xx', que significa *current weight* (peso actual), 'gw: xx', que es a su vez *goal weight* (peso meta, peso a alcanzar), además de encontrar las propias siglas *ed* de manera constante. La figura 4.1 muestra un ejemplo de descripción de una cuenta en Twitter que hace apología de los TCA.

Figura 4.1: Ejemplo de descripción común

Además se han encontrado ciertas conductas comunes en varias de estas cuentas. Muchas de ellas piden consejos cuando les cuesta seguir las estrictas dietas y ayunos (*"edtwt - ive been really bad with fasting recently, anyone got any tips ? #edtwt"* – *"Me está costando mucho el ayuno útimamente, ¿algún consejo?"*) y publican regularmente sus 'logros', como cuántas horas llevan sin comer o las pocas calorías que han ingerido. El sentimiento de obesidad se lee con frecuencia en estos perfiles. Se aprecia obsesión hacia la cantidad de calorías, encontrándose calendarios indicando cuantas calorías comer cada día para estar más delgada en varias de estas cuentas. También muestran tuits de carácter depresivo, de odio hacia sí mismos y de malestar con su familia y con terceros, además de auto-recordarse que no tienen que comer, incluso llegando a amenazarse (*"reminder: if you eat now the boy you like is never gonna like you back"* - *"recordatorio: Si comes ahora no le gustarás al chico que te gusta"*). Muchas de ellas comparten fotos y publicaciones de otras cuentas que también hacen apología y de cuentas de personas extremadamente delgadas. Otro patrón compartido por varias cuentas es la obesofobia: odian los cuerpos gordos, les repugna la obesidad e insultan a aquellos que la padecen, instando a sus seguidores a no estar así. Finalmente se han encontrado carteles de las denominadas *carreras de kilos*. Se trata de competiciones en las que seguir ciertas normas y conductas relacionadas con la cantidad de comida ingerida, calorías, agua y deporte hecho en cierto período. A quien logra alcanzar los objetivos se le hace una mención y a quien gana peso se le expulsa directamente, como se puede observar en los mensajes de la figura 4.2.

Figura 4.2: Cartel de una competición de "carreras de kilos"

Capítulo 5

Conclusiones

En esta última sección se van a remarcar ciertas ideas y conclusiones a las que se ha llegado durante el desarrollo de este proyecto, además de varias potenciales mejoras del mismo.

Los TCA son enfermedades que afectan a casi un 10 % de la población adolescente de entre 13 y 19 años, mayoritariamente femenina. De hecho, en el entorno del desarrollador de este proyecto ha habido varios casos. Estos casos tienden a ser problemáticos, los afectados pueden llegar a autolesionarse y realizar acciones cuyas consecuencias pueden ser nefastas para ellos mismos y para su entorno. Es por ello que se decidió realizar este proyecto, para ayudar a la prevención y tratamiento de los mismos.

Se ha terminado una fase *beta* de la aplicación web, aunque seguirá en desarrollo hasta el 18 de diciembre de 2020 en todo caso. Se han alcanzado todos los requisitos especificados en el capítulo 2. El sistema permite la búsqueda de tuits según ciertos *hashtags*, su posterior análisis y el almacenamiento de las interacciones entre los usuarios en una base de datos. Además, permite la muestra del contenido de la misma mediante un grafo y la presentación de información acerca de este contenido. Este sistema tiene base de datos, una interfaz gráfica y está encapsulado de tal forma que se puede integrar en aplicaciones externas.

Finalmente destacar que el sistema no sirve únicamente para el caso de los TCA, sino que se ha abstraído para permitir la búsqueda de tuits con cualquier *hashtag*. Esto amplía el marco de aplicación del mismo, permitiendo a expertos emplear esta herramienta para extraer conocimiento de Twitter independientemente de la temática.

5.1. Problemas encontrados y perspectiva futura

No ha resultado sencilla la implementación de este sistema debido a varios factores. El primero de ellos ha sido la falta de precisión a la hora de describir la solución a la problemática. No se tenía muy claro al principio qué se quería hacer exactamente, y esta falta de concreción ha hecho cambiar un poco el rumbo del proyecto en varias ocasiones. Todo lo realizado durante el proyecto ha sido educativo y se ha aprendido de ello, aunque partes del mismo no han sido útiles en el desarrollo *a posteriori*.

Otro factor ha sido la dependencia de software de terceros como en el caso de Bloom, pues es necesaria una clave de activación para que el servidor de Bloom funcione. La empresa de Bloom ha tardado en responder a las peticiones realizadas para obtener una licencia temporal, lo que ha retrasado un poco esta parte del desarrollo. Asimismo las limitaciones de la API estándar de Twitter también son problemáticas, ya que los servicios de esta API son limitados y se han de pagar para obtener mejores resultados. En este caso se ha usado la API estándar, que limita las peticiones por día y el número de tuits por petición, limitando el flujo de información a analizar.

Otro hándicap sufrido ha sido la falta de experiencia tanto en el aprendizaje máquina como en el desarrollo de aplicaciones, pues ralentiza todo el proceso. También el desconocimiento de las distintas herramientas y lenguajes de programación a emplear obliga a seguir unas curvas de aprendizaje que, en caso de conocerlas con anterioridad, habrían evitado muchas horas de trabajo.

Como trabajo futuro el primer punto a tratar es qué hacer con aquellos tuits y aquellas cuentas que hacen apología de los TCA. Una primera idea es reportar al equipo de moderación de Twitter para que puedan actuar, tanto eliminando estos contenidos como intentando evitarlos, ya que pueden conocer las características comunes de todos ellos. El alcance del sistema podría ser mucho mayor, consiguiendo la licencia premium de Neo4j y la de Twitter. Esta última eliminaría la limitación de peticiones a la API, además de las restricciones temporales que tiene la API estándar. Así se podrían mostrar los datos por ventanas temporales, lo que permitiría una extracción de conocimiento mayor y más precisa.

Para ampliar la obtención de conocimiento se ha pensado en añadir análisis de sentimientos a los tuits obtenidos además de procurar inferir la edad de los usuarios de Twitter. Esto haría más precisa la información sobre los TCA y las personas que los padecen. Como característica extra se podrían monitorizar los *hashtags* en tiempo real, haciendo peticiones en intervalos cortos de tiempo, lo que daría paso a un análisis más preciso, rico y actualizado.

En este momento la aplicación permite únicamente guardar los datos de una única petición, ya que la base de datos no contiene *clusters* ni puede discernir entre los tuits que están relacionados con un tema y los que están relacionados con otro. Se podría completar esta tarea consiguiendo varias instancias de base de datos, una por petición e ir llenándolas paulatinamente. Así los usuarios podrían tener varias perspectivas guardadas e ir jugando con ellas sin tener que borrar en cada llamada los datos existentes previos.

Por último, la mayor ambición de este proyecto era la detección de apología de los TCA en toda la red. La expansión al resto de redes sociales, blogs, páginas web, etc, es una tarea cuyo resultado será muy beneficioso para la prevención y tratamiento de estos trastornos.

Bibliografía

[1] Neo4j Bloom - Graph Visualization and Collaboration. URL https://neo4j.com/bloom/.

[2] DB-Engines Ranking. URL https://db-engines.com/en/ranking/graph+dbms.

[3] Introduction - Neo4j Cypher Manual. URL https://neo4j.com/cypher-manual/4.1/introduction/.

[4] Digital 2020: October Global Statshot. URL https://datareportal.com/reports/digital-2020-october-global-statshot.

[5] El dilema de las redes (2020). URL https://www.filmaffinity.com/es/film640069.html.

[6] Flask web development. URL https://flask.palletsprojects.com/en/1.1.x/.

[7] Fundación ape - prevención y erradicación de los tca. URL https://fundacionape.org/quienes-somos/.

[8] Leaflet — an open-source JavaScript library for interactive maps. URL https://leafletjs.com/.

[9] Neo4j Graph Platform – The Leader in Graph Databases. URL https://neo4j.com/.

[10] Standard search API. URL https://developer.twitter.com/en/docs/twitter-api/v1/tweets/search/api-reference/get-search-tweets.

[11] Use Cases, Tutorials, & Documentation. URL https://developer.twitter.com/en.

[12] Twython. URL https://twython.readthedocs.io/en/latest/index.html.

[13] Natural Language Processing with Graph Databases and Neo4j, Jan. 2016. URL https://www.slideshare.net/lyonwj/natural-language-processing-with-graph-databases-and-neo4j.

[14] Digital 2020: Global Digital Overview, 2020. URL https://datareportal.com/reports/digital-2020-global-digital-overview.

[15] Agencia EFE. La Generalitat sanciona a una empresa por apología de la anorexia. [Online; https://elpais.com/ccaa/2019/12/18/catalunya/1576681708_487497.html], Dec. 2019. Accedido el 14 de noviembre de 2020.

[16] J. G. Cantero. El dilema social de las redes... y de Netflix, Sept. 2020. URL https://elpais.com/tecnologia/2020-09-18/el-dilema-social-de-las-redes-y-de-netflix.html.

[17] M. M. Chandio and M. Sah. Brexit twitter sentiment analysis: Changing opinions about brexit and uk politicians. In *International Conference on Information, Communication and Computing Technology*, pages 1–11. Springer, 2019.

[18] M. D. Conover, J. Ratkiewicz, M. R. Francisco, B. Gonçalves, F. Menczer, and A. Flammini. Political polarization on twitter. *Icwsm*, 133(26):89–96, 2011.

[19] J. Danjou. *The Hacker's Guide to Python*. Julien Danjou, 2016.

[20] Y. Fernández. API: qué es y para qué sirve, Aug. 2019. URL https://www.xataka.com/basics/api-que-sirve.

[21] A. M. S. Laurent. *Understanding open source and free software licensing: guide to navigating licensing issues in existing & new software*. "O'Reilly Media, Inc.", 2004.

[22] D. Lledó-Raigal. Design and development of a monitoring system for Spanish political tweets in COVID-19 emergency. mathesis, Universidad Politécnica de Madrid, ETSI Telecomunicación, June 2020. Online; http://www.gsi.dit.upm.es/es/investigacion/proyectos?view=publication&task=show&id=567. Accedido el 15 de agosto de 2020.

[23] D. Lorentzen. Polarisation in political twitter conversations. *Aslib Journal of Information Management*, 66, 05 2014. doi: 10.1108/AJIM-09-2013-0086.

[24] J. E. Restrepo and T. Castañeda Quirama. Riesgo de trastorno de la conducta alimentaria y uso de redes sociales en usuarias de gimnasios de la ciudad de Medellín, Colombia. *Revista Colombiana de Psiquiatría*, 49(3):162–169, July 2020. ISSN 0034-7450. doi: 10.1016/j.rcp.2018.08.003.

[25] J. Webber and I. Robinson. The top 5 use cases of graph databases. *Neo Technology*, 2015. URL https://go.neo4j.com/rs/710-RRC-335/images/Neo4j_Top5_UseCases_Graph%20Databases.pdf.

Apéndice A

Distribución del trabajo

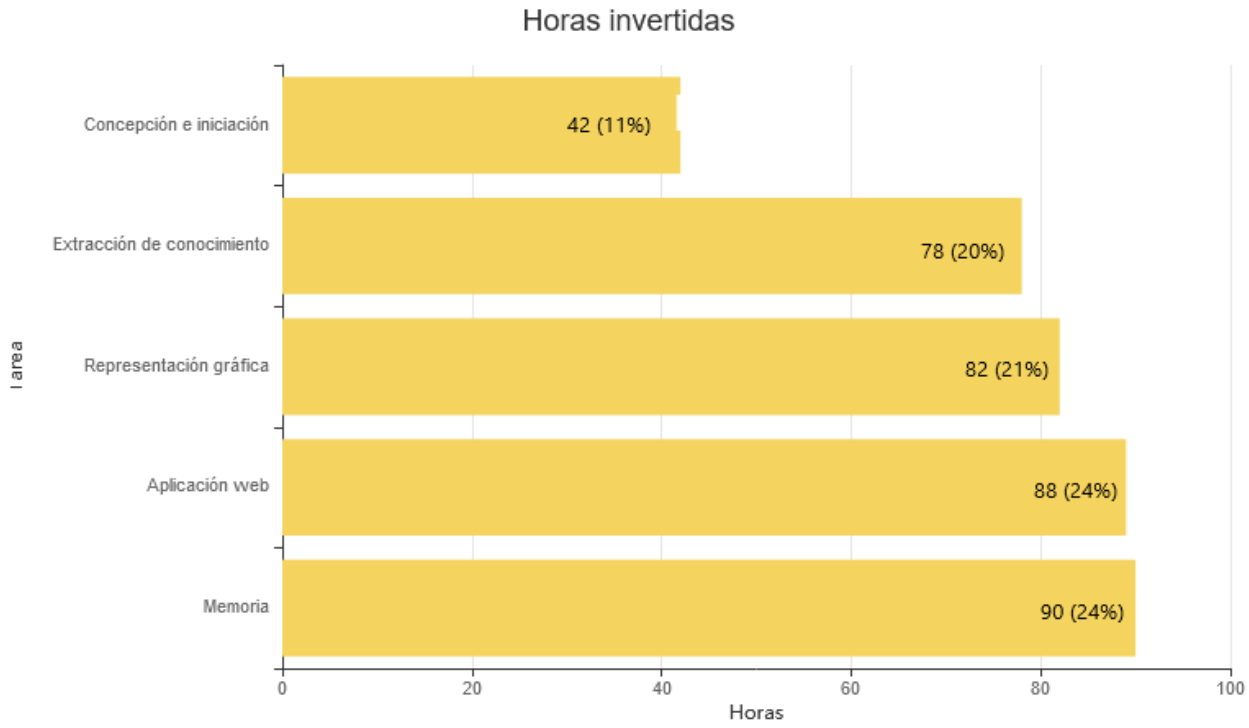

Figura A.1: Horas invertidas por tarea

En la figura A.1 se detalla el tiempo invertido en cada apartado. Al comienzo del proyecto se estuvo estudiando el contexto del mismo, las tecnologías similares y algunas herramientas empleadas. Esto supuso un 11 % del tiempo total. Una vez iniciado, se comenzó con el algoritmo de extracción de conocimiento, cuya primera versión costó unas 40 horas pero sus posteriores mejoras aumentaron esta cifra hasta casi el doble, comprendiendo finalmente un 20 % del tiempo total. Con la base de datos cargada se

comenzó la representación gráfica. La tardanza de Bloom para enviar la licencia hizo que se invirtiera cierto tiempo en la búsqueda de otras tecnologías y esto repercutió en el tiempo total (21 %). Una vez terminada la parte lógica sólo faltaba integrarla en una aplicación web, que supuso alrededor de 90 horas, una cuarta parte del total. Finalmente la redacción de este documento y sus posteriores iteraciones ha supuesto otro 25 % del trabajo total, unas 90 horas.

En total se han invertido en este trabajo 381 horas, dentro del rango esperado. Este proyecto se ha realizado entre junio y noviembre de 2020. En las figuras A.2 y A.3 se muestra la distribución del trabajo en estos meses. La aplicación seguirá en desarrollo hasta el 18 de diciembre de 2020.

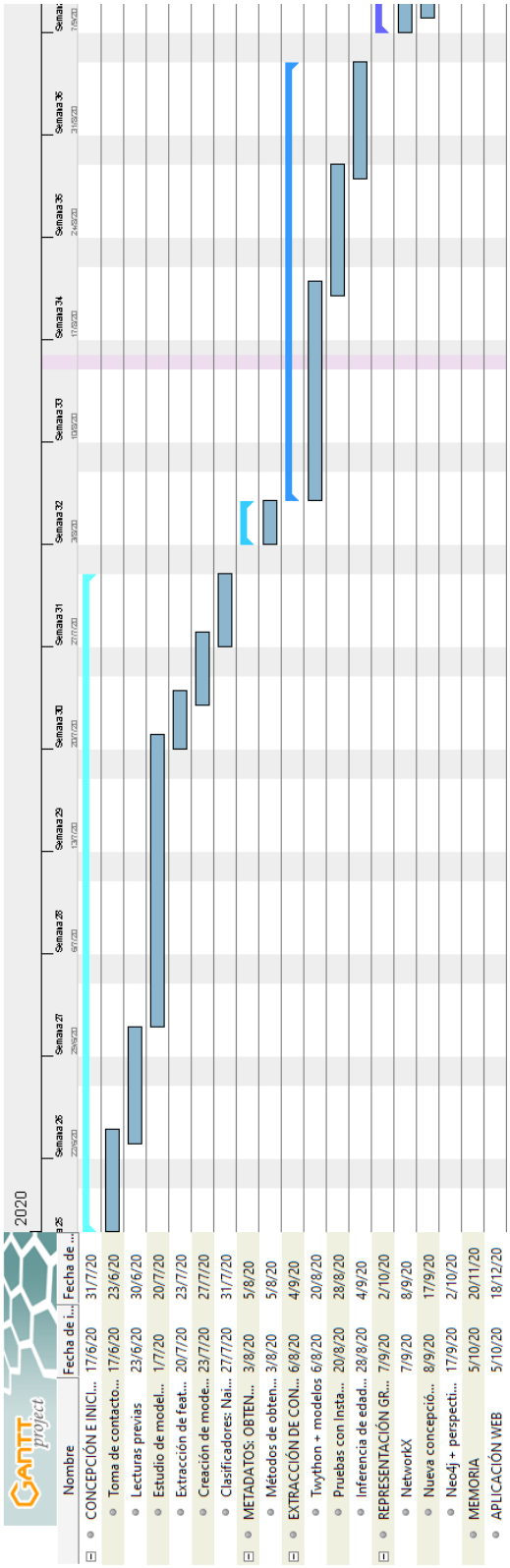

Figura A.2: Distribución hasta septiembre

Figura A.3: Distribución hasta diciembre