



TRABALLO FIN DE GRAO
GRAO EN ENXEÑARÍA INFORMÁTICA
MENCIÓN EN COMPUTACIÓN



Perfilado automático de usuarios en corpus sociales sobre el movimiento Black Lives Matter

Estudiante: Nicolás Míguez García

Dirección: Patricia Martín Rodilla

David Otero Freijeiro

A Coruña, octubre de 2023.

A mi familia

Agradecimientos

 Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Resumen

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Abstract

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Palabras clave:

- Perfilado automático
- Redes Sociales
- Archivos sociales
- Black Lives Matter
- Aprendizaje automático
- Aplicación web
- Procesado de lenguaje natural
- Aprendizaje profundo

Keywords:

- First itemtext
- Second itemtext
- Last itemtext
- First itemtext
- Second itemtext
- Last itemtext
- First itemtext

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En este capítulo se expone el contexto, motivación y objetivos principales del proyecto así como la estructura de la memoria.

1.1 Motivación

EL surgimiento de las redes sociales en la década de los 2000, ha tenido un gran impacto el modo de vida de la población general. Estas se han convertido en el medio social preferido para las comunicaciones personales, la publicación de fotos, opiniones, así como la difusión de información y el debate acerca de asuntos de actualidad. De hecho el crecimiento de estas redes ha propiciado el incremento de temas de debate y actualidad que se dan en la sociedad. Estos abarcan todas las temáticas posibles: política, cultural, moda, ética, ocio...

Por este motivo numerosos investigadores se han dedicado a crear y estudiar archivos nativos digitales desde redes sociales ([?] y [?]), de forma que estos puedan ser reusados de manera posterior por otros investigadores para estudiar de manera posterior determinados procesos sociales.

Un ejemplo de proceso social del que se ha tiene referencia a través de un archivo social es el movimiento [Black Lives Matter \(#BLM\)](#). [Black Lives Matter](#) ("las vidas negras importan" en español) es un movimiento descentralizado surgido en el año 2013 en Estados Unidos en respuesta a la violencia policial dirigida hacia las personas negras. Este busca acabar con la represión sistémica y el racismo estructural que afectan a las comunidades negras. Desde entonces se viene utilizando el lema para protestar contra los símbolos racistas contra la comunidad negra. No obstante, en mayo de 2020 fue cuando alcanzó mayor repercusión debido al asesinato de George Floyd. A partir de entonces, se originaron respuestas en forma de manifestaciones y protestas pacíficas, que también se materializaron en las redes sociales en forma de debates.

Gracias al trabajo de [?], que reunieron unas colecciones de referencia, en inglés¹ y español², acerca de las protestas causadas debido al movimiento, ahora estas se pueden usar para estudiar dicho movimiento en forma de archivo social. Este archivo tiene un año de actividad desde la muerte de George Floyd, e incluye más de 260.000 posts de 90.000 usuarios recogidos de una red social llamada Reddit³, que compartieron mensajes y contenido acerca de #BLM.

Estas colecciones de referencia son muy útiles a la hora de observar las diversas opiniones de los usuarios acerca de este movimiento. Pudiendo llegar a realizar estudios sobre la posición mayoritaria de la gente con respecto al mismo o los argumentos que tienen mayor peso a la hora de apoyarlo. Sin embargo, un aspecto fundamental acerca de la colección del que no se tiene información explícita está relacionado con: ¿Quién publicaba en la colección? No sabemos nada acerca del perfil demográfico, generacional o rasgos psicológicos de los usuarios de la colección. Conociendo esta información podríamos indagar de manera más profunda en el estudio de la misma.

Se conoce como perfilado automático de usuarios al procesamiento automático de información, como publicaciones y textos procedentes de redes sociales, mediante técnicas de procesado de lenguaje natural e inteligencia artificial con el objetivo de extraer características, preferencias o comportamientos acerca de estos. En el pasado, diversos autores ya demostraron que es posible la extracción en base al estilo de escritura y lenguaje utilizado de ciertas características acerca de los autores de textos como pueden ser edad, género o rasgos de personalidad ([?], [?]). No obstante, el incremento por interés en el perfilado automático de usuarios en redes sociales se vio patente en los últimos años en la organización de una serie de competiciones y trabajos con el objetivo de mejorar los algoritmos existentes de perfilado automático en distintos idiomas ([?], [?], [?]).

1.2 Objetivos

El propósito de este trabajo es por tanto estudiar las anteriores colecciones desde la perspectiva del usuario. Esto es: se pretende obtener información de las distintas personas que interactuaban y debatían sobre #BLM en redes sociales. De forma que se pueda estudiar las distintas opiniones de los usuarios en función de sus características demográficas, ideológicas o personalidad de los mismos.

En concreto, este trabajo se centrará completamente en la colección de los usuarios que escribían en idioma español. Para ello, será necesario revisar y analizar las técnicas del estado del arte en materia de perfilado en este idioma, con la intención de replicarlas para el estudio

¹ <https://www.dc.fi.udc.es/~david/heritage/>

² <https://www.dc.fi.udc.es/~david/hdh2021/>

³ <https://www.reddit.com/>

de nuestra colección.

Como resultado, a continuación se exponen en detalle los objetivos identificados en este proyecto:

- Analizar y evaluar los algoritmos y modelos existentes del estado del arte en el ámbito del perfilado automático de usuarios, especialmente en materia de edad y género, en redes sociales como Reddit o Twitter.
- Seleccionar algunos de los algoritmos analizados anteriormente con el objetivo de aplicarlos a las colecciones de #BLM.
- Estudiar los resultados obtenidos de aplicar estos algoritmos mediante la creación de una aplicación web a modo de dashboard que muestre la distribución de usuarios, que publicaban en las colecciones, en función de sus características demográficas.

1.3 Estructura de la memoria

A continuación, se muestra la organización de la estructura de la memoria. Esta está dividida en los siguientes X capítulos:

- **Introducción:** capítulo actual donde se enuncian los aspectos principales del proyecto como motivación y objetivos del mismo. Así mismo, también se expone la estructura global de la memoria.
- **Fundamentos:** este capítulo corresponde a la fase del trabajo donde se evalúan y analizan el estado del arte, en materia de los algoritmos de perfilado automático que se usarán posteriormente en la colección de #BLM.
- **Tecnologías y herramientas utilizadas:** este capítulo presenta las tecnologías y herramientas que puedan ser relevantes mencionar, empleadas en alguna de las fases del proyecto.
- **Análisis colección #BLM:** aquí se muestran los resultados de aplicar los algoritmos seleccionado a la colección de BLM. Estos se muestra a través de un dashboard creado mediante una aplicación web.
- **Metodología y gestión del proyecto:** en él se explica el enfoque metodológico empleado para la realización del proyecto, al igual que la planificación seguida.
- **Desarrollo:** aquí se detalla el análisis de requisitos, el diseño de la arquitectura utilizada y el desarrollo seguido para la construcción de la aplicación web.

- **Conclusiones y trabajo futuro:** en este capítulo se exponen las conclusiones que se derivan del trabajo realizado, así como posibles líneas de trabajo que puedan ser interesante continuar en el futuro.

Capítulo 2

Fundamentos

En este capítulo se expondrán y analizarán brevemente los trabajos previos en la materia de perfilado automático de usuarios en base a texto, que se han tenido en cuenta para la elaboración de las soluciones escogidas.

2.1 Estado del arte

Desde los primeros años del inicio de las redes sociales, el interés por la extracción automática de características demográficas latentes de usuarios se vio patente en una serie de trabajos tempranos. En 2007, [?] demostró como el estilo de escritura y otros aspectos socio-lingüísticos observados en textos procedentes de blogs en idioma inglés se ven afectados por características demográficas como la edad y género de los autores de los mismos. En este otro trabajo del 2006 [?], se obtuvieron buenos resultados en la clasificación de género y edad (precisión mayor al 75% para ambos) usando para la combinación de características basadas en estilo y basadas en contenido, en un corpus compuesto por textos provenientes de blogs en inglés. También se tiene registro de varios trabajos que tratan el tema a partir de la creación de corpus en otros idiomas como [?] en roman urdu.

Sin embargo, en los últimos años destacan especialmente, en materia de perfilado automático y *authorship analysis* en general, las competiciones organizadas por el PAN¹. Estas consisten en una serie de eventos científicos y tareas compartidas sobre texto digital forense y estilometría. Estas tareas se realizan anualmente y en la mayoría de ediciones se pueden encontrar tareas de perfilado de usuarios ([?], [?], [?]).

Por otro lado, desde 2019 se empezaron a organizar la competición análoga al PAN en idioma español: el IberLEF². Esta se enmarca dentro de la Sociedad Española para el Procesamiento del Lenguaje Natural (SEPLN) como una serie de tareas para promover la investigación de

¹ <https://pan.webis.de/>

² <http://sepln2023.sepln.org/iberlef/>

tareas de análisis, procesamiento y generación de textos en las lenguas ibéricas.

Sin embargo, a pesar de estas iniciativas recientes hay que tener en cuenta que los esfuerzos realizados en materia de perfilado automático de usuarios, están centrados mayoritariamente en el idioma inglés. Debido a la naturaleza multilingüe de muchos de los modelos y algoritmos, estos pueden ser adaptados a otros idiomas como el español sin demasiada dificultad. Sin embargo, otro tipo de recursos como pueden ser los conjuntos de entrenamiento para estos modelos son únicos para un idioma por lo tanto en este aspecto se nota en mayor medida la escasez con respecto al idioma inglés.

De este modo, pese al gran interés de este tipo de algoritmos en el idioma español el.....

2.2 Conjuntos de datos

Como nuestro objetivo consiste en conocer las características demográficas de los usuarios de la colección de #BLM es necesario disponer de un corpus o conjunto de datos para entrenar los modelos perfilado automático.

Lo ideal en este caso sería crear un corpus personalizado, a modo de conjunto de entrenamiento, lo más similar posible a la colección, es decir, preferiblemente a partir de textos del mismo foro de Reddit o de otro parecido al que se extrajo esta. Crear un corpus de un tamaño aceptable con estas características, supondría un proceso bastante difícil y costoso en tiempo, ya que la API de Reddit no permite obtener edad y género de manera sencilla. Sería necesario etiquetar manualmente las características demográficas de los usuarios: o bien preguntándoles directamente o bien realizando una labor de investigación sobre los mismos para averiguarlas.

Por este motivo, se decidió usar un corpus con usuarios etiquetados ya creado. Existen varias opciones en idioma español en este sentido. En 2014, [?] crearon un corpus en español compuesto principalmente por texto formal (esto es con pocas palabras que no pertenecen al "diccionario", abreviaturas, vulgarismos...) a partir de texto extraídos de la web. No obstante, el lenguaje usado en la colección de #BLM es más bien de tipo informal, es decir, contiene bastantes vulgarismos, extranjerismos y expresiones de la jerga de las redes sociales. Por esta razón, fue que se descartó el corpus anterior en busca de otro procedente de redes sociales más informales como Reddit o Twitter. En este sentido, los conjuntos de datos creados por los organizadores de la tareas del PAN son las únicas opciones viables que se encontraron, que incluían ambos género y edad. Concretamente usaremos los particiones de los *datasets* correspondientes a textos de Twitter en español de las competiciones de los años 2015³ y 2016⁴ ([?] y [?] respectivamente).

³ <https://zenodo.org/record/3745945>

⁴ <https://zenodo.org/record/3745963>

2.2.1 PAN author profiling 2015

El conjunto de 2015, estaba compuesto por textos procedentes de Twitter completamente y estaba formado por una partición de entrenamiento y otra de test. Además de edad y género los autores de los texto de este *dataset* estaban etiquetados con valores auto evaluados para los rasgos de personalidad, definidos mediante el Modelo de los 5 Grandes ([?]).

En cuanto al formato de los *datasets*, en el conjunto de 2015 cada autor aparece representado con un fichero .xml donde dentro del tag raíz author se encontraban situados los *tweets*. Este contiene exactamente 100 *tweets* distintos cada uno como contenido de un tag document hijo del tag raíz author. Por cada autor hay exactamente 100 tags.

Para conocer las etiquetas de género, edad, y personalidad existe un único archivo en texto plano en el que aparece por cada autor una línea donde se asocia sus valores demográficos y de personalidad. Como podemos ver en la figura ?? el contenido del archivo está a modo de archivo .csv donde el separador es la cadena ”::”. Las columnas del archivo se corresponderían con el identificador del autor, rango de edad, género y valores de personalidad como se indica [?].

1	e4ce3b69-c0c7-47e6-92be-85e6a16c87d8:::F:::25-34:::0.0:::0.3:::0.2:::0.2:::0.2
2	8192b89d-8335-45b2-a4fa-10f7b54af66b:::M:::50-XX:::0.1:::0.1:::0.1:::0.4:::0.0
3	0aa1b46b-69b1-4e38-ad93-551ca756fccb:::M:::25-34:::0.1:::0.2:::0.3:::0.3:::0.1
4	067b0dab-4394-46ed-a3c7-3a47581eb68b:::F:::25-34:::0.3:::0.2:::0.1:::0.5:::0.1
5	cab5fc4b-3823-4743-b698-339d23d6b47b:::M:::35-49:::0.3:::0.3:::0.1:::0.3:::0.1
6	8a271d2b-387a-470e-aaa6-f1bfd8fb4d7c:::M:::18-24:::0.2:::0.4:::0.1:::0.1:::0.4
7	3ea0fd5e-d7aa-41b3-bd4a-08ecab713081:::M:::50-XX:::0.3:::-0.1:::-0.1:::0.0:::-0.1
8	9823ea62-a1d9-49ae-81d4-f9f640d991b2:::F:::25-34:::0.4:::-0.2:::0.2:::0.4:::0.4
9	1b5188b2-904f-4ea7-8116-e4616477d60b:::F:::25-34:::0.2:::0.0:::-0.1:::0.5:::0.1

Figura 2.1: Formato archivo truth.txt *dataset* 2015

2.2.2 PAN author profiling 2016

Por otra parte, el conjunto de 2016 incluía una partición de entrenamiento formada por textos sacados de usuarios de Twitter y una partición de test formada por texto sacados de Blogs. Estos últimos utilizan un lenguaje más formal, habitualmente conteniendo poemas, canciones o fragmentos de texto no escritos por los usuarios etiquetados que podían afectar negativamente al rendimiento del algoritmo de perfilado. Por estas razones, se decidió utilizar únicamente la partición de entrenamiento para entrenar nuestro clasificador.

El formato de este corpus es bastante similar al del 2015: cada autor aparece representado en un archivo .xml nombrado con un identificador único a partir del cual se puede sacar sus datos demográficos en un archivo truht.txt.

La diferencia con el de 2015, está en que en vez de aparecer los tweets directamente en el .xml, aparecen las URLs a esos tweets. Además en vez haber exactamente 100 *tweets* por

autor, en la mayoría hay hasta 1000. Para conseguir los textos correspondientes a los *tweets* el PAN disponía de un programa⁵ que los descargaba y escribía directamente el texto de los mismos en los .xml. Sin embargo, desde 2020 la API de Twitter cambió y este programa dejó de funcionar⁶.

Por este motivo, se decidió crear un web scraper que fuera leyendo las direcciones URL de los .xml que se correspondían con los tweets de los usuarios y descargara el texto de estos sin necesidad de usar la API de Twitter. Sin embargo, este proceso era mucho más lento que el uso del programa antes mencionado (1000 *Tweets*/minuto frente a aproximadamente 30 *Tweets*/minuto). Esto hizo que decidiera ejecutar de forma *serverless* este programa diariamente en forma de notebook en Kaggle⁷, esta web permite programar ejecuciones de jupyter notebooks de forma asíncrona que pueden durar hasta 12 horas siendo 10 el número máximo de notebooks ejecutándose concurrentemente. De esta forma en un par de días ya tenía descargada toda la colección, sin contar los tweets y usuarios eliminados que eran bastantes. En la figura ?? se puede ver una captura de los logs resultantes de realizar una ejecución de 12 horas del *scraper* para descargar los *tweets* del *dataset*.

The screenshot shows a Kaggle notebook interface. At the top, there's a profile icon, the notebook title 'ScrapeTweets', the author 'NICOLÁS MIGUÉZ GARCÍA - 23D AGO - 110 VIEWS - PRIVATE', and a 'Logs' tab which is currently selected. Below the tabs, there's a message: 'Your notebook was stopped because it exceeded the max allowed execution duration. Exit code: 137'. The main area displays a table of log messages:

Time	#	Log Message
30178.1s	1	Scraped 30 / 100 tweets
30179.0s	2	Scraped 32 / 100 tweets
30179.4s	3	Scraped 31 / 100 tweets
30179.4s	4	Executed in 12.689604759216309 seconds

Figura 2.2: Logs resultado de una ejecución del notebook a modo de *scraper* para la obtención de los *tweets* del *dataset* de 2016

⁵ <https://github.com/pan-webis-de/pan-code/tree/master/clef16/author-profiling/twitter-downloader>

⁶ Enlace al issue de github donde lo indican: <https://github.com/pan-webis-de/pan-code/issues/3>

⁷ <https://www.kaggle.com/>

2.2.3 Distribución de usuarios de los *dataset*

En la tabla ?? podemos ver la distribución de autores en función de edad y género de ambos conjuntos. Como se puede observar los *datasets* están equilibrados en cuestión de género, mientras que están bastante más desequilibrados en cuestión de edad.

Vale la pena mencionar que aunque en el *dataset* de 2016, originalmente había 250 de Training, al descargar los textos correspondientes a estos muchos de ellos no estaban disponibles (puesto que los *tweets* eran se había publicado originalmente en el año 2013), pues se habían eliminado su cuenta de Twitter. Consecuentemente, descartando estos usuarios quedan un total de 222.

Categoría	PAN-AP 2015		PAN-AP 2016
	Training	Test	Training
18-24	22	18	11
25-34	46	44	54
35-49	22	18	116
+50	10	8	41
Hombres	50	44	111
Mujeres	50	44	111
Total	100	88	222

Tabla 2.1: Distribución de usuarios en función de edad y género en los conjuntos de entrenamiento utilizados.

2.3 Búsqueda y comparación de algoritmos

En esta sección, se detalla la fase del proyecto consistente en la investigación sobre los algoritmos o modelos existentes de clasificación de texto para tareas de perfilado automático de usuarios.

El objetivo principal de esta fase era el análisis del estado del arte en este campo con la finalidad de escoger y replicar los algoritmos, de perfilado de usuario, más robustos para nuestro trabajo. En consecuencia, vale la pena resaltar que la intención no era realizar un modelo original desde cero sino utilizar uno ya implementado y del que se tenga conocimiento que reporta buenos resultados.

Para ello, se orientó la búsqueda hacia las competiciones ya mencionadas en el apartado ?? como PAN ([?]) o IberLef ([?]). Esto tiene dos motivos: primero que seleccionando los

algoritmos de entre los participantes en una de estas competiciones, contamos con la información previa de la bondad de los algoritmos con los otros competidores, pudiendo así seleccionar los que mejor se comportaron en las mismas; por otra parte, estas competiciones al estar de alguna forma abiertas al público es más sencillo encontrar de forma pública las implementaciones de los algoritmos participantes en repositorios como Github⁸.

En la tabla ?? prueba se muestra una pequeña comparativa de los algoritmos del estado del arte con implementación disponible en Github. En ella se muestran en cada columna: la referencia a las *working notes* del equipo participante de la tarea, la competición para la que se desarrolló, la posición final en la que quedó el equipo participante (teniendo en cuenta perfilado en idioma español solamente), el modelo de aprendizaje automático utilizado y un resumen muy general de las características extraídas para la clasificación del texto.

Como se puede ver en la tabla muchos de estos algoritmos usan modelos y características bastante similares entre ellos. En consecuencia, solo se decidieron replicar tres de ellos ([?], [?] y [?]).

Equipo	Competición	Posición	Modelo	Características
Grivas [?]	PAN 2015 [?]	3º	SVM	Tfidf n-grams stylistic features
Modaresi [?]	PAN 2016 [?]	2º	LR	Tfidf n-grams stylistic features
Daneshvar [?]	PAN 2018 [?]	1º	SVM	Tfidf ngrams w LSA
Bacciu [?]	PAN 2019 [?]	3º	SVM	Tfidf n-grams w LSA
Baseline	PAN 2020 [?]		LR	Tfidf n-grams
LosCalis [?]	IberLef 2022 [?]	1º	MLP	Embeddings (BERT + RoBERTa)
Holgado [?]	IberLef 2022 [?]	5º	EM	n-grams + embeddings + stylistic features

Tabla 2.2: Comparación de algoritmos del estado del arte en perfilado de usuarios de los cuales se encontró la implementación disponible en Github.

2.3.1 Primera aproximación

En primer lugar, tenemos al ganador de la competición del IberLef 2022: "PoliticEs 2022: Perfilado de usuarios para ideología política". Esta competición consistía en la predicción de

⁸ <https://github.com/>

género, profesión e ideología política binaria (izquierda y derecha) y multiclasificación (izquierda, centro-izquierda, centro-derecha y derecha) de usuarios de Twitter sobre un corpus de *tweets* procedentes de periodistas, políticos y españoles.

En esta competición [?], la mayoría de aproximaciones usadas por los participantes eran modelos basados en transformers [?]. Estos consistían en modelos de lenguaje pre-entrenados (BETO [?], MarIA [?], BERT multilingüe, ALBERT...) afinados con el corpus de la competición, con el objetivo de extraer características del texto a nivel de documento. Este tipo de arquitecturas, basadas en transformers, son la gran novedad con respecto a los enfoques utilizados en competiciones de años anteriores del PAN.

El enfoque utilizado por LosCalis [?] por tanto, combinaba el modelo BERT [?] pre-entrenado en idioma español, conocido como BETO [?] junto con el modelo RoBERTa pre-entrenado en español, llamado MarIA [?]. Este emplea ambas arquitecturas para extracción de características a nivel de documento y en base a estas se entrena un perceptrón multicapa para decodificar las etiquetas de los usuarios. En la figura ?? se muestra la arquitectura del modelo utilizado.

Preprocesado

Por una parte está el preprocesado realizado a la hora de la creación del corpus por los organizadores de la competición. Este consistió en descartar *retweets*, *tweets* donde se parafrasean titulares de periódicos o *tweets* en otros idiomas distintos del español. Además de esto, se anonimizó la colección sustituyendo las menciones a usuarios por el token "@user". Por otro lado, el preprocesado efectuado por LosCalis se limitó a la agrupación de los *tweets* de un mismo autor en bloques de máximo 512 tokens tras la tokenización con BERT y RoBERTa. Esto es así debido a que los modelos basados en BERT no aceptan entradas de mayor tamaño. En nuestro caso, se realizó un preprocesado parecido con la excepción de que se mantuvieron todos los *tweets* de la colección aunque contuvieran fragmentos en otros idiomas o parafrasearan titulares de noticias o blogs. Por un lado, esto se hizo debido a que nuestros *datasets* cuentan con un volumen mucho menor de textos en relación a los corpus de la competición del IberLef, por esa razón decidimos no realizar esa limpieza para mantener un corpus lo más grande posible.

Por otro lado, también se optó por sustituir las *urls* encontradas por el token '<URL>', debido a que en el corpus de la competición no se incluían estas tampoco. Además, se optó por sustituir los emoticonos encontrados en el texto por el token '<EMOTICON>' y los emojis por su descripción sacada de la librería emoji⁹.

⁹ <https://pypi.org/project/emoji/>

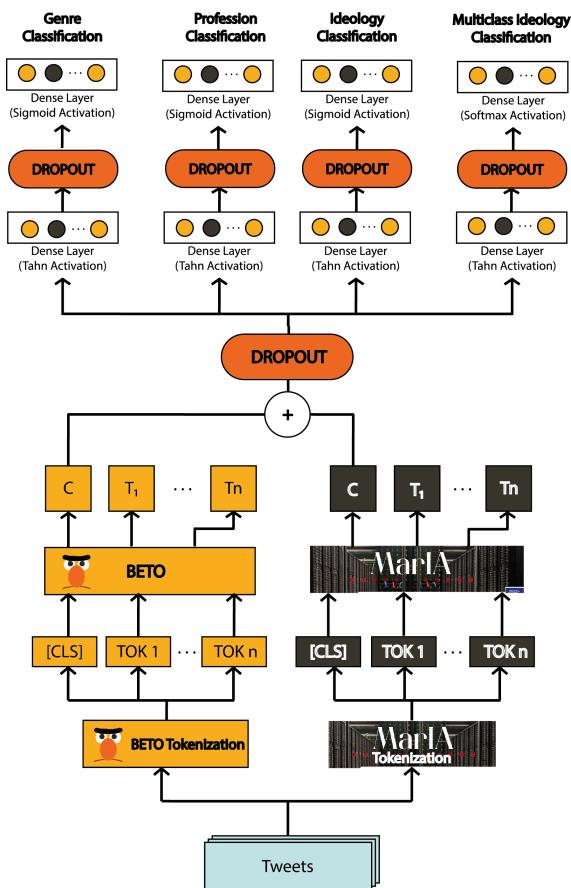


Figura 2.3: Arquitectura del modelo ganador de la competición de Political Author Profiling del IberLef 2022, utilizado por LosCalis [?].

Modelo

En nuestro caso se replicó el mismo modelo usado por los participantes de la tarea, con las adaptaciones debidas para en las salidas de la red neuronal. En el caso de LosCalis, tenían tres salidas binarias (género, profesión, ideología binaria) y una multiclasificación (ideología multiclasificación). En nuestro problema, se tienen únicamente dos salidas: género y edad. En consecuencia se utilizó la arquitectura que se puede ver en la figura ???. En ella se muestra que se mantienen las mismas capas y funciones de activación que en la arquitectura original con la diferencia de que en vez de cuatro bloques separados de decodificación de las etiquetas, ahora tenemos únicamente dos.

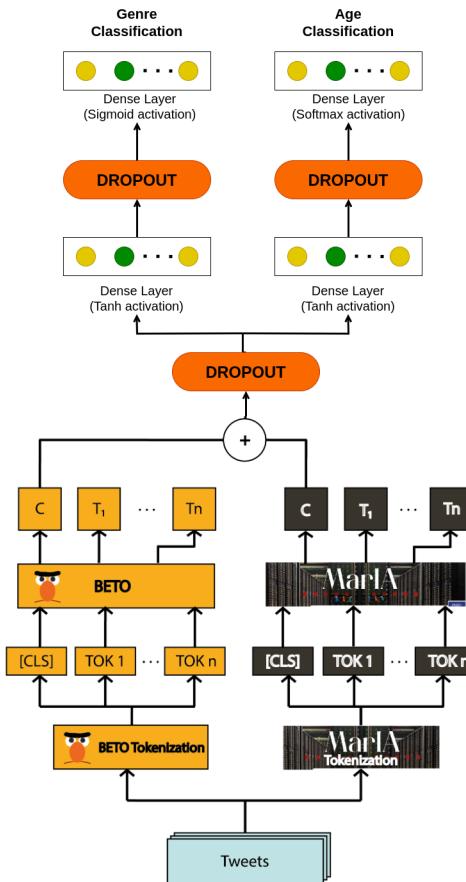


Figura 2.4: Arquitectura basada en transformers adaptada de la aproximación de [?].

Experimentos realizados

Se realizaron distintas pruebas con los conjuntos de datos expuestos en la sección ???. Para estas se utilizaron la misma configuración de hiper-parámetros propuesta por [?], que se

puede ver en la tabla ??.

Parámetro	Valor
Train batch size	64
Predict batch size	8
Learning rate	3e-5
Training epochs	5
Max sequence lenght	512
Dropout	0.15
Optimizer	Adam
Dense layer units	768

Tabla 2.3: Hiperparámetros utilizados en los experimentos realizados, propuestos por [?]

En la tabla ?? se pueden ver los experimentos realizados así como los resultados obtenidos en cada uno de ellos en términos de precisión para género y edad.

Entrenamiento	Test	Género	Edad
Training 2015	Test 2015	0.8863	0.6250
2015 (Train+Test)	Training 2016	0.4865	0.2703
Training 2016	2015 (Train+Test)	0.4468	0.3191
Training 2016	Test 2016	0.4545	0.4181

Tabla 2.4: Precisión para género y edad en distintos experimentos realizados utilizando el enfoque propuesto por [?] con las particiones de entrenamiento y test de las competiciones de PAN de 2015 y 2016.

Lo primero que llama la atención de estos resultados es la diferencia de rendimiento obtenida usando el *dataset* de 2015, a modo de réplica de la competición del PAN [?], con el resto de experimentos realizados.

En el primer caso los resultados obtenidos no se alejan demasiado de los conseguidos por el equipo ganador de la misma: 0.9659 y 0.7955, para género y edad respectivamente.

Sin embargo, cuando se trata de aumentar estos datos para obtener mejores resultados, por ejemplo usando el *dataset* completo de 2015 (entrenamiento y test) para entrenar el modelo y se prueban a predecir los autores del corpus de entrenamiento del 2016 se puede ver como los resultados obtenidos empeoran considerablemente, hasta el punto de ser peores que un

clasificador aleatorio. Esto se repite al entrenar con la partición de Training de 2016 y testear con la de 2015 completa o con la de Test de 2016 (competición PAN 2016).

Para solventar este problema, lo primero que se probó fue separar una parte del conjunto de entrenamiento para usarlo como conjunto de validación, con la finalidad de observar como se comportaba la función de pérdida o loss¹⁰, de la red.

La función de pérdida en aprendizaje automático mide la discrepancia entre la clasificación de las predicciones de la red y los datos reales. En las redes neuronales, el objetivo del entrenamiento consiste en la minimización del valor de esta función en cada ciclo mediante del ajuste del valor de los pesos de las conexiones de la red, por medio del algoritmo de retro-propagación del error. Como se explica en [?], existen diversas alternativas a la hora de escoger una función de pérdida, sin embargo, para problemas de clasificación es bastante común el uso de la entropía cruzada (pérdida logarítmica), por lo cual es la elegida para nuestro algoritmo.

De esta forma viendo la evolución de la función de pérdida durante el entrenamiento, se podría saber si el sistema estaba sobre-entrenado o no. Pues bien, los resultados de esta prueba se pueden ver en la figura ??.

Como se puede ver a la hora de entrenar el modelo con los datos de 2015 el loss del conjunto de

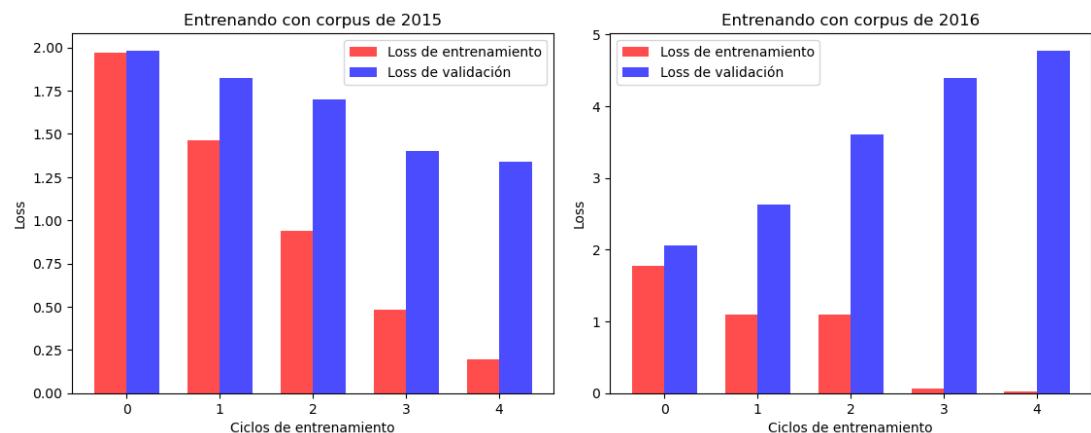


Figura 2.5: Evolución del loss de entrenamiento y validación del modelo propuesto por LosCalis al entrenar con los *datasets* de 2015 y 2016.

validación disminuía al mismo tiempo que el loss del conjunto de entrenamiento; sin embargo, a la hora de entrenar usando el corpus de entrenamiento de 2016 pasaba justo lo contrario, el loss de validación aumentaba cada ciclo mientras el de entrenamiento disminuía (signo de que el modelo se está sobre-ajustando a los datos de entrenamiento y no consigue generalizar).

Tras descubrir esto, lo primero que se me ocurrió fue investigar un poco los dos conjuntos y compararlos para descubrir si existía alguna diferencia entre ellos. Lo que se descubrió fue que probablemente en el *dataset* de 2015, tanto en el conjunto de entrenamiento como en el de

¹⁰

test existían autores repetidos, de modo que el modelo obtenía buenos resultados debido a que el modelo se aprendía las características textuales de usuarios concretos y no a distinguir la edad o género de una persona aleatoria.

Por otro lado, la diferencia de rendimiento entre el modelo adaptado a nuestro problema y el propuesto por [?] se especula que se deba probablemente a el mayor tamaño del corpus usado para entrenar en la competición del IberLef (alrededor de 400.000 *tweets* de más de 700 usuarios distintos, frente a 222 usuarios con aproximadamente 800 tweets cada uno) unido a un vocabulario más formal usado por políticos y periodistas españoles junto a unas temáticas mucho más menos variadas, comparado con el vocabulario del usuario promedio de twitter lleno de expresiones de jerga, abreviaturas y disparidad de temáticas tratadas.

Debido a estos malos resultados obtenidos y a la falta de un corpus de mayor tamaño necesario para entrenar un modelo complejo basado en aprendizaje profundo, se decidió orientar el trabajo hacia modelos más sencillos como los expuestos a continuación.

2.3.2 Segunda aproximación

Por otro lado el segundo algoritmo de perfilado probado es el correspondiente a [?]. Este obtuvo la tercera posición, en general, en la tarea de author profiling de PAN 2015 ([?]). Los **datasets** usados en la competición son los explicados en la subsección ??.

Preprocesado

El preprocesado realizado es distinto dependiendo del tipo de características extraídas. Así los autores de este trabajo separan por un lado características estructurales y características estilométricas.

Las primeras incluyen el numero de menciones, *hashtags* y URLs que contienen los *tweets*. Las segundas, se refieren a la longitud del *tweet* en caracteres y tf-idf n-grams.

Para las estructurales no se realizó ningún preprocesado del texto a parte de concatenar todos los tweets de un autor. Por otro lado, para las estilométricas se eliminaron cualquier tag HTML encontrado, URLs, menciones así como los caracteres '#' de los *hashtags*.

Modelo

Para la clasificación del género, se usó una **SVM** con kernel rbf, y se usaron como características únicamente los tf-idf n-grams. En cuanto a la edad, se usó una **SVM** con kernel linear a la que se le pasaron como características: tf-idf ngrams, longitud del *tweet*, número de URLs, menciones y *hashtags*. Además, se compensó el desbalanceo de los distintos grupos de edad mediante pesos inversamente proporcionales a la frecuencia de las clases.

Experimentos y resultados

En la tabla ?? se pueden observar los resultados de los experimentos realizados en esta aproximación. Las métrica elegida tanto para género como edad es la precisión, debido a que es la que se usaba en las competiciones del PAN 2015 y 2016.

N-gram	Entrenamiento	Test	Género	Edad
Carácter [3, 3]	Training 2015	Test 2015	0.9091	0.7159
Carácter [3, 5]	Training 2015	Test 2015	0.9205	0.6932
Carácter [3, 3]	Training 2015	Training 2016	0.5766	0.3423
Carácter [3, 3]	Training 2015	Training 2016	0.5856	0.3018
Carácter [3, 3]	2015 (Train+Test)	Training 2016	0.5405	0.3468
Carácter [3, 5]	2015 (Train+Test)	Training 2016	0.5586	0.2972
Carácter [3, 5]	Training 2016	2015 (Train+Test)	0.5638	0.4627
Carácter [3, 5]	Training 2016	Test 2016	0.5091	0.4545
Carácter [3, 3]	Training 2016	*10-Fold CV	0.6712	0.5045
Carácter [3, 5]	Training 2016	*10-Fold CV	0.6757	0.5045
Carácter [5, 8]	Training 2016	*10-Fold CV	0.6667	0.5045
Palabra [1, 3]	Training 2016	*10-Fold CV	0.6937	0.4910
Carácter [3, 3]	2015+2016 (Train)	*10-Fold CV	0.7024	0.6073
Carácter [3, 5]	2015+2016 (Train)	*10-Fold CV	0.7170	0.5707
Carácter [5, 8]	2015+2016 (Train)	*10-Fold CV	0.7170	0.6195
Palabra [1, 3]	2015+2016 (Train)	*10-Fold CV	0.6610	0.5390
Palabra [1, 1]	2015+2016 (Train)	*10-Fold CV	0.7122	0.5829

Tabla 2.5: Resultados en términos de precisión para género y edad de los experimentos realizados para la aproximación 2 (modelo propuesto por grivas).

Una vez más sucede que los resultados obtenidos de replicar la competición del PAN 2015, son mucho mejores que el resto de experimentos realizados, lo que afianza la teoría de que en ese *dataset* existen autores repetidos.

Por otro lado, esta aproximación reporta mejores resultados tanto en género como en edad en el resto de experimentos realizados. Por ejemplo al entrenar con el dataset de 2016 y hacer test con 2015 el perfilador ya es ligeramente mejor que un clasificador aleatorio: 56% y 46% de precisión para género y edad respectivamente. Lo que refuerza la idea de que en caso

de disponer de un corpus de tamaño relativamente pequeño los modelos tradicionales más sencillos son preferibles a los grandes modelos de lenguaje basado en aprendizaje profundo y transformers que están de moda actualmente.

Los resultados de replicar la competición de PAN 2016 (entrenamiento 2016 y test 2016) muestran como este modelo no generaliza bien a otros géneros a parte de Twitter como son los textos de Blogs, utilizados en el *dataset* de Test de 2016.

Para finalizar, en las últimas filas de la tabla se investigó acerca del tipo de características basadas en n-grams que mejor funcionaban en estos *datasets*. Para ello se utilizó un 10-fold Cross-Validation con el dataset de training de 2016 y el dataset de training de 2016 sumado al de 2015 completo (Training y Test). El cross-validation se utilizó en parte para evitar el sesgo de que pueda haber autores repetidos en los datos del 2015 y porque en 2016 la partición de Test contenía textos de un género distinto (textos de Blogs) que la de entrenamiento. Los resultados que se sacan de estos experimentos son que los n-grams basados en secuencias de 1-3 palabras son las que mayor precisión alcanzan para la predicción de género. Mientras que, los n-grams basados en secuencias de 5-8 caracteres son las que mejores resultados obtienen para edad. No obstante, entre los rangos probados no hay una diferencia significativa de rendimiento en la clasificación.

2.3.3 Tercera aproximación

El último perfilador analizado es el correspondiente al trabajo de [?]. Este fue propuesto para la competición del PAN 2016. En ella los participantes debían realizar un modelo que predijese el género y edad de autores de textos sacados de Blogs¹¹ habiendo entrenado el mismo mediante textos sacados de usuarios de Twitter (?). Por lo tanto, ese algoritmo debe ser lo más independiente del género de escritura posible, lo cual es beneficioso para nuestro problema, ya que entrenamos con textos sacados de una red social como Twitter para predecir usuarios de Reddit.

En esta competición los resultados fueron bastante bajos comparados con las anteriores: el autor de este modelo ([?]), el cual quedó en segunda posición en idioma español, obtuvo 0.6964 y 0.5179 para género y edad respectivamente.

Preprocesado

El procesado realizado en este algoritmo, tenía la intención de eliminar la información específica de género presente en los textos. De se usaron distintas operaciones de preprocesado en función de las características extraídas del texto. A continuación se enumeran todas las operaciones de preprocesado que se realizan para el perfilado:

¹¹ <https://www.blogger.com/about/>

- p1: Pasar el texto a minúsculas.
- p2: Filtrar cualquier URL encontrada.
- p3: Eliminar menciones en las que aparezca un nombre de usuario del tipo '@username'.
- p4: Eliminar todos los Hashtags del texto.
- p5: Eliminar los retweets encontrados (en teoría no existen en el *dataset*).
- p6: Eliminar cualquier carácter no latino encontrado.
- p7: Eliminar acentos latinos de palabras, para favorecer la precisión de características basadas en n-grams.
- p8: Eliminar caracteres no alfabéticos, como aquellos encontrados en emojis.
- p9: Eliminar stop-words basadas en una lista predefinida.

Además de estas operaciones se procedió a la típica operación de concatenar los texto procedentes de un mismo autor para la extracción de características.

Características extraídas

En la tabla ?? se especifican las operaciones que se realizaron para cada tipo de característica extraída. A continuación se explica en qué consisten estas:

Característica	Operaciones de preprocessado
Unigrams	p9 ◦ p8 ◦ p7 ◦ p6 ◦ p5 ◦ p4 ◦ p3 ◦ p2 ◦ p1
Bigrams	p8 ◦ p7 ◦ p6 ◦ p5 ◦ p4 ◦ p3 ◦ p2 ◦ p1
Promedio de errores de escritura	—
N-grams de caracteres	p2 ◦ p3 ◦ p4 ◦ p1
Características de puntuación	—

Tabla 2.6: Operaciones de preprocessado realizadas según la característica extraída del texto.

- **Unigrams** y **Bigrams**: consisten en secuencias de n-grams basadas en palabras de longitud uno y dos respectivamente.
- **Promedio de errores de escritura**: se determina un valor relativo para las palabras correctamente escritas y se va sumando, de modo que cuanto mayor sea esta suma en relación a la longitud del texto, menores errores de escritura habrá.

- **N-grams de caracteres:** en concreto se utilizan secuencias de longitud 4, de caracteres con límites, es decir, los caracteres de la secuencia deben pertenecer todos a la misma palabra.
- **Puntuación:** además se mide el número promedio de comas, puntos y marcas de exclamación del texto.

Para la predicción de la edad se usaron todas estas características, para el género se omitió la puntuación.

Modelo

En este algoritmo, se probó con diferentes técnicas como Gradient Boosting o Random Forests, sin embargo los resultados obtenidos por medio de Regresión Logística fueron superiores. Por ello se empleó este modelo.

Se usa la estrategia uno frente a todos para la clasificación multiclas de la edad. Ademñas se establece el hiperparámetro $C = 10^{-3}$.

Experimentos y resultados

En la tabla ?? se muestran los experimentos realizados en esta aproximación. Como en las anteriores la métrica utilizada es la precisión.

Dataset		Género		Edad	
Entrenamiento	Test	Acc	F1	Acc	F1
Training 2015	Test 2015	0.8977	0.8977	0.5795	0.5795
2015 (Train+Test)	Training 2016	0.6368	0.6368	0.3722	0.3722
Training 2016	2015 (Train+Test)	0.6968	0.6968	0.3830	0.3830
Training 2016	Test 2016	0.5818	0.5818	0.5090	0.5090
Training 2016	*10-Fold CV	0.7657	0.7657	0.5124	0.5124
2015+2016 (Train)	*10-Fold CV	0.8173	0.8173	0.6423	0.6423

Tabla 2.7: Resultados en términos de precisión para género y edad de los experimentos realizados para la aproximación 3 (modelo propuesto por modaresi).

En esta ocasión se puede ver como sigue el patrón de resultados muy buenos para el benchmark de la competición del PAN 2015: género alrededor de 90% de precisión, y para la edad en este caso los resultados son bastante peores que en con los otros dos modelos.

Sin embargo, esta aproximación mejora a las otras dos en cuanto al perfilado para géneros distintos (entrenamiento y test 2016), donde vemos que el género ya supera el 50% de precisión y la edad lo alcanza también.

En líneas generales, se puede observar que esta aproximación mejora significativamente a las otras dos en la predicción del género sobre todo, pues consigue una precisión media de 76% para el 10-fold Cross-Validation del *dataset* de 2016 y llega al 81% si añadimos el corpus del 2015, frente al 69% y 71% de la aproximación 2. En cuanto a la predicción de la edad no obstante, no se aprecia una mejora significativa con respecto a la segunda aproximación.

Capítulo 3

Tecnologías utilizadas

En este capítulo se presentan las herramientas tecnológicas utilizadas durante el desarrollo de este trabajo, al igual que la justificación de su elección sobre otras alternativas. En primer lugar, se presentan las herramientas tecnológicas empleadas para el desarrollo y entrenamiento de los modelos de perfilado automático seleccionados. Por otro lado, tenemos las herramientas utilizadas para la construcción de la aplicación web. Por último, se exponen las herramientas ayuda al desarrollo que se han utilizado transversalmente durante todo el proceso software.

3.1 Algoritmos de perfilado

En esta sección se muestran las herramientas empleadas para la creación y entrenamiento de los modelos de perfilado automático, así como para la extracción, carga y preprocesado de los conjuntos de datos de entrenamiento y test.

3.1.1 Python

Python¹ es un lenguaje de programación de alto nivel, interpretado y multiparadigma. Actualmente es uno de los lenguajes de programación más populares en diversos ámbitos como la ciencia y análisis de datos, inteligencia artificial y programación web. Algunas de las razones por las que es tan popular son: su sintaxis sencilla y legible, su baja curva de aprendizaje, pero sobre todo el gran soporte que tiene en cuanto a librerías populares como PyTorch, Pandas, Numpy, Tensorflow o Scikit-learn, ampliamente utilizadas en el ámbito de la ciencia de datos y el aprendizaje automático.

¹ <https://www.python.org/>

3.1.2 Scikit-learn

Scikit-learn² es la librería por excelencia en python para el desarrollo de modelos de aprendizaje automático diferentes de redes neuronales. Esta cuenta con una gran variedad de modelos para tareas de clasificación, regresión o agrupamiento como **SVM**, **LR**, árboles de decisión... Además, también dispone de módulos para el procesado de las características textuales, funciones de evaluación de los clasificadores y realización de validación cruzada. Como alternativa a esta tenemos a **Mlib**³ de Spark que permite realizar trabajos de aprendizaje automático de manera distribuida. Sin embargo, su uso sería un poco innecesario pues los modelos creados con Scikit-learn se pueden ejecutar en el equipo local sin problemas ya que no suponen una gran carga computacional. Se usaron las implementaciones de esta librería para los modelos como **SVM** y **LR**, además de las funcionalidades de preprocesado, validación cruzada y evaluación mencionadas antes.

3.1.3 Tensorflow, Keras y Transformers

Tensorflow⁴ es una librería de código abierto creada por Google para el desarrollo de modelos basados en redes neuronales. Tiene una gran capacidad de distribución y optimización, permitiendo la aceleración por hardware de sus modelos mediante el uso de varios núcleos de CPUs, GPUs y TPUs. Una gran ventaja de Tensorflow respecto a PyTorch (las dos librerías de aprendizaje profundo más populares) es su gran integración con Keras⁵.

Keras es un API de alto nivel construído por encima de Tensorflow que simplifica muchas de las operaciones habituales de Tensorflow. De esta forma, su facilidad de uso hace que sea ideal para principiantes en el aprendizaje profundo, pues es más fácil de aprender que las otras dos APIs de más bajo nivel.

Por otro lado, Transformers⁶ es otra biblioteca de código abierto desarrollada por Hugging Face, muy usada en el ámbito del **Natural Language Processing** y en visión artificial. Esta librería pone a disposición del público una gran variedad de modelos preentrenados de aprendizaje profundo. Estos están disponibles en distintos idiomas, como los utilizados de BERT y RoBERTa, y además admiten la posibilidad de usar aceleración por hardware. En cuanto a modelos de lenguaje avanzados no existe realmente ninguna alternativa a esta.

² <https://scikit-learn.org/stable/>

³ <https://spark.apache.org/mllib/>

⁴ <https://www.tensorflow.org/>

⁵ <https://keras.io/>

⁶ <https://huggingface.co/docs/transformers>

3.1.4 Jupyter Notebooks y Google Colab

Las Jupyter Notebooks se tratan de documentos interactivos formados por celdas que pueden incluir código, texto de marcado y gráficas. El hecho de poder ejecutar una celda y que se muestre la salida de la misma justo debajo en formato numérico, de texto o en una gráfica, hace que sean ideales para el ámbito del análisis y ciencia de datos, además de inteligencia artificial y propósitos educativos.

Por otro lado, Google Colaboratory⁷ es una plataforma de Google que dispone de recursos gratuitos de ejecución en la nube de documentos Jupyter Notebooks. Permite la ejecución de código Python, y cuenta con multitud de librerías populares ya preinstaladas como Tensorflow. También proporciona acceso gratuito a recursos de aceleración por hardware como GPUs y TPUs.

Se hizo uso de Colab debido a la gran carga computacional que supone el entrenamiento modelos de aprendizaje profundo, basados en grandes modelos del lenguaje como son BERT y RoBERTa, lo que hace que sea imposible entrenarlos en un equipo convencional sin hardware especializado como GPUs y TPUs. Aunque Kaggle es otra plataforma similar que proporciona acceso a hardware dedicado gratuitamente, se optó por Colab debido a su entorno ya integrado de librerías preinstaladas.

3.1.5 Kaggle y Playwright

Kaggle es una plataforma online que tiene una gran comunidad dedicada a la ciencia de datos. Esta además de albergar competiciones, conjuntos de datos y recursos para la enseñanza de aprendizaje automático, dispone de kernels de uso gratuito de ejecución en la nube de código en forma de Jupyter Notebooks en lenguaje Python y R. Estos kernels ponen a disposición del público recursos de computación como GPUs y TPUs para la aceleración por hardware del entrenamiento de modelos de aprendizaje automático. Además se puede programar la ejecución de cargas de trabajo (para la extracción de datos por ejemplo) de forma asíncrona con duración de hasta 12 horas.

Por otro lado, Playwright⁸ es una biblioteca de Python para la realización de *pruebas end-to-end* sobre aplicaciones web. Al igual que otras bibliotecas de pruebas como Selenium⁹ o Pupeteer¹⁰, se pueden utilizar para hacer *web scraping*. La elección de Playwright sobre las anteriores es debido la posibilidad de ejecutar el navegador en modo headless, es decir, sin interfaz de usuario. Este hecho lo hace ideal para la ejecución de scrapers en plataformas en la nube como Kaggle.

⁷ <https://colab.google/>

⁸ <https://playwright.dev>

⁹ <https://www.selenium.dev/>

¹⁰ <https://pptr.dev/>

De esta manera, se usaron conjuntamente estas dos herramientas para la extracción de los textos de los tweets que forman parte del conjunto de datos de 2016 ??.

3.2 Aplicación web y servicios

En esta sección, se explican las herramientas empleadas para la creación de la aplicación web. Estas incluyen las referentes al micro-servicio de perfilado, a la capa servidor que constituye el *back-end* de la aplicación y se comunica con la base de datos y a la capa cliente o *front-end* de la aplicación.

En este punto, cabe mencionar que se creó el módulo de perfilado automático como un micro-servicio al margen del resto de la capa servidor de la aplicación. La razón fue que los modelos usados para el perfilado estaban escritos en una versión de Python ya en desuso como es Python 2.7. De este modo, se decidió conservar estos modelos en esta versión debido a los problemas que supondría hacer la migración a una versión más reciente. No obstante, se juzgó adecuado desarrollar el resto de la capa servidor, que se comunica con la base de datos y el ya mencionado micro-servicio de perfilado, en una versión más reciente (como es Python 3.10) por motivos evidentes de seguridad y compatibilidad.

3.2.1 Back-end

Flask

Para la creación tanto de la capa servidor como del micro-servicio de perfilado se optó por Flask¹¹. Flask es un micro-framework para el desarrollo de aplicaciones web. Se le denomina micro-framework debido a su simplicidad y sencillez de uso. Simplifica enormemente la definición rutas para el procesado de peticiones HTTP, mediante el uso del decorador *route* delante de la función que procesará dicha petición.

También se barajó el uso de una alternativa popular como Django¹², el cual es un framework mucho más completo que dispone de forma de predeterminada de funcionalidad para facilitar la autenticación, y la administración de bases de datos, lo que lo hace ideal para proyectos grandes. No obstante, al ser más completo Django también tiene una curva de aprendizaje más alta por lo que en nuestro caso se decidió usar Flask ya que se consideró que la web a construir sería relativamente sencilla (sin necesidad de autenticación, o demasiada administración) y se prefirió la flexibilidad que nos aporta un framework pequeño como este.

Cabe mencionar que como opción para la validación de datos y serialización de los recursos del API se optó por Pydantic¹³. Esta es una moderna librería de python que permite la validación

¹¹ <https://flask.palletsprojects.com/en/2.3.x/>

¹² <https://www.djangoproject.com/>

¹³ <https://docs.pydantic.dev/latest/>

de los datos de entrada, de una forma declarativa, proporcionando un tipado estático de los mismos mediante el uso de las recientemente introducidas anotaciones de tipo de python (PEP 484). Esta característica es especialmente relevante en un lenguaje de tipado dinámico como es python facilitando así la detección de errores en tiempo de compilación.

Gunicorn

Como servidor web WSGI se ha optado por Gunicorn¹⁴. La especificación [WSGI \(Web Server Gateway Interface\)](#) se utiliza en el marco de las aplicaciones web desarrolladas en python para referirse a un módulo intermedio entre la aplicación web y el servidor de peticiones HTTP que define la comunicación entre los mismos. Estos servidores permiten la separación de responsabilidades entre la aplicación web que contiene la lógica de la misma y el servidor que se encarga de aspectos como la gestión de conexiones.

Los principales motivos de utilización de gunicorn son: su facilidad de uso, la disponibilidad de un entorno de ejecución concurrente con independencia entre hilos, la compatibilidad con diferentes frameworks basados en WSGI como Flask o Django; y por último la facilidad de integración con herramientas de administración de contenedores como es en este caso Docker.

3.2.2 *Front-end*

HTML5, CSS y JavaScript

[HyperText Markup Language, versión 5 \(HTML5\)](#)¹⁵ es un lenguaje de marcado que se ha convertido en el estándar para la construcción de páginas web. Está definido por el [W3C \(World Wide Web Consortium\)](#) que se encarga de la estandarización de las tecnologías asociadas a la web. Este lenguaje define la estructura y significado del contenido de las páginas web. La versión actual que entienden la mayoría de navegadores modernos es la 5.

[CSS \(Cascade Style Sheets\)](#)¹⁶ por otro lado, es un lenguaje de diseño gráfico, estandarizado también por el [World Wide Web Consortium](#), define el estilo de las páginas web en cuanto a su presentación visual. Su objetivos es marcar la separación entre el contenido de una página web y su presentación (colores, fuentes, *layouts*...). Cabe mencionar que el diseño de la web se ha realizado partiendo de una hoja de estilos basada en elementos como es [Water.css](#)¹⁷.

Por último, [JavaScript](#)¹⁸ es un lenguaje de programación de alto nivel, multiparagidma e

¹⁴ <https://gunicorn.org/>

¹⁵ <https://developer.mozilla.org/es/docs/Web/HTML>

¹⁶ <https://developer.mozilla.org/es/docs/Web/CSS>

¹⁷ <https://watercss.kognise.dev/>

¹⁸ <https://developer.mozilla.org/es/docs/Web/JavaScript>

interpretado que es utilizado principalmente en el desarrollo web para agregar interactividad y dinamismo a las páginas web a través de la manipulación del **DOM** (**Document Object Model**). Es un lenguaje multi-plataforma ya que se ejecuta en la mayoría de navegadores web modernos, aunque también es utilizado en entornos fuera del navegador. Hoy en día, se ha popularizado su uso mediante diferentes librerías o *frameworks* que facilitan el desarrollo web como pueden ser ReactJs, AngularJs o VueJs.

ReactJs, JSX y React Router

ReactJs¹⁹ es una librería JavaScript de código abierto diseñada para facilitar el desarrollo de interfaces web **SPA** (**Single Page Application**). Fue desarrollada por Facebook con la idea de facilitar la creación de interfaces «reactivas».

Está basada en la reutilización del código a través de la creación de componentes. Estos son bloques de código que representan una parte de la interfaz de usuario y se pueden usar para crear páginas web más complejas. Cada componente se encarga por tanto de renderizar una parte de la interfaz HTML.

Por este motivo, se utiliza la extensión de lenguaje **JSX** (JavaScript XML) que tiene una sintaxis muy similar al código HTML, con la diferencia de que admite la incorporación de otros componentes React dentro del mismo lenguaje, favoreciendo de esta manera la creación y mantenimiento de nuevos componentes.

Otro punto importante de React es el uso de un Virtual **DOM**, y es que React mantiene un árbol DOM virtual que se actualiza cada vez que cambia una parte de la interfaz para luego actualizar únicamente las partes del DOM del navegador que realmente han cambiado, favoreciendo así el rendimiento del navegador. Por otro lado, React no se considera un *framework* JavaScript entre otros motivos debido a que no incorpora elementos como el mantenimiento del estado o el enrutamiento de forma nativa. Por este razón, se ha escogido usar **React Router** que sirve para la gestión de la navegación de las páginas web (**SPA**) desarrolladas con React. Esta extensión, posibilita la navegación de diferentes vistas o páginas sin necesidad de recargar la interfaz completa. Asimismo, permite definir las diferentes rutas de la página web de forma declarativa, especificando que componente React se debe renderizar según la ruta que se cargue de la misma, facilitando la organización y comprensión del proyecto.

ChartJs

ChartJs²⁰ es un librería de JavaScript que facilita la creación de gráficos HTML. Pone a disposición del usuario una forma sencilla de crear gráficos y visualizaciones web interactivos y visualmente atractivos. Además de su facilidad de uso destaca la gran capacidad de personalización

¹⁹ <https://react.dev/>

²⁰ <https://www.chartjs.org/>

de sus visualizaciones.

Otra librería que se tuvo en cuenta para crear los gráficos de la web fue D3.js que se basa en la selección y manipulación directa de elementos HTML y SVG, tiene una gran flexibilidad y permite crear unos diseños mucho más personalizables que ChartJs. Sin embargo, su enfoque altamente imperativo contrasta bastante con la filosofía principalmente declarativa de React, además de esto su mayor complejidad de uso supone una barrera trabajando con plazos ajustados.

3.2.3 Persistencia

Con el objetivo de mantener un registro de las colecciones y usuarios perfilados en nuestra web, se juzgó necesario disponer de cierto tipo persistencia. Sin embargo, la hora de elegir el tipo de persistencia a utilizar, el hecho de que nuestro modelo de datos no tenga una estructura predefinida clara, la naturaleza no estructurada del texto proveniente de comentarios de redes sociales y la preferencia por una mayor flexibilidad de los datos que por su integridad, dio lugar que se optase por un sistema de gestión de bases de datos no relacional.

MongoDb

MongoDb²¹ es un sistema de gestión de bases de datos Not only SQL (NoSQL) de código abierto y orientado a documentos. Esto es en vez de guardar la información en tablas y filas como las bases de relacionales, lo hace en documentos BSON (Binary Json). Es ampliamente utilizado en el contexto del desarrollo web debido a su gran flexibilidad, escalabilidad y eficiencia para manejar grandes volúmenes de datos.

3.3 Herramientas de desarrollo

3.3.1 Docker

Docker²² es una plataforma que permite el desarrollo, envío y despliegue de aplicaciones de forma aislada en unidades llamadas contenedores. Los contenedores son unidades mínimas que incluyen todo el código, librerías y dependencias necesarias para ejecutar una aplicación. Al igual que las máquinas virtuales son una forma de virtualización de un equipo informático incluyendo su hardware, los contenedores se pueden ver como una forma de virtualización del sistema operativo. Esto permite que pueda haber múltiples aplicaciones distintas corriendo en una misma máquina, cada una de forma aislada con su propio entorno, mediante el uso de contenedores.

²¹ <https://www.mongodb.com/>

²² <https://www.docker.com/>

Los beneficios del uso de contenedores por tanto, son la portabilidad, aislamiento, seguridad, eficiencia de recursos y facilidad de desarrollo de las aplicaciones.

En nuestro caso, este aislamiento entre contenedores nos permite ejecutar en la misma máquina dos versiones distintas de Python ?? (una para el micro-servicio de perfilado y la otra para el resto del *back-end*). En Docker, los contenedores se definen a partir de imágenes que son paquetes de solo lectura que determinan el entorno y dependencias de las aplicaciones. Estas se definen a partir de un fichero llamado Dockerfile.

Por otra parte, la utilidad *Compose* de docker permite definir varios servicios que se ejecutan en contenedores independientes, pudiendo crear redes privadas entre ellos para la comunicación, evitando así exponer los mismos al exterior. Consecuentemente también se pueden especificar los puertos de red que se quieren exponer, el mapeo de directorios, y las dependencias entre los distintos servicios. Estos servicios son definidos en un fichero con formato YAML (docker-compose.yaml), permitiendo el despliegue de los mismos mediante una única instrucción de línea de comandos.

Por tanto, en nuestro caso se crearon cuatro servicios distintos para: *back-end*, *front-end*, micro-servicio de perfilado y base de datos (MongoDb); junto con una red privada para la comunicación entre los mismos. Una alternativa a Docker podría ser Podman²³ que recientemente está ganando popularidad, sin embargo, se optó por la primera debido a su gran repercusión, extensa documentación, y gran cantidad de imágenes predefinidas con las que cuenta.

3.3.2 Control de versiones

El control de versiones constituye una práctica esencial en el desarrollo de un proyecto software. Algunas de las razones de su uso son:

- Seguimiento de cambios: permite registrar y rastrear las modificaciones realizadas en archivos durante el tiempo.
- Revisión de cambios: permite devolver el proyecto a un estado previo del mismo en caso de errores.
- Colaboración en equipo: facilita la colaboración en equipo en proyectos con varios miembros.
- Gestión de ramas: facilita el desarrollo de versiones separadas del software de forma simultánea.

Por estos motivos se escogió el uso de **Git**²⁴. Git es un software de control de versiones desarrollado en 2005 por Linus Torvalds y ampliamente utilizado desde entonces. Permite

²³ <https://podman.io/>

²⁴ <https://git-scm.com/>

mantener un registro de los cambios realizados a lo largo del ciclo de vida de un proyecto de forma eficiente y distribuida, a través del uso de repositorios locales y/o remotos.

Como plataforma de alojamiento de repositorios remotos se ha optado por Github²⁵.

3.3.3 Taiga

3.3.4 Editor de código

Visual Studio Code²⁶ es un editor de código gratuito desarrollado por Microsoft. Entre sus puntos fuertes están que es ligero, altamente personalizable y cuenta con funcionalidades de **IDE (Integrated Development Environment)** como: depuración, integración de control de versiones, resaltado de sintaxis, autocompletado, refactorización. Su gran cantidad de extensiones hace que tenga soporte para casi cualquier lenguaje de programación.

Debido a la comodidad que ofrece y la familiaridad del desarrollador con el editor, se utilizó durante todo el proceso de desarrollo incluyendo algoritmos de perfilado, *back-end* y *front-end*. Aunque estrictamente no sea considerado un **IDE** como tal, la gran capacidad de personalización que ofrece, puede sustituir la necesidad de emplear otro **IDE**.

3.3.5 Prototipado

Para realizar los prototipos de la interfaz web se ha utilizado **Balsamiq**²⁷. Balsamiq es una herramienta de prototipado para realizar diseños de interfaz de usuario de baja fidelidad, conocidos como *mockups*. Esta herramienta cuenta con aplicación de escritorio, así como una versión web de la misma que te permite alojar tus prototipos en la nube. Además cuenta con una amplia biblioteca de componentes prediseñados como botones, cuadros de texto, iconos que los desarrolladores pueden usar para armar fácilmente sus *mockups*.

²⁵ <https://github.com/>

²⁶ <https://code.visualstudio.com/>

²⁷ <https://balsamiq.com/>

Capítulo 4

Metodología y gestión del proyecto

En este capítulo se aborda la metodología seguida para la realización del proyecto, explicando los principios fundamentales da la misma además de los motivos de su elección, junto con las adaptaciones que se consideraron necesarias en el contexto de este proyecto. Por otro lado, también se describe la planificación realizada para la consecución del proyecto incluyendo el análisis de riesgos realizado.

4.1 Metodología

La metodología constituye un aspecto esencial para garantizar que un proyecto se desarrolle de manera eficiente, produzca un producto de alta calidad y cumpla con los objetivos propuestos. Esta proporciona una estructura y un enfoque organizado que permite aumentar la productividad y evitar las desviaciones de estos. La elección de una metodología adecuada es crucial para el éxito del proyecto y debe adaptarse a las necesidades, características y contexto específicos tanto del proyecto como del equipo de trabajo.

En nuestro caso, existen varios factores dentro del proyecto que condicionan en gran medida la elección del enfoque metodológico:

- El carácter altamente innovador de la propuesta del proyecto.
- La escasez relativa y falta de homogeneidad en los experimentos en técnicas de clasificación de texto, como perfilado automático, en idioma español en comparación con el inglés.
- El escaso conocimiento inicial del alumno acerca de estas técnicas y la relativa complejidad y novedad de los algoritmos de procesado de lenguaje actual.
- La falta de acotación y ambigüedad de los requisitos iniciales del producto propuesto.

Teniendo estos factores en cuenta, las metodologías ágiles se destacan como una opción atractiva. Estas metodologías ofrecen la flexibilidad necesaria para adaptarse a cambios en los requisitos a medida que se adquiere un mejor conocimiento del dominio, promueve el aprendizaje continuo dentro de un entorno complejo y permiten la entrega incremental de resultados. Dentro de las opciones disponibles dentro de este grupo de metodologías se ha optado por Scrum ([?]) por ser una de las de mayor relevancia y conocimiento en el sector TIC actual.

4.1.1 Scrum

Scrum es un marco de trabajo ágil ampliamente utilizado en el desarrollo de software y en la gestión de proyectos. Fue creado para abordar los desafíos de proyectos complejos y cambiantes, permitiendo la entrega de productos con el máximo valor posible optimizando la creatividad y productividad en el proceso. En la figura ?? se puede ver un diagrama explicativo donde se muestra el proceso de Scrum.

Scrum se basa en la teoría del control empírico de procesos, o empirismo. Esta sostiene que el

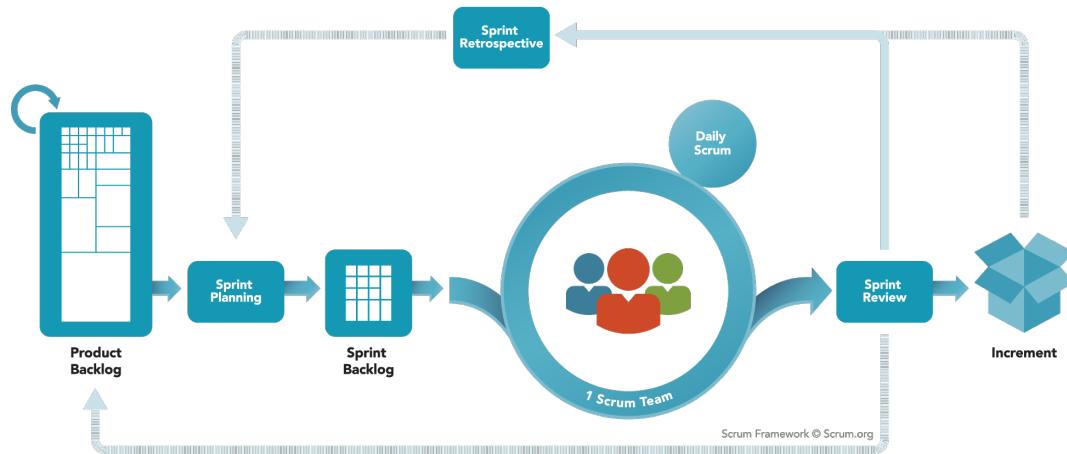


Figura 4.1: Diagrama de la metodología *Scrum*.

conocimiento proviene de la experiencia y de tomar decisiones basadas en lo que se sabe. Este marco de trabajo propone un enfoque iterativo e incremental para mejorar la predictibilidad y controlar el riesgo. Tres pilares respaldan cada implementación del control empírico de procesos: transparencia, inspección y adaptación.

- **Transparencia:** Aspectos significativos del proceso deben ser visibles para quienes son responsables del resultado. La transparencia requiere que esos aspectos estén definidos

por unos estándares comunes para que los responsables tengan un entendimiento acorde acerca de estos resultados.

- **Inspección:** Los usuarios de Scrum deben revisar con frecuencia los artefactos de Scrum y el progreso hacia un objetivo común para detectar desviaciones no deseadas.
- **Adaptación:** Si durante la inspección del trabajo un miembro del equipo determina que ha habido desviaciones significativas con respecto a los objetivos iniciales el proceso y las tareas deben adaptarse lo antes posible con el objetivo de reducir o eliminar esta desviación.

Roles

Un equipo de Scrum debe ser auto-organizativo y multifuncional: el equipo debe ser capaz de dirigirse a sí mismo y debe poseer las competencias necesarias para la consecución del trabajo sin dependencias externas. Para ello se definen los siguientes roles:

- *Product Owner:* este es responsable de maximizar el valor del producto resultado del trabajo del *equipo de desarrollo*. La forma en que esto se consiga puede variar de un equipo a otro.
- *Equipo de desarrollo:* este está formado por profesionales que se encargan de la realización del producto «Terminado» después de cada incremento, el cual constituye los resultados del mismo.
- *Scrum master:* este es responsable de promover y apoyar que se siguen las pautas correctas de realización del Scrum en el proyecto. Debe ayudar a que el equipo entienda los conceptos de scrum para poder maximizar el valor entregado.

Eventos

Además de los roles anteriores existen una serie de eventos pre-establecidos con una duración predeterminada. Estos tienen como finalidad la establecer una regularidad en el proyecto de forma que se minimicen las reuniones no establecidas por scrum. Estos eventos son:

- *Sprint:* constituye un incremento o iteración del producto. Tiene una duración generalmente de entre 2-4 semanas en el que se desarrolla una parte del producto. Durante el sprint, el equipo se compromete a entregar un conjunto específico de funcionalidades.
- *Sprint planning:* la planificación del sprint es una reunión entre todos los miembros del equipo, donde se define de forma colaborativa el trabajo que se llevará a cabo durante

el mismo. Tiene una reunión máxima de ocho horas, aunque suele ser menor según la duración del sprint.

- *Daily scrum*: se trata de una breve reunión (máximo 15 minutos) donde el equipo de trabajo planifica las tareas que se llevarán a cabo durante el día. Además, se realiza una inspección para verificar que el equipo progrese adecuadamente hacia los objetivos marcados.
- *Sprint review*: esta reunión de carácter más informal se da al final de cada sprint. En ella el equipo completo (junto a *stakeholders*¹) realiza la inspección del incremento demostrando los resultados del sprint y añadiendo al *product backlog* aquellas funcionalidades «terminadas».
- *Sprint retrospective*: reunión de carácter más filosófico que se da entre el *Sprint review* y el próximo *Sprint planning* donde el equipo «se inspecciona a sí mismo»: se comparten impresiones acerca del incremento, reflexionando acerca de aquellos aspectos del proceso que han ido bien y aquellos que son sujetos de mejora.

Artefactos

Los artefactos de Scrum representan valor o trabajo diseñados para mejorar la transparencia y facilitar la inspección del proyecto. Estos artefactos son los siguientes:

- *Product backlog*: esta es una lista ordenada de todas aquellas funcionalidades y tareas que serán necesarias en el producto. Este representa los requisitos deseado del producto. Esta nunca está del todo completa: evoluciona a lo largo del tiempo con el producto a medida que los requisitos cambian o se entienden mejor.
- *Sprint backlog*: conforma el conjunto de elementos del *Product backlog* escogidos para su realización en el marco de un *Sprint*. Este incluye el plan diseñado para la entrega de incremento en cuestión y la realización del objetivo del mismo.

4.1.2 Adaptaciones de *Scrum* al proyecto

Varios factores como la presencia de un único miembro en el equipo de desarrollo, la falta de regularidad en la dedicación del mismo debido a obligaciones académicas han impedido que se haya podido llevar a cabo la metodología seleccionada de forma totalmente fiel a sus pautas. Por un lado se ha realizado la siguiente asignación de roles:

- El rol de *Product owner* ha sido desempeñado por

¹ Aquellos interesados en la realización del proyecto.

- El rol de *Scrum master* ha sido desempeñado por
- El de equipo de trabajo estuvo constituido por un único miembro, el alumno Nicolás Míguez García.

A continuación, se presentan las modificaciones implementadas en la metodología original para adecuarla a las necesidades de este proyecto:

- La duración de cada *sprint* ha ido variando (entre 3 y 5 semanas).
- De la misma forma que la duración, la carga de trabajo en cada sprint ha ido variando, en función de la dedicación disponible por parte del equipo de desarrollo.
- Dado que el equipo de desarrollo está compuesto por una sola persona, el evento de *Daily scrum* pierde el sentido de su existencia. Sin embargo, se mantiene la práctica de la revisión del trabajo diario como parte de las responsabilidades del alumno.

Historias de usuario

Para definir los elementos del *Product backlog* (y por tanto los requisitos funcionales del proyecto) se han usado las historias de usuario. Estas constituyen una técnica muy utilizada en metodologías ágiles para la especificación de requisitos debido a su sencillez y claridad. Estas se tratan descripciones breves (una o dos líneas) de las funciones del sistema desde la perspectiva del usuario. Las historias de usuario tienen de la siguiente forma:

Como <rol> quiero <algo> para poder <beneficio>.

Por otro lado, debido a que la primera fase del proyecto estaba más orientada hacia la investigación y desarrollo de los algoritmos de perfilado automático que al desarrollo de un producto software en sí mismo, el modelado de requisitos desde la perspectiva de usuario no tiene demasiado sentido. Es por ello que los requisitos de esta fase se modelan como tareas técnicas orientadas al desarrollador en vez de al usuario.

4.2 Gestión del proyecto

La gestión del proyecto se encarga de la planificación, seguimiento y control del mismo, teniendo en cuenta factores como la estimación, la asignación y control de recursos, y la gestión de riesgos. Esto proporciona una visión global acerca del estado del proyecto en cualquier momento de su desarrollo.

4.2.1 Recursos**4.2.2 Planificación****4.2.3 Gestión de riesgos**

Capítulo 5

Análisis de requisitos

En este capítulo se muestra el análisis de requisitos efectuado para el desarrollo del dashboard web.

5.1 Requisitos funcionales

Como ya se ha mencionando en el capítulo ?? se han usado las historias de usuarios como técnica de especificación de las funcionalidades que debe cubrir el producto. A continuación en la tabla ?? se muestran las historias de usuario extraídas.

Id	Historia de usuario
H1	Como usuario quiero perfilar el género y edad de los usuarios de una colección
H4	Como usuario quiero ver las estadísticas de edad en forma de gráfico de los usuarios perfilados en una colección
H7	Como usuario quiero ver las estadísticas de género en forma de gráfico de los usuarios perfilados en una colección
H5	Como usuario quiero ver estadísticas propias de la colección perfilada como nombre, tiempo, algoritmo y usuarios totales de la misma
H2	Como usuario quiero elegir entre los distintos algoritmos de perfilado disponibles para ver las diferencias entre las predicciones de cada uno de ellos
H3	Como usuario quiero ver la lista de usuarios perfilados con su información asociada
H6	Como usuario quiero ver las publicaciones de un usuario específico de la colección junto a sus datos demográficos para comparar el tipo de redacción y temáticas tratadas según los grupos demográficos predichos
H8	Como usuario quiero guardar las colecciones previamente perfiladas para acceder a ellas sin necesidad de volver a perfilarlas de nuevo
H9	Como usuario quiero ver una lista ordenada temporalmente que incluya datos generales de las colecciones previamente perfiladas para buscar entre ellas con facilidad

Tabla 5.1: Requisitos funcionales de la aplicación.

5.2 Requisitos no funcionales

Los requisitos no funcionales, por otro lado, no se ven reflejados en funcionalidades independientes que satisfacen necesidades específicas de una aplicación. Sino que, estos constituyen requerimientos generales sobre cómo debería comportarse el sistema de forma global y están relacionados con la operatividad y calidad software. En nuestro sistema se ha priorizado nuestra atención en los enumerados en la tabla ??.

Id	Nombre	Descripción
RNF-1	Usabilidad	La aplicación debe tener un interfaz intuitiva y familiar con el usuario permitiendo una interacción lo más eficiente posible.
RNF-2	Mantenibilidad	La aplicación debe ser diseñada de forma legible y modular con el objetivo de permitir extender fácilmente su funcionalidad.
RNF-3	Fiabilidad	El sistema debe ser confiable y consistente, siendo tolerante a fallos.
RNF-4	Portabilidad	La aplicación debe ser capaz de ejecutarse en distintos entornos sin modificaciones significativas.
RNF-5	Escalabilidad	El sistema debe ser fácilmente escalable ante el crecimiento de la carga de trabajo debido a un aumento de la demanda.

Tabla 5.2: Requisitos no funcionales detectados.

5.3 **Modelo de datos**

Como parte del análisis del dominio se realizó un diagrama entidad-relación, para conocer las entidades relevantes de nuestro sistema y enfocar de manera clara el diseño de la base de datos. No obstante, aunque las entidades principales del mismo se han mantenido, debido al enfoque incremental del proyecto, a medida que ha ido avanzando estas han estado sujetas a cambios. Por lo que en la figura ?? se muestra el diagrama entidad-relación al finalizar el proyecto.

Como se puede observar se tienen únicamente dos entidades fundamentales relacionadas entre sí: *Colección* y *Usuario*.

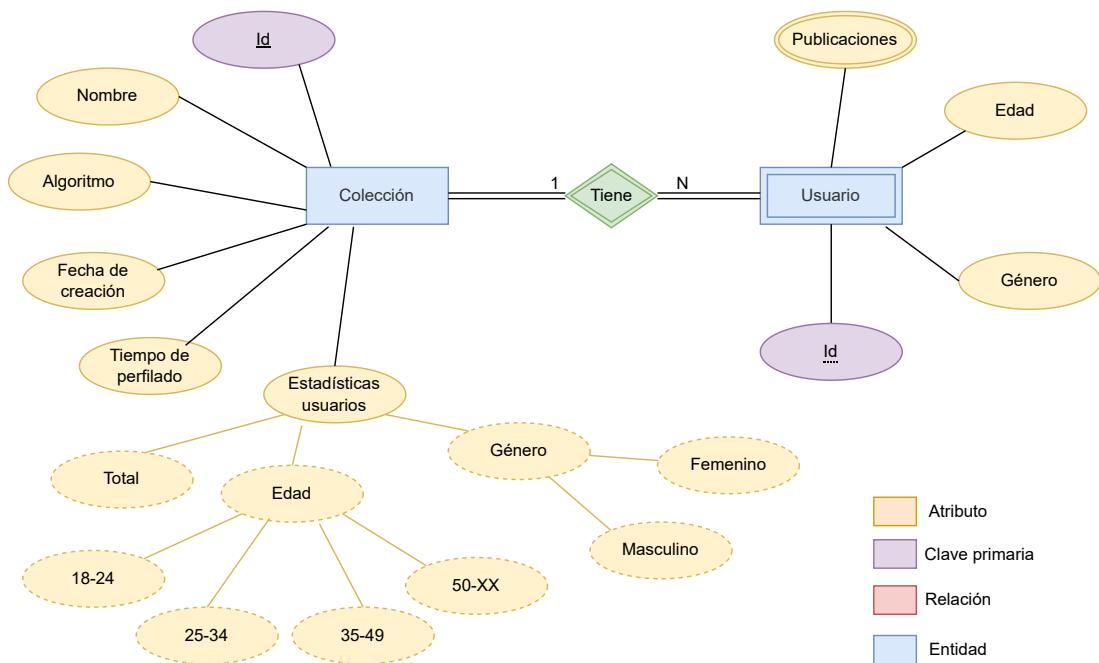


Figura 5.1: Diagrama de entidad-relación del sistema.

Capítulo 6

Diseño

6.1 Prototipado

El prototipo de una aplicación conforma una versión inicial «hueca» de un producto. Es una herramienta fundamental para visualizar conceptos de diseño, funcionalidad y interacción con el usuario en una etapa temprana de desarrollo. Permite una comunicación efectiva entre el equipo de desarrollo y el cliente, haciendo posible la identificación de problemas antes de la implementación del producto, donde un cambio de requisitos es poco costoso, garantizando que la aplicación satisfaga las necesidades y expectativas de los usuarios finales.

En nuestro caso, como técnica de prototipado hemos optado por utilizar *wireframes*. Los *wireframes* son una herramienta de prototipado basados en representaciones visuales simplificadas de la interfaz de una aplicación. En estos se plasma la estructura y disposición de los elementos fundamentales de la aplicación sin añadir detalles decorativos como elementos de diseño gráfico o colores, centrándose en la funcionalidad y navegación de la misma.

En la figura ?? se puede ver el diseño de la página principal de nuestra interfaz. Como vemos en ella se expone la funcionalidad principal de perfilar una colección de usuarios a partir de un archivo. Esta página, está formada por un formulario donde se puede seleccionar el algoritmo de perfilado junto con el fichero que contiene la colección a perfilar. Por otro lado también tiene una barra de navegación en la parte de arriba que se mantendrá para el resto de *wireframes* de la aplicación.

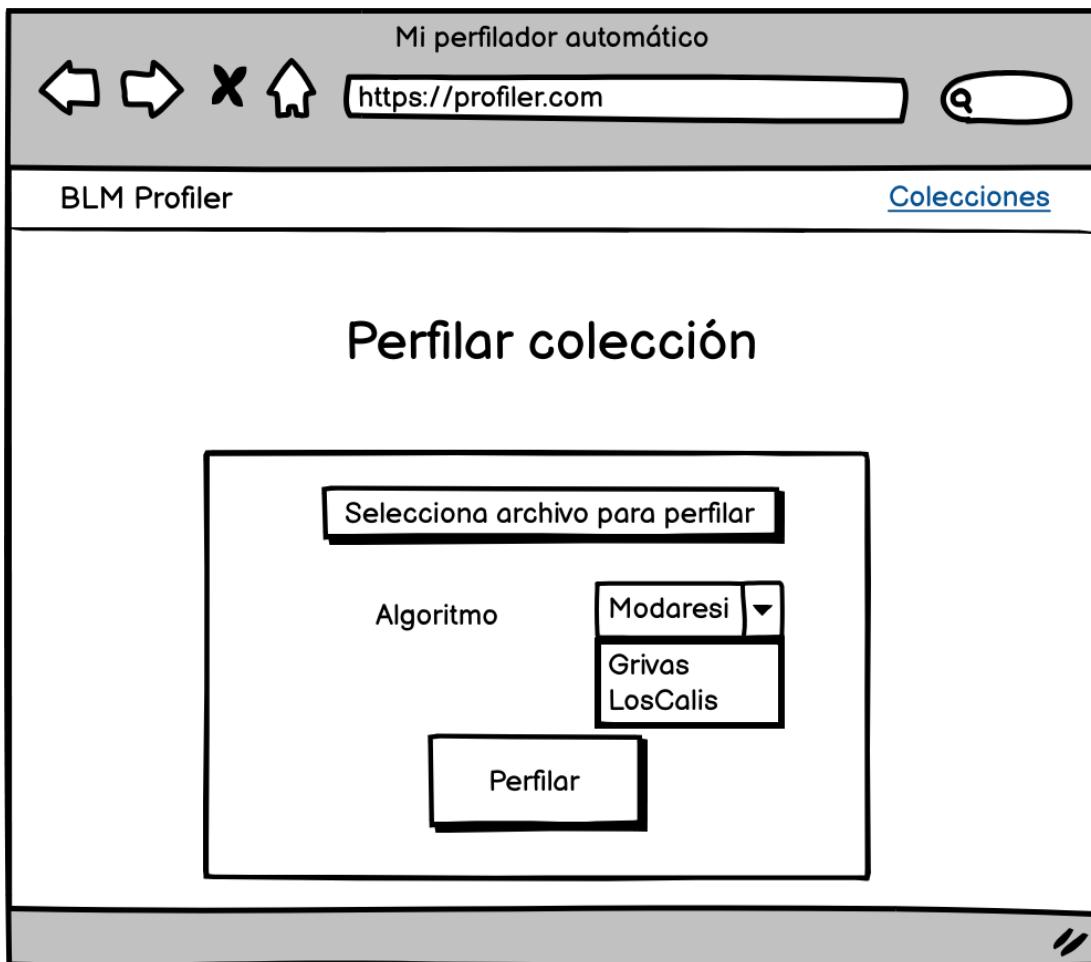


Figura 6.1: Prototipo de la página principal de la aplicación.

En el caso de seleccionar el enlace «Colecciones», de la barra de navegación, nos iríamos a una página como la de la figura ???. En esta se muestra una tabla, ordenada temporalmente, con todas las colecciones de usuarios previamente perfilados de nuestra aplicación, junto a información general sobre cada una como: algoritmo, usuarios totales o fecha de perfilado.

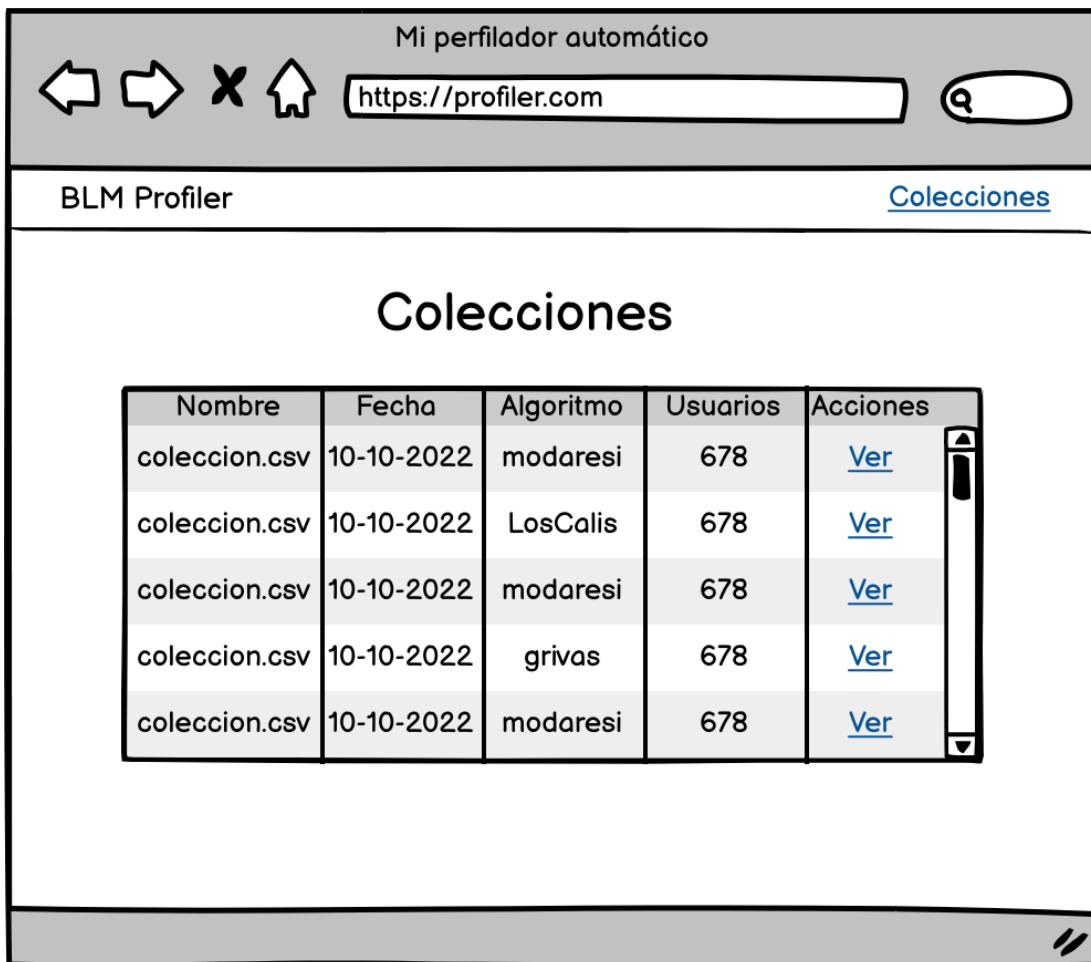


Figura 6.2: Prototipo de ver colecciones perfiladas.

En el caso de seleccionar «ver» una colección de la lista de colecciones (??) o tras perfilar una nueva en la página principal (??) nos iríamos a la visualización del *dashboard* o cuadro de mando de la colección (??). En este *wireframe* se muestran dos gráficas sobre los datos demográficos, edad y género en este caso, de los usuarios perfilados. Además también se muestra una tabla con la lista de usuarios de la colección junto con las predicciones realizadas sobre cada uno y un enlace sobre una muestra de una publicación del mismo. Por otro lado, en la parte de arriba tenemos unos desplegables que nos permiten filtrar los datos de la tabla y gráficas según la categoría elegida; junto con unas tarjetas en las que se muestran detalles generales de la colección.

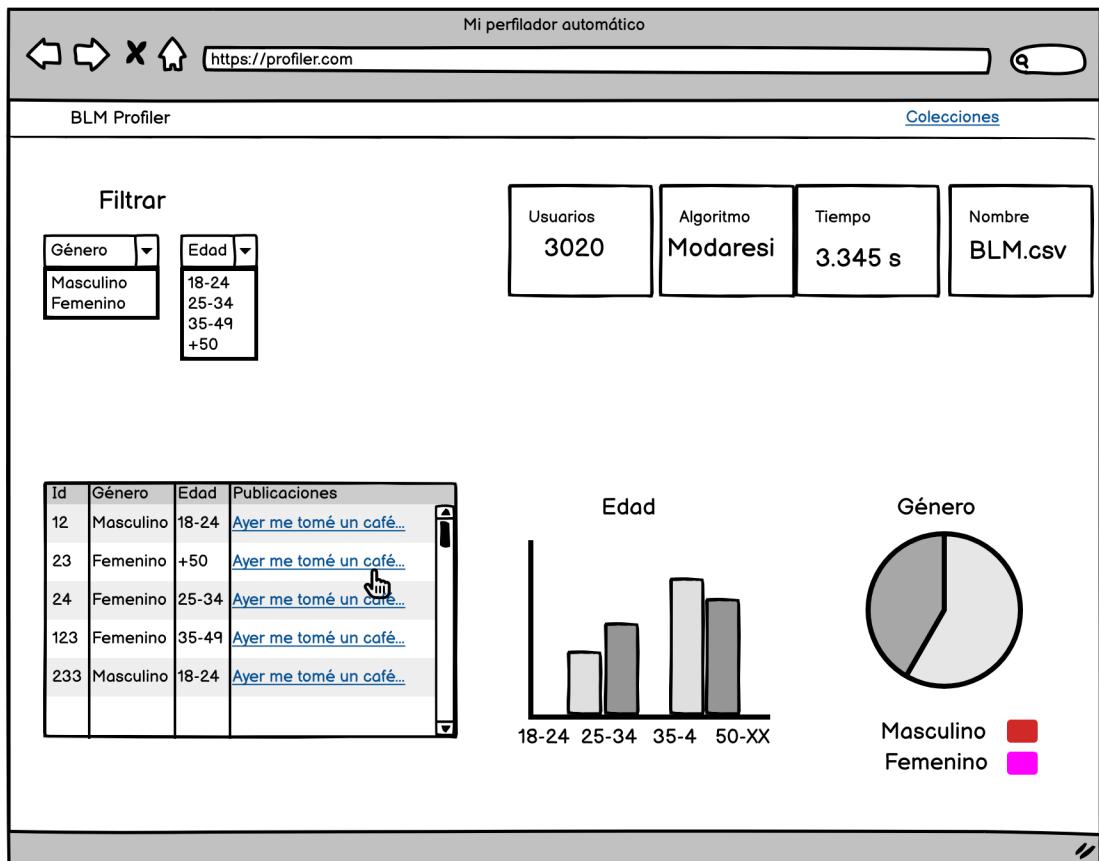


Figura 6.3: Prototipo del cuadro de mando de la aplicación.

En el caso de seleccionar algún filtro en el *dashboard* se añadiría al mismo una lista, como la de la figura ??, con aquellos seleccionados donde se podrían quitar de nuevo estos. Y se mostrarían las modificaciones en los gráficos y lista.

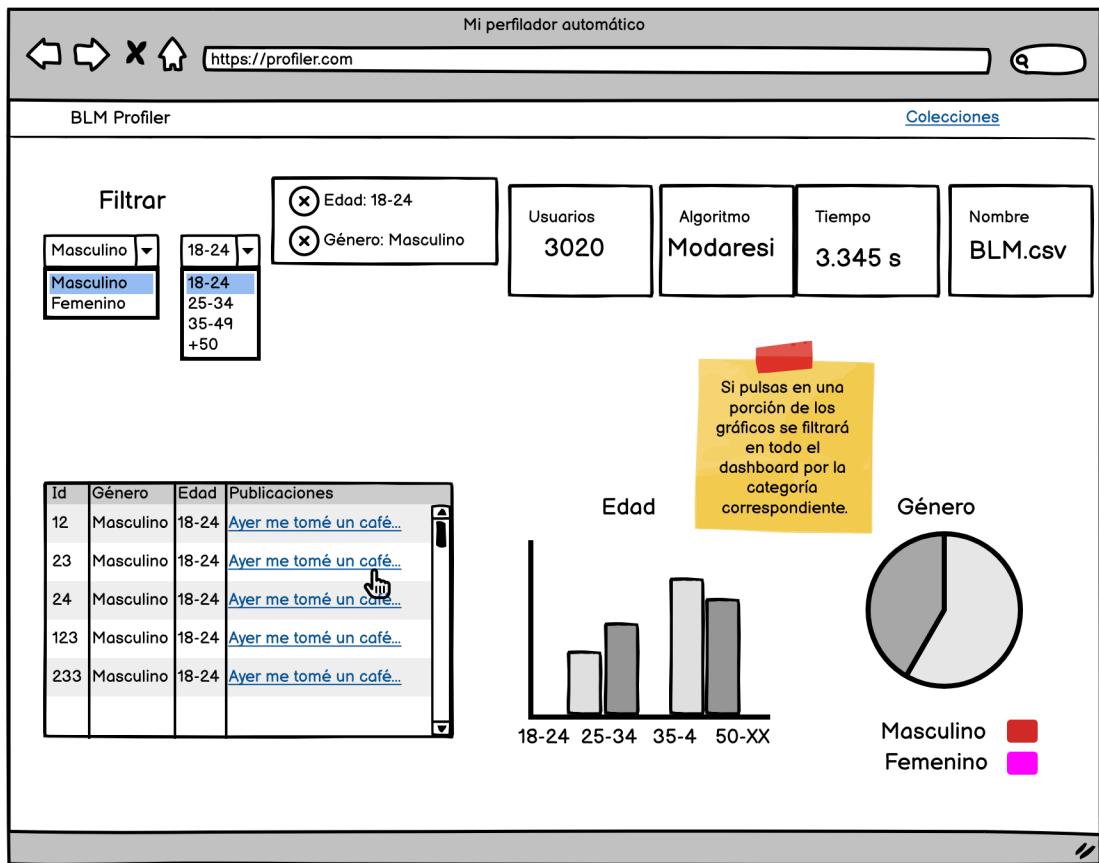


Figura 6.4: Prototipo del cuadro de mando al especificar filtros sobre edad y género.

Por último, al seleccionar el enlace de la publicación de un usuario del *dashboard*, se navegaría hasta la última página de nuestra web, donde se podría ver en detalle las publicaciones del usuario seleccionado junto con sus datos generales. Además, esta tendría un botón para volver al *dashboard* de la colección. En el *wireframe* de la figura ?? se muestra la disposición de la misma.

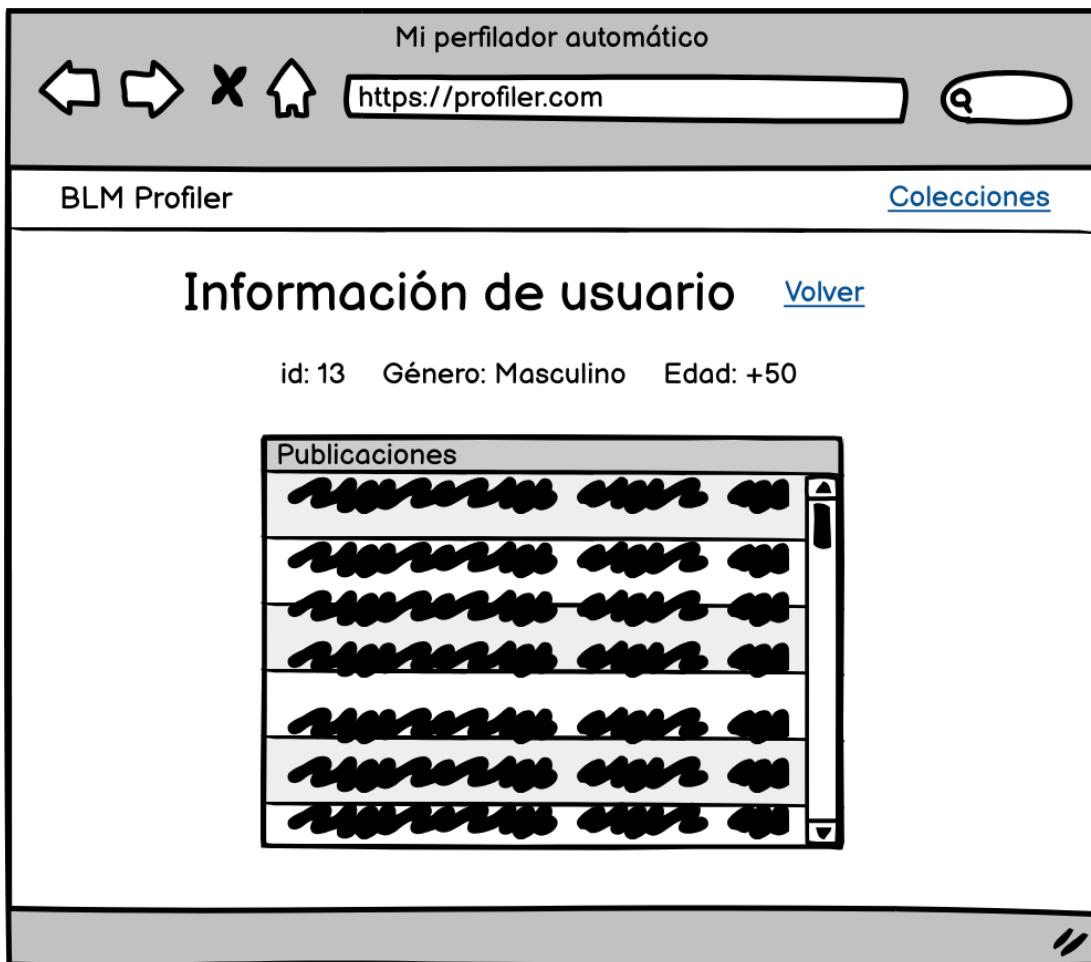


Figura 6.5: Prototipo de ver información de usuario en detalle.

6.2 Arquitectura

El diseño de la arquitectura de una aplicación es un aspecto de vital importancia para cumplir con los requisitos tanto funcionales como no funcionales de la misma. Dada la naturaleza ágil de la metodología seguida, la arquitectura del sistema ha ido evolucionando durante el desarrollo del mismo, pasando de un diseño más sencillo a uno más complejo. Por este motivo, en este apartado se describirá el diseño final de la arquitectura propuesta.

En nuestro caso, como se puede ver en la figura ?? hemos seguido una arquitectura cliente-servidor distribuida o en n-capas, donde el servidor está dividido en servidor web, servidor de aplicación y servidor de datos. El servidor web es el que aloja la aplicación web SPA, que su vez está dividido en interfaz de usuarios y capa de acceso a servicios. El servidor de aplicación se encarga de exponer la API REST que ejecuta la lógica de negocio de la aplicación y el acceso a

datos¹. Por último, el servidor de datos es el encargado de almacenar los datos de la aplicación.

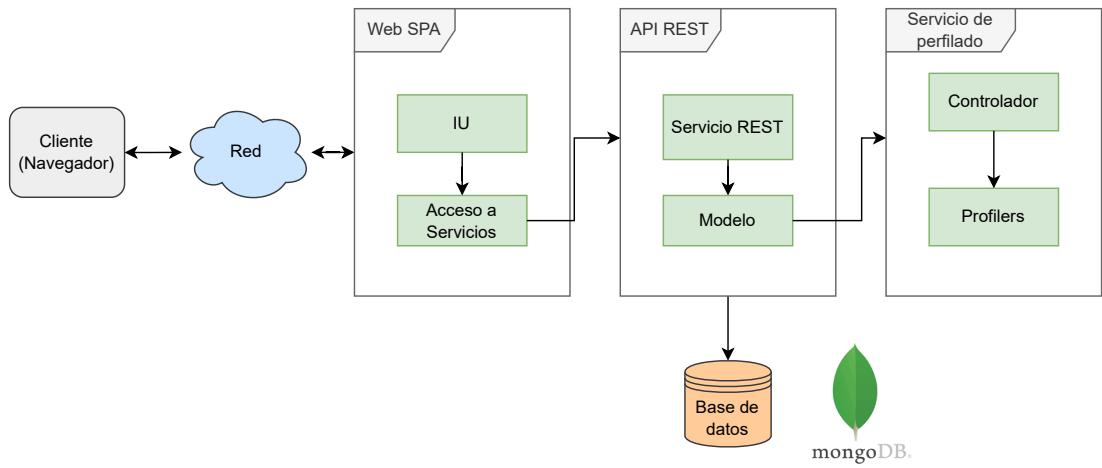


Figura 6.6: Diagrama de la arquitectura global del sistema.

Además se puede ver que existe un servidor adicional, que se podría definir como un micro-servicio, cuya única funcionalidad es la de ejecutar los algoritmos de perfilado sobre los datos de usuarios que le envíe la API REST y devolver estos perfilados. Por lo tanto, la el servidor de la aplicación también actúa como cliente de este servicio.²

La decisión de tener esta arquitectura distribuida en varias capas tiene varias ventajas:

- Para empezar, se favorece la escalabilidad del sistema RNF-5. El hecho de tener servidores distintos para «frontend», «backend» y el micro-servicio de perfilado permite que se puedan escalar horizontalmente estos de forma selectiva. Esto es importante porque el micro-servicio de perfilado representa la carga más significativa del sistema en cuanto a recursos computacionales y tiempo, lo que la convierte en un posible cuello de botella para el funcionamiento general del mismo. Por otra parte, el servidor web es el que menos recursos computacionales consume.
- Por otro lado, se favorece la mantenibilidad y extensibilidad del software (RNF-2). El hecho de tener las distintas capas desacopladas posibilita el poder extender o reemplazar cualquiera de ellas sin necesidad de hacer un cambio en las demás³. Un ejemplo de esto sería añadir un cliente móvil además del web, añadir otra API que consumiera el micro-servicio de perfilado, o incluso extender este último para dar soporte a nuevos

¹ Y de comunicarse con el micro-servicio de perfilado como se comenta más adelante.

² La decisión de separar este servicio del resto del «backend» se tomó inicialmente debido a la dificultad de hacer la migración de los algoritmos existentes a una versión actual de python como se comentará en el capítulo de desarrollo, sin embargo, este enfoque tiene otros beneficios importantes que se comentan a continuación.

³ Este hecho se alinea con unas buenas prácticas de ingeniería, cumpliendo con un principio de diseño fundamental como es el abierto-cerrado.

algoritmos o mejoras de rendimiento. De igual forma esta separación nos permite elegir el lenguaje de programación o tecnología más adecuada para la implementación de cada capa.

- Por último, otra ventaja de este enfoque sería el aumento de fiabilidad del mismo (RNF-3). El fallo en cualquier capa no implica que el resto no puedan seguir funcionando.

6.2.1 «Backend»

Como ya se ha comentado el «backend» está dividido en dos capas: servicios y modelo que a su vez se podría dividir en lógica de negocio y acceso a datos. Para la implementación de estas capas se han seguido diferentes patrones de diseño muy comunes en el desarrollo de este tipo de aplicaciones, como pueden ser el uso del patrón fachada para exponer la funcionalidad de los servicios, el uso de DTOs para la validación y serialización de los datos, la implementación de DAOs con operaciones CRUD para acceder a la base de datos o el modelado de la conexión a la base de datos como un *singleton*. En la figura ?? se puede ver el diagrama de clases del «backend».

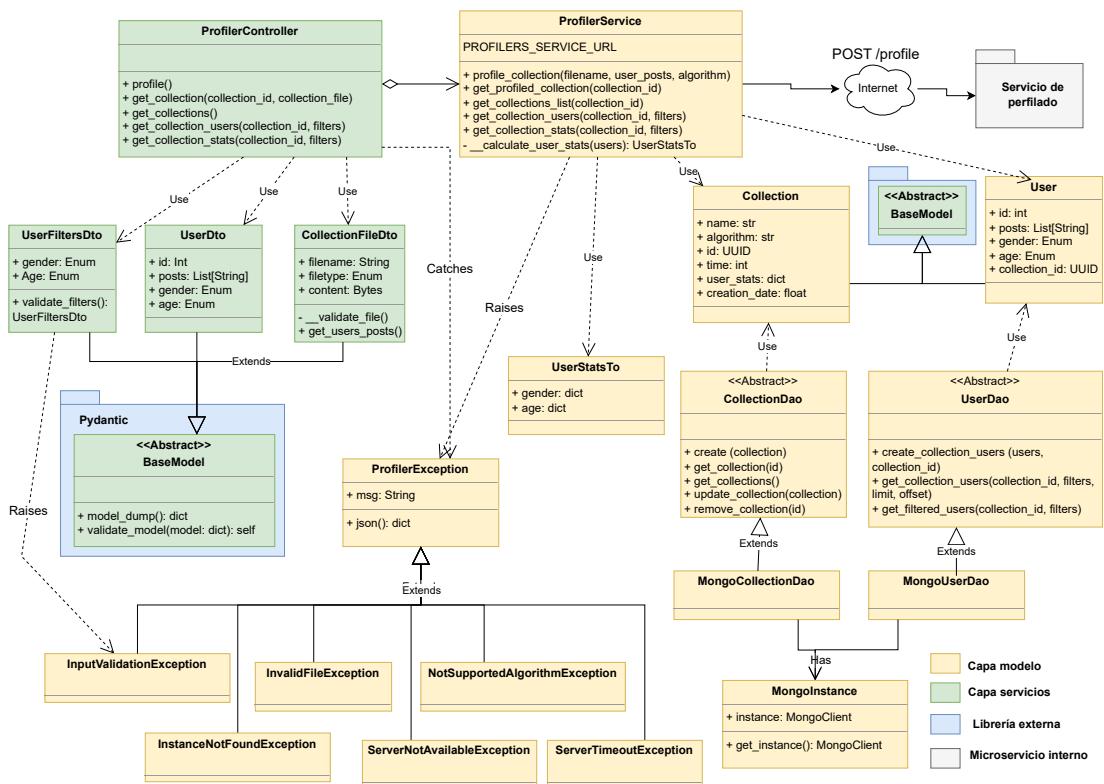


Figura 6.7: Diagrama de clases final del API REST del sistema.

6.2.2 Micro-servicio de perfilado

En cuanto al micro-servicio de perfilado se puede comentar que también se ha seguido un enfoque REST, con un único endpoint de entrada, para la comunicación con el «backend». En el diagrama de la figura ?? se puede ver la estructura de este micro-servicio.

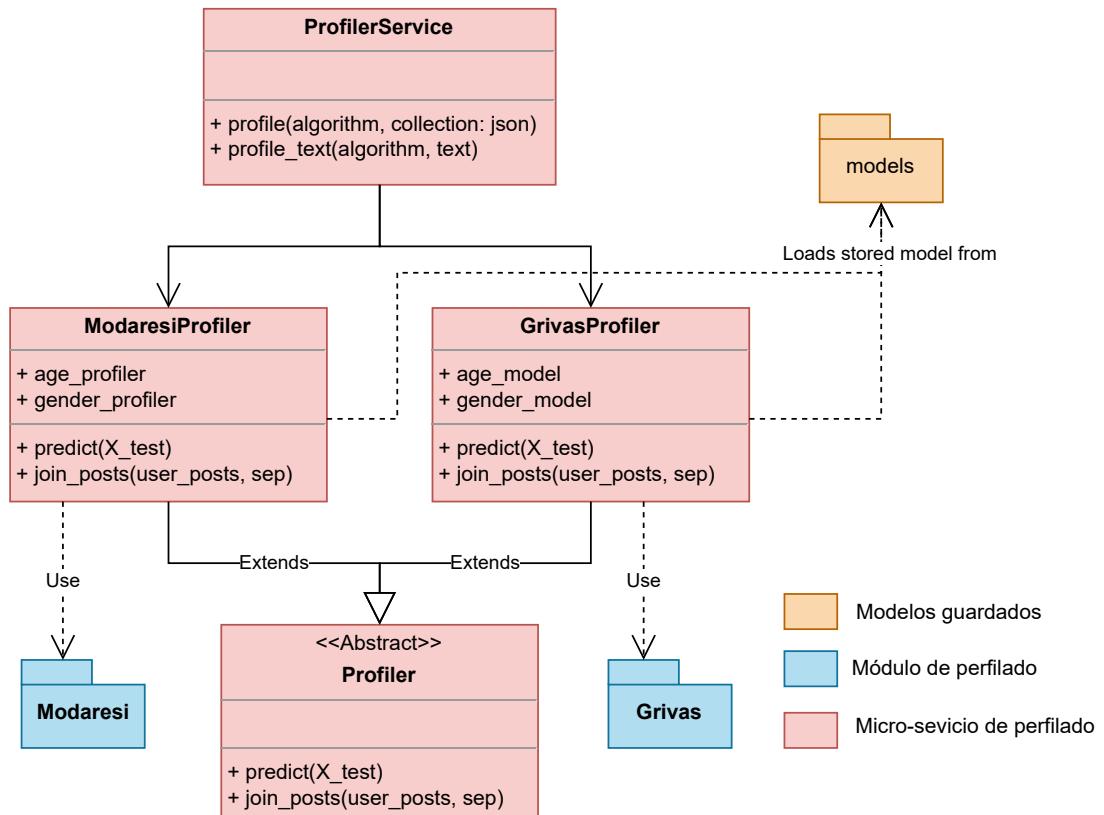


Figura 6.8: Diagrama de clases final del micro-servicio de perfilado.

En el mismo se muestra que las clases que implementan los *profilers* cargan los modelos y pipelines de preprocessamiento de un módulo llamado «models» donde se encuentran todos aquellos modelos necesarios ya entrenados y listos para el perfilado de los algoritmos.

Estos modelos guardados tienen dependencias con los módulos de perfilado, que se ven color azul en el diagrama, que son aquellos que contienen el código para el preprocessamiento y creación de los modelos de AA. Sin embargo, como son módulos de autores externos adaptados para el proyecto se decidió no mostrar su estructura en el diagrama de clases.

Capítulo 7

Desarrollo

7.1 Sprints

7.1.1 Sprint 0: 8-2-2023 a 14-3-2023

:

- Configuración del entorno: búsqueda y experimentación con librerías de procesado del lenguaje natural como Spacy o NLTK
- Introducción a estado del arte del perfilado automático de usuarios en edad y género.

7.1.2 Sprint 1: 14-3-2023 a 29-3-2023

:

- Experimentando técnicas de preprocesado de los textos de la colección como: lematización, sustitución de emojis por sus significado, etc.
- Investigación y búsqueda de conjuntos de entrenamiento en idioma español para la tarea de perfilado automático de usuarios.
- Estudio del lenguaje empleado en el corpus BLM con el objetivo de ***** meter la cabeza en el problema*****

7.1.3 Sprint 2: 29-3-2023 a 17-4-2023

:

- Búsqueda, comparación y selección de algoritmos de perfilado automático
- Selección de conjuntos de entrenamiento a usar para entrenar los modelos seleccionados.

7.1.4 Sprint 3: 17-4-2023 a 20-6-2023

:

- Carga de la colección BLM, agrupación de los usuarios y preprocesado de los posts.
- Adaptación y ejecución de pruebas con algoritmo LosCalis. Investigación de malos resultados obtenidos e intento de realización de mejoras: cambios de preprocesado, hiperparámetros, asignación de pesos a salidas en la red.
- Creación de scraper de twitter para poblar los ficheros que constituyen el dataset de 2016.

7.1.5 Sprint 4: 20-6-2023 a 20-7-2023

:

- Ejecución en batches del scraper para descargar la colección entera.
- Adaptación y ejecución de algoritmos de modaresi y grivas.
- Realización de pruebas con estos (pruebas con distintos parámetros, features, preprocesado).
- Documentación de resultados y redacción de los dos primeros capítulos de la memoria (?? y ??).

7.1.6 Sprint 5: 20-7-2023 a 5-8-2023

- Inicio del backend: creación del endpoint, entrenamiento y guardado de algoritmos de perfilado para producción.
- Diseño arquitectura backend.
- Containerización

7.1.7 Sprint 6: 6-8-2023 a 21-8-2023

- Desarrollo modelo base de datos.
- Diseño e implementación del frontend.

7.1.8 Sprint 7: 21-8-2023 a 11-8-2023

- Como usuario quiero poder ver los distintos datos demográficos de los usuarios de la colección en forma de lista.
- Como usuario quiero ver en detalle los posts que ha publicado un usuario concreto de la colección.
- Estilado de elementos del frontend.

Capítulo 8

Análisis colección #BLM

Es importante recordar que el trabajo previo de: la comparación entre algoritmos de perfilado de usuarios y el análisis, diseño e implementación de una herramienta para analizar las publicaciones de usuarios de redes sociales; viene motivado por la necesidad de extraer información acerca del fenómeno social [Black Lives Matter \(#BLM\)](#).

Concretamente, en este capítulo se aplicará nuestro sistema para conocer mejor el tipo de usuarios que publicaban en el corpus, en idioma español, construido por [?] en 2020 a modo de archivo social sobre sobre las discusiones acerca de los conflictos raciales motivados por el asesinato de George Floyd. Este corpus se encuentra disponible en <https://www.dc.fi.udc.es/~david/hdh2021/>.

8.0.1 Procesado corpus

Al bajar el corpus de la dirección anterior, este viene distribuido entre múltiples archivos XML por cada *thread* o «hilo» del subreddit de BLM. Esta agrupación tiene sentido ya que cada «hilo» representa un tópico distinto y los comentarios en cada uno tienen un contexto común. Sin embargo, para nuestra finalidad de perfilar cada usuario de la colección tiene mayor importancia usar el mayor número de publicaciones por usuario para que de esta forma las predicciones tengan mayor fiabilidad.

En este sentido, se procedió a crear un único archivo CSV, con dos columnas: identificador del usuario y texto de la publicación. Este CSV contendrá todas las publicaciones de cada autor de la colección sin tener en cuenta el hilo concreto de cada una. Es importante señalar que no se agrupan las publicaciones de cada autor en una fila, sino que cada publicación de partida se escribe en una fila distinta del csv. De esta forma, al tener las publicaciones de cada autor separadas, cada algoritmo de perfilado se encarga de unirlas y preprocesarlas de la manera oportuna.

8.1 Aplicación

Posteriormente, tras haber creado este archivo CSV con el conjunto de todas las publicaciones de los autores de #BLM ya se puede subir a la aplicación para perfilar el corpus. Para ello, arrancamos la aplicación en el entorno local y accedemos desde un navegador a la dirección de *localhost* en el puerto 3000. Tras este paso se podrá ver una página de inicio como la de la figura ???. En cada figura de esta sección se mostrarán dos subfiguras con las vistas de la aplicación desde: ordenador de sobremesa (*desktop*) y móvil (*mobile*).

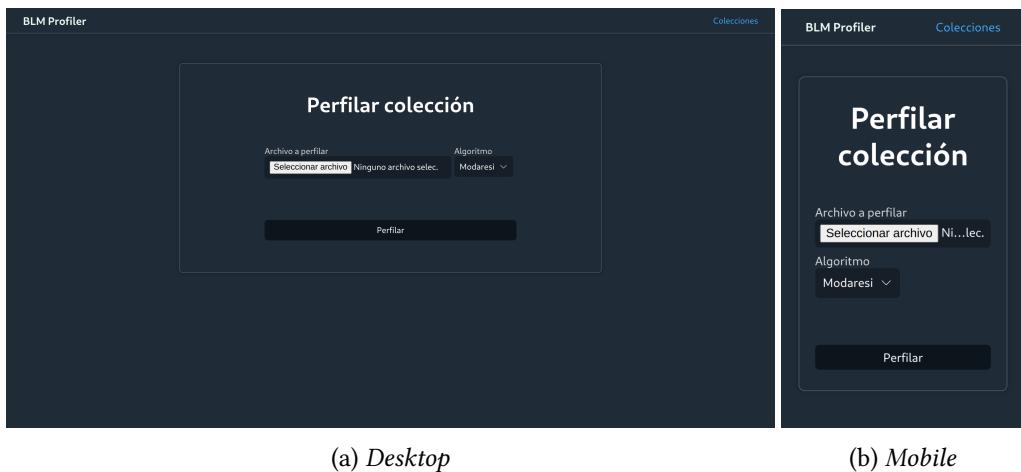


Figura 8.1: Página de inicio.

8.1.1 Perfilado del corpus

En esta página se puede ver un formulario con dos campos: uno obligatorio que es el selector del fichero que contiene el corpus a perfilar y otro opcional selector para escoger el algoritmo de perfilado deseado, que por defecto será el de «modaresi». Tras seleccionar el fichero que contiene el corpus a perfilar, que debe tener extensión .csv o .txt, se nos mostrará un mensaje en función de si este es válido o no para el perfilado¹. En función de ello nos dejará o no perfilar la colección. Tras este paso si podremos pulsar el botón de perfilar y la página mostrará un spinner a modo de carga mientras se perfila la colección como se puede ver en la figura ??.

¹ Para que el fichero sea válido debe tener al menos dos columnas llamadas *id* y *posts*, que se refieren al identificador del usuario y su publicación.

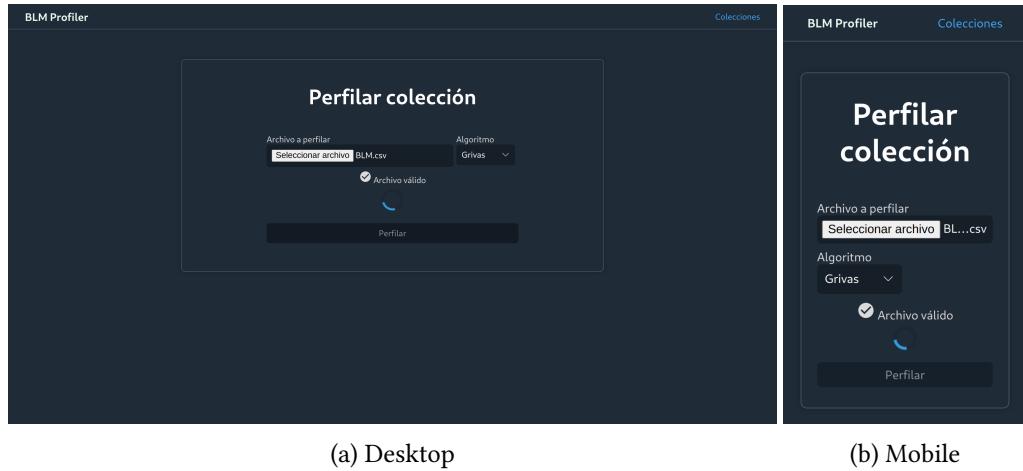


Figura 8.2: Página de inicio mientras se perfila una colección.

8.1.2 Visualización de resultados

Cuando termine el perfilado la aplicación nos redirigirá al *dashboard* o cuadro de mando de la misma que se puede ver en la figura ??.



Figura 8.3: Dashboard de la colección perfilada.

Como podemos ver en la parte de abajo tenemos una tabla y dos gráficos. En el gráfico de barras se puede ver la distribución de los usuarios del corpus según su edad, hay cuatro barras para cuatro rangos de edad distintos. En el gráfico en forma de tarta, en cambio, se muestra la distribución de los usuarios en función del género de los mismos. La tabla, por otra parte, es un listado de todos los usuarios del corpus en el que se muestra el id del mismo², el género

² Correspondiente a la columna id del archivo de subida de la colección

y edad con los que fue clasificado y una muestra de una publicación del usuario en forma de enlace, que explicaremos más adelante.

Luego, en la parte superior, se encuentran varias tarjetas que presentan detalles generales sobre la colección, tales como el tiempo de perfilado, el algoritmo utilizado, el nombre del archivo de subida y el número total de usuarios.

Filtrado de usuarios por categorías

Por último, al inicio de la página se encuentra un texto titulado «Filtrar» y debajo podemos ver dos botones. Al situar el ratón sobre cualquiera de ellos veremos como realmente son botones desplegables en los que se presentan los posibles filtros para cada categoría. Al seleccionar un filtro de alguna de las categorías veremos como se añade un elemento después de las tarjetas de la colección, que indica el filtro concreto escogido para esa categoría. En la figura ??, podemos ver el aspecto del filtro añadido.

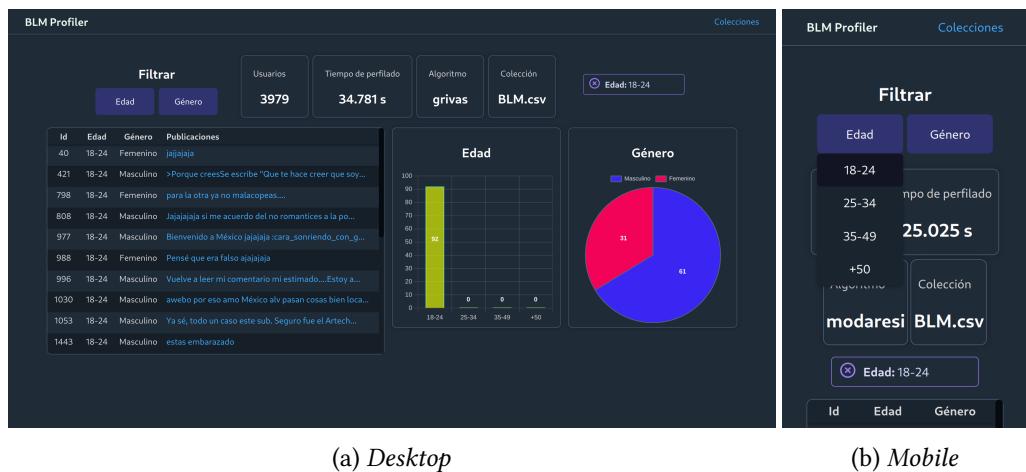


Figura 8.4: Dashboard de la colección perfilada filtrado únicamente por edad.

Como se puede apreciar en este caso, el proceso de filtrado afecta a la lista de usuarios y a ambos gráficos. Por un lado, en la lista, podemos observar que únicamente se muestran usuarios cuyas edades se sitúan en el rango de 18-24 años. En el gráfico de edad, el filtrado por una grupo de la misma categoría no es especialmente interesante, ya que no nos aporta información nueva. Sin embargo, en el gráfico de género el filtrado por edad nos permite estudiar como cambia la distribución del género de los usuarios en función del rango de edad de los mismos.

Otra opción relevante, es la de filtrar el *dashboard* por ambas categorías. Como ya hemos comentado en los gráficos nos brindaría ninguna información nueva. Por el contrario, al realizar este filtrado podemos estudiar el tipo de publicaciones que realiza un grupo demográfico

concreto lo que también nos aporta pistas sobre los criterios que se emplean en el perfilado de cada grupo. En la figura ??, se puede contemplar el filtrado en función de ambas categorías.

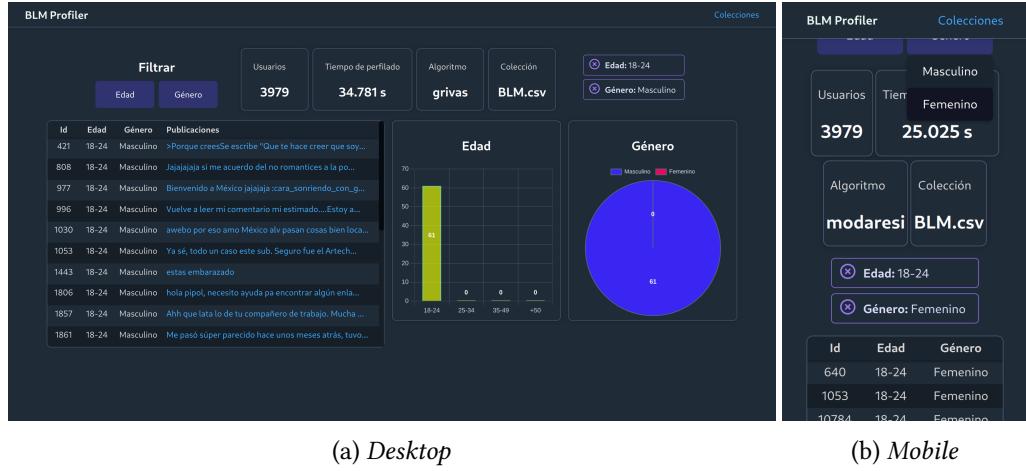


Figura 8.5: Dashboard de la colección perfilada filtrado por género y edad.

8.1.3 Publicaciones de un usuario

Por ejemplo, si deseáramos examinar el modo en que una usuaria que se encuentra en el rango de edades de 18 a 24 años redacta, tendríamos dos opciones. En la versión de escritorio, podríamos hacer clic en el enlace ubicado en la columna "Publicaciones". Mientras que en la versión móvil, podríamos lograrlo simplemente seleccionando una fila de la tabla. Al realizar esta acción, accederíamos a las publicaciones de la usuaria seleccionada en una página similar a la que se muestra en la figura de referencia. Por ejemplo, si deseáramos examinar el estilo de redacción de un usuario masculino de entre 25 a 34 años, haríamos clic sobre el enlace de la columna «Publicaciones» en la versión *desktop*, o directamente seleccionando una fila de la tabla, en la versión *mobile*. Con esta acción podríamos ver las publicaciones del usuario seleccionado en una página similar a la de la figura ??.

The figure shows two side-by-side screenshots of the BLM Profiler application. Both screens are titled 'Información de usuario' (User Information) and feature a dark blue header with the BLM Profiler logo and a 'Colecciones' button.

(a) Desktop: This version has a sidebar on the left containing three cards: 'Id' (13), 'Edad' (25-34), and 'Género' (Masculino). The main content area is a large text box containing several paragraphs of text. At the top right of this box is a 'Volver' (Back) button.

(b) Mobile: This version has a header with three buttons: 'Id' (13), 'Edad' (25-34), and 'Género' (Masculino). Below this is another 'Publicaciones' section with text, and at the top right is a 'Volver' button.

Figura 8.6: Página de la vista en detalle de un usuario de la colección.

8.1.4 Listado de colecciones

En cambio, si no deseamos perfilar una colección nueva sino que queremos ver el listado histórico de colecciones ya perfiladas, en la barra de navegación tenemos un enlace, arriba a la derecha, llamado «Colecciones» que nos llevará a una página como la de la figura ??.

The figure shows two side-by-side screenshots of the BLM Profiler application. Both screens are titled 'Colecciones' (Collections) and feature a dark blue header with the BLM Profiler logo and a 'Colecciones' button.

(a) Desktop: This version displays a table with five columns: 'Fichero', 'Algoritmo', 'Fecha', 'Usuarios', and 'Acciones'. The table contains eight rows of data, each representing a collection. The 'Acciones' column includes a 'Ver' link for each row.

(b) Mobile: This version displays a similar table structure, but the 'Acciones' column is represented by a small square icon that, when tapped, navigates to the detailed view of the collection.

Figura 8.7: Página donde se puede ver el listado de colecciones perfiladas.

En esta, tendremos una tabla con las colecciones perfiladas, ordenadas cronológicamente de más a menos recientes. También podremos ver detalles como el nombre del fichero subido, la fecha de subida y el número de usuarios de la colección y algoritmo de perfilado usado en el caso de la versión *desktop*. Asimismo en la columna «Acciones» hay un enlace llamado «Ver» que nos dirigirá a la página del *dashboard* de esa colección. En la versión *mobile* se puede navegar al *dashboard* tocando sobre la fila de la colección que queramos ver.

8.2 Análisis de resultados

Tras haber explorado el funcionamiento e interfaz de la herramienta de perfilado desarrollada, se procede a una segunda fase en la que se analizan y discuten los resultados obtenidos mediante su uso.

Esta sección se dividirá en un apartado para cada algoritmo de perfilado utilizado, así como un apartado final a modo de conclusiones generales sobre los resultados de ambos *profilers*.

8.2.1 Algoritmo de Modaresi

Vamos a comenzar viendo en detalle los resultados del perfilado mediante el algoritmo de modaresi ([?]).

En cuestión de edad, se pueden observar las distribuciones de edad obtenidas por género en la figura ?? .Lo primero que llama la atención es el gran desequilibrio de usuarios en favor del grupo de 25-34 años, el cual supone prácticamente la totalidad de los mismos, un 99.8% de la colección. Mientras que, los grupos de entre 18-24 y 35-49 solo alcanzan 9 usuarios cada uno (menos del 0.1%) y el grupo de mayores de 50 años se queda vacío. En las distribuciones de usuarios por género se mantiene en ambas la misma tendencia.



Figura 8.8: Distribuciones de edad obtenidas mediante algoritmo de modaresi [?], en corpus #BLM en español, según género de usuarios.

En cuestión de género, se puede ver en el gráfico en forma de tarta de la figura ?? que los usuarios masculinos que publicaban en #BLM constituyen casi tres cuartas partes del total de la colección (70.72%). Sin embargo, si vemos los gráficos por edades podemos darnos cuenta que este desequilibrio solo se da en el rango de edad de entre 25 y 34 años, ya que en el de 18-24 y 35-49 los usuarios femeninos representan más de la mitad en esos grupos demográficos. No obstante, al haber un desequilibrio tan grande en cuestión de edad estas mayorías no son apenas significativas comparadas con el total de usuarios.

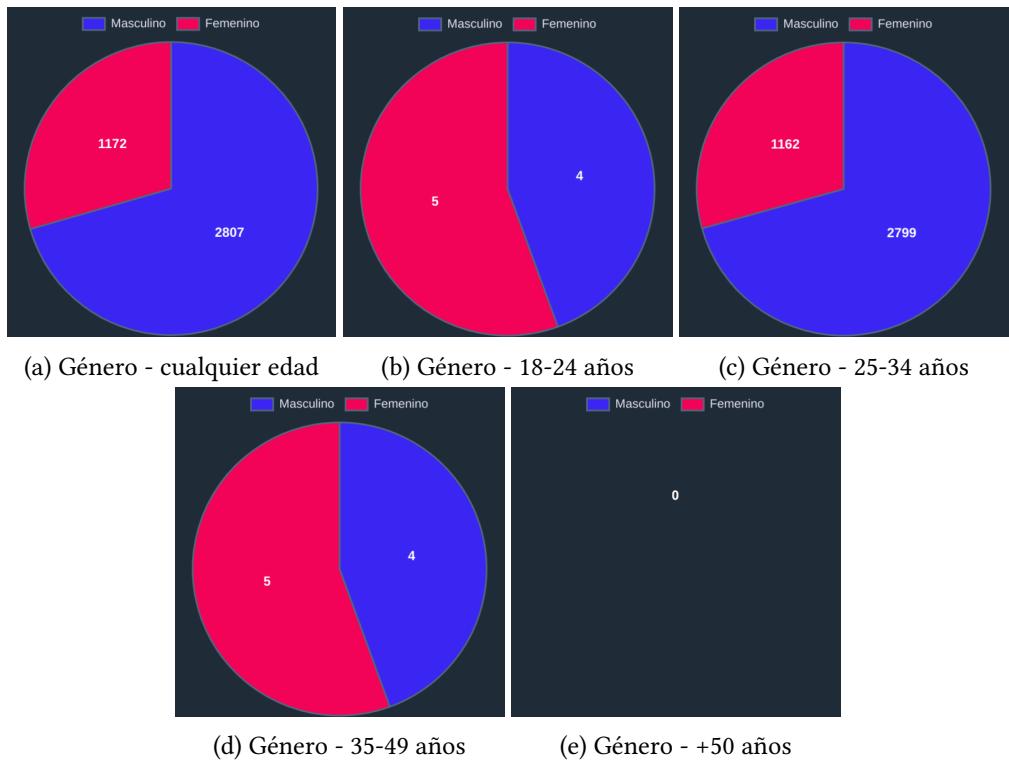


Figura 8.9: Distribuciones de género obtenidas mediante algoritmo de modaresi [?], sobre corpus #BLM en español, según rango de edad de usuarios.

En la tabla

Edad \ Género	18-24	25-34	35-49	+50	Total
Femenino	31	154	0	1	186
Masculino	61	3725	3	4	3793
Total	92	3879	3	5	3979

Tabla 8.1: Tabla resumen de los usuarios de la colección #BLM.

8.2.2 Algoritmo de Grivas

([?]).

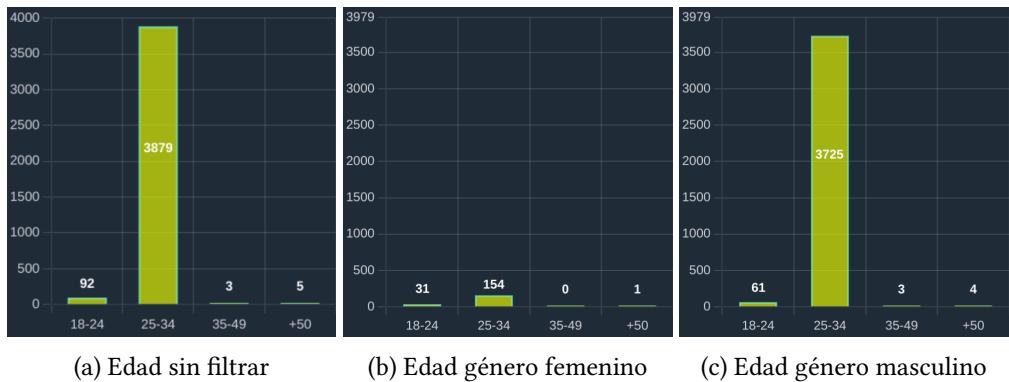


Figura 8.10: Página donde se puede ver el listado de colecciones perfiladas.

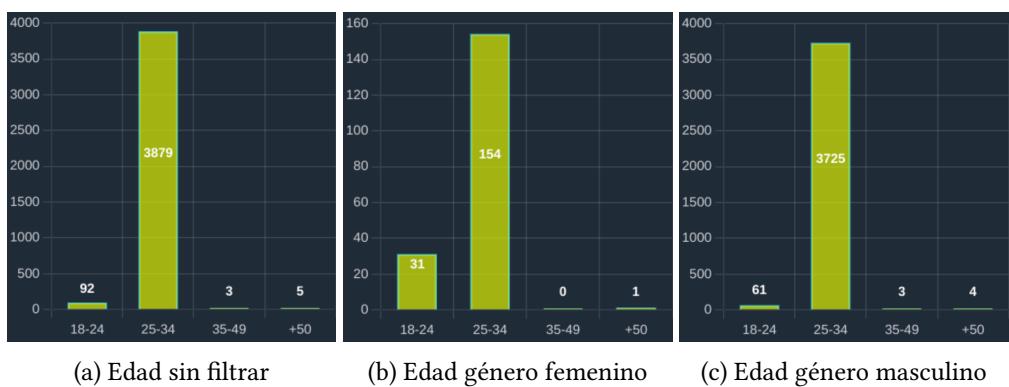


Figura 8.11: Página donde se puede ver el listado de colecciones perfiladas.

Género \ Edad	18-24	25-34	35-49	+50	Total
Femenino	5	1162	5	0	1172
Masculino	4	2799	4	0	2807
Total	9	3961	9	0	3979

Tabla 8.2: Tabla resumen de los usuarios de la colección #BLM.

Edad**Género**

Capítulo 9

Conclusións

DERRADEIRO capítulo da memoria, onde se presentará a situación final do traballo, as leccións aprendidas, a relación coas competencias da titulación en xeral e a mención en particular, posibles liñas futuras,...

Apéndices

