

Universidades de Burgos, León y
Valladolid

Máster universitario

Inteligencia de Negocio y Big Data en Entornos Seguros



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

**Herramienta *open-source* para
análisis de sentimientos en redes
sociales**

Presentado por Liviu Viorel Jula Vacar
en Universidad de Burgos — 13 de julio
de 2023

Tutor: Dr. Álvar Arnaiz González

Universidades de Burgos, León y Valladolid



Máster universitario en Inteligencia de Negocio y Big Data en Entornos Seguros

D. Álvar Arnaiz González, profesor del departamento de Ingeniería Informática, Área de Lenguajes y Sistemas Informáticos.

Expone:

Que el alumno D. Liviu Viorel Jula Vacar, con DNI dni, ha realizado el Trabajo final de Máster en Inteligencia de Negocio y Big Data en Entornos Seguros titulado “Herramienta *open-source* para análisis de sentimientos en redes sociales”.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 13 de julio de 2023

Vº. Bº. del Tutor:

D. Álvar Arnaiz González

Resumen

Las opiniones públicas que se realizan en Internet sobre diversos productos, servicios, marcas o lugares pueden influenciar el comportamiento de las personas. La gran mayoría de estas opiniones se difunden en redes sociales, foros de discusión y en los diferentes sitios web de reseñas.

El objetivo de este trabajo es emplear la información disponible públicamente para crear una herramienta que permita realizar un análisis de sentimientos sobre ciertas palabras clave o temas específicos de los que se quiera obtener información. Se realizará un proceso *ETL* (*Extract – Transform – Load*) para el procesamiento de los datos y se implementará un *dashboard* que permita visualizar y explorar la información obtenida finalmente.

Para conseguir estos objetivos, se emplearán tecnologías *open-source* para el desarrollo de la herramienta y técnicas de procesamiento de lenguaje natural para el análisis de sentimientos basado en aspectos (*Aspect-Based Sentiment Analysis, ABSA*).

Descriptores

Deep learning, procesamiento de lenguaje natural, sentiment analysis, big data, ETL, *dashboard*, visualización de datos, *open-source*.

Abstract

Public opinions made on the Internet about various products, services, brands or places can influence people's behavior. The vast majority of these opinions are shared on social media, discussion forums and on the different review websites.

The aim of this project is to make use of publicly available information to create a tool able to perform sentiment analysis on certain keywords or specific topics for which we want to obtain information. An ETL (Extract – Transform – Load) workflow will be used to process the data and a dashboard will be implemented to visualize and explore the final information obtained.

To achieve these objectives, open-source technologies will be used for the development of the tool and natural language processing techniques for aspect-based sentiment analysis.

Keywords

Deep learning, natural language processing, sentiment analysis, big data, ETL, dashboard, data visualization, open-source.

Índice general

Índice general	iii
Índice de figuras	vi
Índice de tablas	viii
Memoria	3
1. Introducción	3
1.1. Estructura de la memoria	3
1.2. Materiales adjuntos	4
1.3. Planteamiento del proyecto	5
2. Objetivos del proyecto	9
2.1. Objetivos generales	9
2.2. Objetivos técnicos	10
2.3. Objetivos personales	11
3. Conceptos teóricos	13
3.1. Big Data	13
3.2. Proceso ETL	17
3.3. Big Data Knowledge Discovery	20
3.4. Procesamiento de lenguaje natural	22
4. Técnicas y herramientas	29
4.1. Técnicas	29
4.2. Herramientas	30

5. Aspectos relevantes del desarrollo del proyecto	43
5.1. Extracción de datos	43
5.2. Carga de datos	53
5.3. Transformación de los datos	55
5.4. Visualización de los datos	57
5.5. Orquestación de los procesos	58
5.6. Interfaz web de acceso centralizado	61
6. Trabajos relacionados	63
6.1. Herramientas similares	63
6.2. Artículos científicos	65
7. Conclusiones y Líneas de trabajo futuras	67
7.1. Conclusiones	67
7.2. Líneas de trabajo futuras	69
Apéndices	71
Apéndice A Plan de Proyecto Software	73
A.1. Introducción	73
A.2. Planificación temporal	73
A.3. Estudio de viabilidad	85
Apéndice B Especificación de Requisitos	87
B.1. Introducción	87
B.2. Objetivos generales	87
B.3. Catalogo de requisitos	87
B.4. Especificación de requisitos	87
Apéndice C Especificación de diseño	89
C.1. Introducción	89
C.2. Diseño de datos	89
C.3. Diseño procedimental	112
C.4. Diseño arquitectónico	116
Apéndice D Documentación técnica de programación	119
D.1. Introducción	119
D.2. Estructura de directorios	119
D.3. Manual del programador	121
D.4. Compilación, instalación y ejecución del proyecto	132
D.5. Pruebas del sistema	136

Apéndice E Documentación de usuario	137
E.1. Introducción	137
E.2. Requisitos de usuarios	137
E.3. Instalación	138
E.4. Manual del usuario	139
Bibliografía	145

Índice de figuras

1.1.	Vista básica de la arquitectura del proyecto	6
3.2.	Diferencia entre la cantidad de datos generados por minuto entre 2013 y 2022 de algunas plataformas	14
3.3.	Cantidad de datos generados cada minuto durante el año 2021 en Internet	14
3.4.	Ilustración básica de un proceso <i>ETL</i>	18
3.5.	Arquitectura básica del sistema <i>SoMABiT</i>	21
3.6.	Evolución de las técnicas <i>NLP</i>	23
3.7.	Esquema básico de neuronas recurrentes	24
3.8.	Esquema básico de neuronas recurrentes con funciones de olvido incluidas	24
3.9.	Arquitectura del modelo <i>Transformer</i>	25
3.10.	Mecanismo de atención del modelo <i>Transformer</i>	26
3.11.	Entrenamiento y <i>fine-tuning</i> del modelo <i>BERT</i> . El pre-entrenamiento del <i>LLM</i> permite su posterior adaptación mediante transferencia de aprendizaje a nuevas tareas o conjuntos de datos	27
5.12.	Visión general de la arquitectura de <i>Airbyte</i>	48
5.13.	Configuración básica del conector original de <i>Airbyte</i> para Twitter	49
5.14.	Configuración ampliada del conector mejorado de <i>Airbyte</i> para Twitter	50
5.15.	Discrepancia en el eje temporal al utilizar los datos recientes de la <i>API</i> de Twitter y los del conjunto de datos de demostración	52
5.16.	Tareas de la <i>data pipeline</i> sobre el <i>dataset</i> de demostración	60
5.17.	Aplicación web desarrollada como punto de acceso centralizado	62
C.1.	Estructura de datos en MongoDB y nomenclatura de las colecciones	91
C.2.	Diseño inicial de la pestaña <i>Raw data</i>	104
C.3.	Primera iteración de la pestaña <i>Raw data</i>	104

C.4. Segunda iteración de la pestaña <i>Raw data</i>	105
C.5. Diseño inicial de la pestaña <i>Sentiment overview</i>	106
C.6. Primera iteración de la pestaña <i>Sentiment overview</i>	107
C.7. Diseño inicial de la pestaña <i>Sentiment analysis</i>	108
C.8. Primera iteración de la pestaña <i>Sentiment analysis</i>	109
C.9. Segunda iteración de la pestaña <i>Sentiment analysis</i>	110
C.10. Primera iteración de la pestaña <i>User analysis</i>	111
C.11. Primera iteración de la pestaña <i>Other</i>	112
C.12. Diagrama de secuencia de la interacción entre los componentes que forman parte de la etapa de extracción de datos del proceso <i>ETL</i>	113
C.13. Diagrama de secuencia de la interacción entre los componentes que forman parte de la fase de procesamiento de la etapa de transformación de datos del proceso <i>ETL</i>	114
C.14. Diagrama de secuencia de la interacción entre los componentes que forman parte de la fase de inferencia de la etapa de transformación de datos del proceso <i>ETL</i>	115
C.15. Diagrama de secuencia de la interacción entre el usuario y los componentes a nivel general de la plataforma	116
C.16. Vista general de la arquitectura de la plataforma junto a las redes <i>docker</i> a las que pertenece cada componente	117
C.17. Diagrama de secuencia de la interacción entre el usuario y los componentes a nivel general de la plataforma	118
D.1. Configuración del origen de datos en Airbyte: Twitter	123
D.2. Configuración del destino de datos en Airbyte: CSV Local	125
D.3. Configuración de la conexión entre origen y destino de los datos en Airbyte	127
D.4. Interfaz gráfica de la vista para creación y configuración de nuevos conectores para Airbyte	128
D.5. Esquema <i>YAML</i> de la vista para creación y configuración de nuevos conectores para Airbyte	129
D.6. Consumo de memoria RAM con la plataforma «en reposo»	133
E.1. Vista del <i>SQL Lab</i> para la exploración de datos de Apache Superset	141
E.2. Vista de edición de visualizaciones de Apache Superset	142

Índice de tablas

5.1. Recursos disponibles a través de la <i>API</i> de Twitter	44
5.2. Muestra de nodos disponibles en <i>Graph API</i> de Facebook	45
5.3. Muestra de nodos disponibles en <i>Graph API</i> de Instagram	46
5.4. Recursos disponibles a través de la <i>API</i> de YouTube	46
5.5. Recursos disponibles a través de la <i>API</i> de Reddit	47
C.1. Esquema de datos para la base de datos « <i>dataset</i> », colecciones « <i>airbyte_raw_*</i> »	95
C.2. Esquema de datos para la base de datos « <i>raw_twitter</i> », colección « <i>airbyte_raw_movie</i> »	98
C.3. Esquema de datos para colección « <i>clean_tweets</i> »	99
C.4. Esquema de datos para colección « <i>clean_users</i> »	99
C.5. Esquema de datos para colección « <i>clean_hashtags</i> »	99
C.6. Esquema de datos para colección « <i>clean_mentions</i> »	99
C.7. Esquema de datos para colección « <i>clean_urls</i> »	100
C.8. Esquema de datos para colección « <i>clean_annotations</i> »	100
C.9. Esquema de datos para colección « <i>clean_context_annotations</i> »	100
C.10. Esquema de datos para tabla « <i>tweets</i> »	101
C.11. Esquema de datos para tabla « <i>users</i> »	102
C.12. Esquema de datos para tabla « <i>hashtags</i> »	102
C.13. Esquema de datos para tabla « <i>mentions</i> »	102
C.14. Esquema de datos para tabla « <i>urls</i> »	102
C.15. Esquema de datos para tabla « <i>annotations</i> »	103
C.16. Esquema de datos para tabla « <i>context_annotations</i> »	103

Memoria

Introducción

Cada día se genera una inmensa cantidad de datos en Internet, esto supone una fuente de información muy útil para numerosos casos de uso. Las redes sociales ofrecen de manera pública la gran mayoría de estos en forma de opiniones de personas, lo que invita a su estudio mediante el uso de técnicas como el análisis de sentimientos.

No obstante, para llegar a obtener conocimiento a partir de todos esos datos en bruto, es necesaria la capacidad de explotarlos de manera eficiente. Este proyecto tomará como objetivo la creación de una plataforma *big data* de código abierto para el análisis de sentimientos.

En los siguientes apartados se describe el planteamiento del proyecto, la estructura de la memoria y los materiales adjuntos a la misma.

1.1. Estructura de la memoria

La memoria está organizada de la siguiente manera:

- **Introducción:** La estructuración de la memoria, los materiales adjuntos a la misma y el planteamiento del proyecto.
- **Objetivos del proyecto:** Descripción de los objetivos definidos inicialmente que se han intentado cumplir en su totalidad.
- **Conceptos teóricos:** Explicación de temas a tener en cuenta para el desarrollo del proyecto y de los modelos de *Deep Learning* empleados.
- **Técnicas y herramientas:** Que se han utilizado para la realización del proyecto y facilitar algunas labores de programación, desarrollo o gestión.

- **Aspectos relevantes del desarrollo del proyecto:** Se entra en detalle en los puntos clave que han tenido mayor importancia para la correcta realización de este proyecto. En esta sección se expondrán los desafíos presentados y los pasos llevados a cabo para solventarlos.
- **Trabajos relacionados:** Aplicaciones existentes relacionadas con el proyecto y «estado del arte» sobre técnicas de procesamiento de lenguaje natural.
- **Conclusiones y líneas de trabajo futuras:** Deducciones y resultados obtenidos a lo largo del desarrollo de este Trabajo de Fin de Máster y posibles mejoras posteriores.

Junto a la memoria, se aportan también los siguientes anexos:

- **Plan de proyecto software:** La planificación temporal del proyecto y estudio de la viabilidad económica y legal del mismo.
- **Especificación de requisitos:** Detalle de los requisitos técnicos, explicados los objetivos generales y el catálogo de requisitos.
- **Especificación de diseño:** Explicación de la arquitectura de la plataforma desarrollada y el diseño de datos.
- **Documentación técnica de programación:** La documentación técnica para la instalación, despliegue e integración de futuras funcionalidades.
- **Documentación de usuario:** Guía para utilizar el proyecto de manera segura y correcta.

1.2. Materiales adjuntos

Además de la memoria y de los anexos, se proporcionan también los siguientes materiales disponibles de manera *online*:

- Repositorio del proyecto en *GitHub*.
<https://github.com/liviuvj/sentiment-analysis-platform>
- Contenedor *Docker* con las mejoras desarrolladas del conector para Twitter de Airbyte.
<https://hub.docker.com/r/liviuvj/airbyte-source-twitter/tags>

- Vídeos sobre el funcionamiento de la plataforma:
 - Funcionamiento general de la plataforma.
...
 - Explicación detallada de la plataforma.
...
- El análisis realizado en *Google Colaboratory*.
https://colab.research.google.com/drive/1d_0bU9idFqjsDi7ezeFs1CORxTUagh7V#offline=true&sandboxMode=true
- Las particiones creadas a partir del conjuntos de datos utilizado.
 - *dataset_movie*.
<https://drive.google.com/file/d/1doLbhxFP5y4TRoMUpx6kgAIR3MUFVDVX/view>
 - *dataset_got*.
<https://drive.google.com/file/d/1sQaR19JcS1909MFqUaDK76Kz5rOo-KiL/view>
 - *dataset_season8*.
https://drive.google.com/file/d/1tSA5bGkBGfgVWdltxLEEf_d_NNm0qnUYs/view
 - *dataset_daenerys*.
<https://drive.google.com/file/d/1hL3eh3K21KtMNEkaG2JSgNSXjoNtHyTG/view>
 - *dataset_jon*.
<https://drive.google.com/file/d/1Uji4IajDAY1Aj3yhQdQ9B1NcSFg-pqrG/view>

1.3. Planteamiento del proyecto

El proyecto está formado por varios componentes, todos ellos de código abierto, que se encargan de diferentes tareas. Todos los componentes de la plataforma están completamente «*dockerizados*», es decir, empaquetados y aislados en contenedores *Docker* independientes y desplegables en cualquier máquina.

Con estas decisiones de diseño se elimina la dependencia entre componentes, creando así una arquitectura modular para la plataforma. Por lo que resulta adaptable a distintos casos de uso o a la utilización de distintos componentes a los integrados actualmente para cada parte del proyecto, creando

así una plataforma configurable según las necesidades que se planteen. En la [Figura 1.1](#) se puede observar la arquitectura planteada del proyecto.

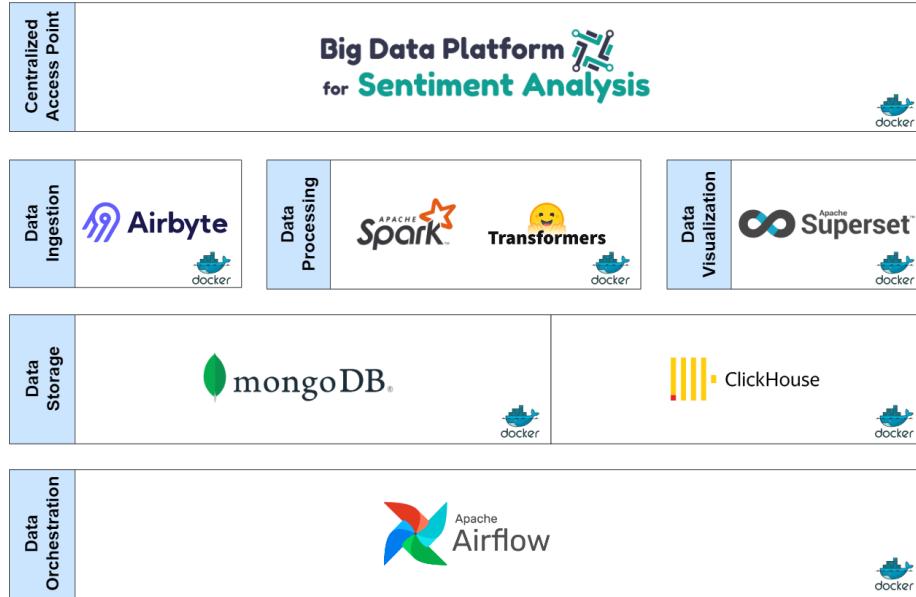


Figura 1.1: Vista básica de la arquitectura del proyecto

El primero es *Airbyte*, una herramienta que permite la extracción de datos desde distintos orígenes, como *APIs*, bases de datos o archivos disponibles en *web*. Se ha utilizado para la ingestión de datos desde *API* y desde *Google Drive*. También se ha desarrollado y mejorado el conector base de Twitter que presenta la herramienta y se ha publicado en el repositorio oficial de la misma, quedando así disponible su uso para la comunidad.

El segundo es *MongoDB*, una base de datos no relacional orientada a documentos. Se encarga de almacenar los datos en bruto de todas las fuentes de datos, actuando de esta manera como un *Data Lake*¹ para la plataforma.

El tercero es *Apache Spark*, un *framework* de computación distribuida que permite procesar grandes volúmenes de datos de forma paralela y eficiente. Su labor en el proyecto se centra en el preprocesamiento de los datos extraídos, tanto su limpieza y filtrado como el enriquecimiento mediante «metadatos».

¹Un *Data Lake* es un repositorio centralizado que almacena una gran cantidad y variedad de datos en su formato original. Esto permite a los usuarios acceder a los datos de forma flexible y rápida. Además de facilitar el descubrimiento de patrones, tendencias e *insights* [7].

El cuarto es *HuggingFace Transformers*, una librería del lenguaje de programación Python que ofrece modelos preentrenados de procesamiento del lenguaje natural (*NLP, Natural Language Processing*) basados en *Deep Neural Networks*. Esta librería se utiliza para aplicar técnicas de *NLP* sobre los datos preprocesados, desde la clasificación de sentimientos hasta la detección de entidades.

El quinto es *ClickHouse*, una base de datos columnar orientada al procesamiento analítico de datos en línea (*OLAP, On-Line Analytical Processing*) que permite realizar consultas rápidas y complejas sobre los datos. Su labor es ejercer de *Data Warehouse*² del proyecto, puesto que almacenará los datos finales enriquecidos tras las etapas anteriores.

El sexto es *Apache Superset*, una herramienta de visualización de datos que permite crear cuadros de mando interactivos y personalizados con gráficos, mapas o tablas. Su labor consistirá en permitir a los usuarios la creación y visualización de *dashboards* para explotar la información obtenida hasta el momento, como exploración de datos, métricas resultantes y análisis de sentimientos.

El séptimo es *Apache Airflow*, una plataforma de orquestación de flujos de trabajo que permite automatizar y programar tareas. Este componente se ha utilizado para gestionar las *data pipelines* diseñadas y organizar las interacciones entre los demás componentes, como el despliegue dinámico de *Spark* y *Transformers* solamente cuando resulte necesario, permitiendo así la optimización de los recursos.

El octavo es una interfaz web como desarrollo propio y a medida para este proyecto. Tiene como objetivo servir de punto de acceso centralizado a las demás aplicaciones web que ofrecen las distintas herramientas empleadas.

El planteamiento de esta plataforma ofrece una solución *big data* modular e integral para el análisis de sentimientos. La combinación de los componentes descritos ha sido elegida por formar una plataforma con tecnologías modernas, flexibles y altamente escalables que permitan explotar de manera eficiente los datos y obtener el valor esperado.

²Un *Data Warehouse* es un sistema de almacenamiento de datos que integra información de diversas fuentes y que facilita la ejecución de consultas y análisis complejos. La diferencia con una base de datos tradicional reside en que está diseñado para facilitar el procesamiento de datos analítico (*OLAP*), no transaccional [55].

Objetivos del proyecto

Este apartado explica de forma precisa y concisa los objetivos que se persiguen con la realización del proyecto. Se realiza una distinción entre los objetivos de carácter general, los de carácter técnico (propios del proyecto) y también los personales.

2.1. Objetivos generales

- Realizar extracciones de datos de dominio público.
- Llevar a cabo tareas de procesamiento, limpieza y asegurar la calidad de los datos.
- Utilizar técnicas de procesamiento de lenguaje natural para enriquecer los datos.
- Diseñar e implementar cuadros de mando interactivos sobre la información obtenida.
- Aplicar procesos de Extracción, Transformación y Carga (*ETL*).
- Diseñar y desarrollar una plataforma *open-source* para análisis de sentimiento sobre *Big Data*.

2.2. Objetivos técnicos

- Utilizar herramientas y tecnologías de código abierto.
- Emplear tecnologías distribuidas y escalables capaces de soportar *Big Data*.
- Estudiar los *frameworks* y herramientas más convenientes a utilizar en cada etapa del proyecto.
- Diseñar e implementar *data pipelines* para realizar el proceso *ETL*.
- Permitir la programación temporal de la ejecución de las *data pipelines*.
- Automatizar los procesos mediante herramientas de orquestación.
- Diseñar e implementar los *dashboard* para la toma de decisiones siguiendo las buenas prácticas recomendadas.
- Mantener la calidad de los datos extraídos y procesados.
- Diseñar la plataforma con una arquitectura modular que permita el cambio por otras tecnologías en las distintas etapas del proyecto.
- Mantener la plataforma segura con distinción de usuarios y gestión de roles y permisos.
- Diseñar el esquema de datos a utilizar y las interacciones entre los componentes de la plataforma.
- Permitir la gestión o monitorización de la plataforma mediante interfaces web.
- Realizar el despliegue del proyecto mediante contenedores *Docker*.

2.3. Objetivos personales

- Poner en práctica la mayoría de conocimientos obtenidos a lo largo del máster sobre la construcción de infraestructuras para *Big Data*, metodologías distribuidas y escalables de procesamiento de datos y *business intelligence*.
- Aprender a utilizar tecnologías modernas sobre extracción, transformación y carga de datos.
- Diseñar la arquitectura completa de una plataforma integral para ofrecer soluciones *Big Data*.
- Utilizar modelos del estado del arte sobre técnicas de procesamiento de lenguaje natural.
- Diseñar *dashboards* interactivos y llamativos.

Conceptos teóricos

En las siguientes secciones se detallarán los principales conceptos teóricos que se tratan en este proyecto. No obstante, debido a la naturaleza práctica de este trabajo, el enfoque central se desarrolla en mejor medida en el [Capítulo 4.2](#). Por esta razón, los apartados a continuación expondrán temas relevantes a la construcción de la plataforma *big data* para análisis de sentimientos desarrollada en este proyecto.

3.1. Big Data

La gran cantidad de datos que se genera diariamente a nivel mundial en Internet, y de manera pública, convierten a esta red en una inmensa fuente de información en bruto a la espera de ser procesada y explotada para la obtención de conocimiento.

En la última década, la cantidad de datos generados cada minuto ha aumentado más de un 90 % en algunas plataformas como *YouTube* o *Instagram* (véase [Figura 3.2](#)). En la [Figura 3.3](#) se puede observar el gran aumento en el volumen de los datos y la velocidad a la que se generan, teniendo en cuenta también la variedad de los distintos tipos de formato que pueden tomar en cada una de estas plataformas.

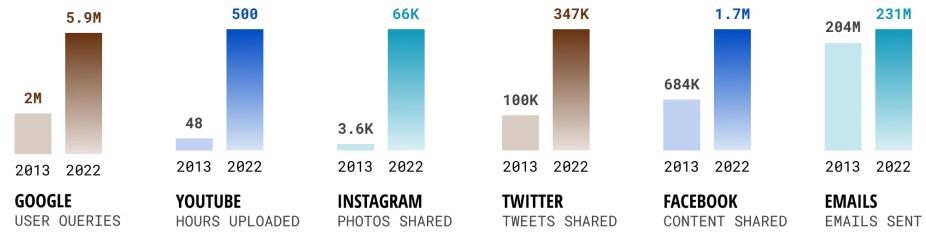


Figura 3.2: Diferencia entre la cantidad de datos generados por minuto entre 2013 y 2022 de algunas plataformas. Fuente: [24]

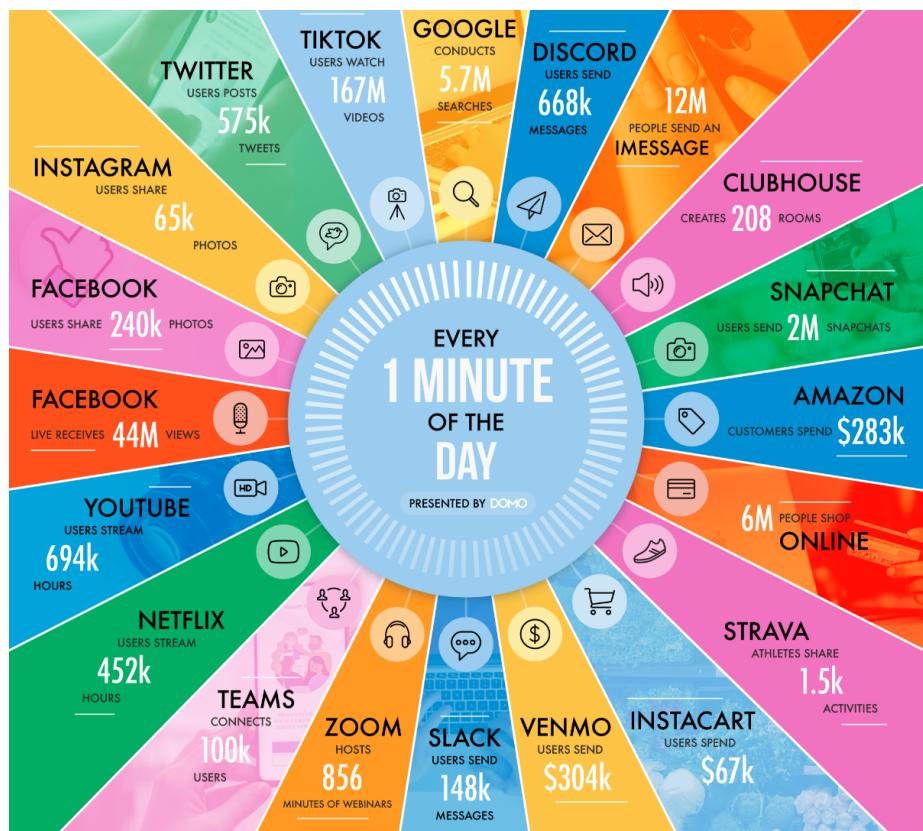


Figura 3.3: Cantidad de datos generados cada minuto durante el año 2021 en Internet. Fuente: [25]

A causa de esto, el término *Big Data* se ha convertido en un tema cada vez más recurrente e importante en la tecnología. Se trata de un fenómeno que ha generado la necesidad de evolucionar las técnicas y herramientas tradicionales a otras metodologías capaces no solo de almacenar toda esta cantidad de datos, sino también de explotarla de manera eficiente.

Características

Para entender correctamente qué es el *Big Data* [54], es necesario conocer sus características y sus diferencias respecto a los datos normales y corrientes.

Principales características

A continuación, se describen las principales características del *Big Data*, comúnmente conocidas como «Las 3 Vs del *Big Data*».

- **Volumen.** Como se ha comentado anteriormente, una de las mayores diferencias es la gran cantidad de datos que implica. En este sentido, se habla en el nivel de *terabytes* e incluso cientos de *petabytes* de datos. Estos pueden ser recogidos de distintos orígenes, como por ejemplo flujos de acciones o *clicks* de los usuarios en una aplicación web, sensores de equipos industriales en fábricas, monitorización de equipos médicos de alta sensibilidad, etc.
- **Velocidad.** Otra de las características principales es la velocidad a la que se producen o utilizan estos datos. Los sensores que monitorizan equipos de alta importancia pueden estar generando un flujo continuo de cientos o miles de registros cada segundo, siendo necesario su análisis en tiempo real para evitar consecuencias graves en ciertos escenarios o actuar con la máxima certeza posible.
- **Variedad.** Mientras que los datos tradicionales suelen estar estructurados y habituarse en gran parte al esquema de una base de datos relacional, el *Big Data* no suele adecuarse en este ámbito. Los datos son en su mayoría no estructurados o semi-estructurados, con una gran variedad de tipos (texto, audio, vídeo, ...) que pueden presentar poca densidad (lo que quiere decir que son posibles los casos en los que uno o varios atributos estén presentes en algunos registros, pero estén ausentes de casi todos los demás, aún perteneciendo al mismo flujo de datos).

Otras características importantes

Aparte de las principales propiedades que se asignaron originalmente al *Big Data*, con el tiempo fueron apareciendo más aspectos, hasta llegar a los 42 que existen en la actualidad [57]. Sin embargo, a continuación se definen otras dos características que resultan de igual importancia que las descritas anteriormente:

- **Valor.** Los datos en bruto no suelen presentar valor aparente por sí mismos. Generalmente, es necesario procesar estos datos y aplicar técnicas de análisis para poder obtener conocimiento útil. Algunas empresas o servicios de popular demanda se basan completamente en los datos de sus usuarios para poder ofrecer un valor añadido. Por ello, resulta necesario que los datos tengan el potencial de generar un valor suficiente.
- **Veracidad.** Otro de los puntos de gran relevancia en el momento actual es trabajar con datos veraces. Resulta de vital importancia conocer qué tan fiables son los datos disponibles para poder asegurar la toma de decisiones futuras a partir de los mismos.

Desafíos y oportunidades

Tras entender mejor el concepto *Big Data*, se pueden plantear una serie de desafíos al trabajar con ello que no existían previamente con los datos tradicionales. Los principales retos a superar se pueden deducir a partir de «Las 5 Vs del *Big Data*» detalladas anteriormente.

En primer lugar, es necesario disponer de la capacidad de procesar grandes volúmenes de datos en tiempo real. Como se ha indicado previamente, los datos en bruto no suelen presentar valor por sí mismos. Por ello, normalmente se realizan operaciones de integración de datos, en los que se combina información originada de distintas fuentes de datos, tras lo cual se procesan en conjunto y se aplican agregaciones, reglas de negocio o asegurando la calidad de los mismos. Esto ha provocado la necesidad de desarrollar nuevas herramientas capaces de realizar esta labor de procesamiento en tiempo real sobre grandes cantidades de datos.

También surge la necesidad de poder persistir toda la información obtenida de distintos orígenes, teniendo en cuenta las posibles diferencias entre los tipos de datos con los que se trabaja en cada uno de ellos. De esta manera, el *Big Data* ha promovido también el auge de bases de datos no relacionales que puedan ser capaces de soportar tanta variedad de datos.

Resulta de igual relevancia comentar las técnicas con las que se trabajan los datos obtenidos. Distintos tipos de datos requieren de diferentes técnicas de análisis o transformaciones alternativas para poder explotarlos correctamente. La evolución de los métodos tradicionales y la creación de nuevas metodologías de procesamiento de datos resulta de gran ayuda en la obtención de nuevos *insights*.

Finalmente, es fundamental destacar la importancia de la seguridad en el ámbito de los datos, sobre todo al tratar con información de carácter personal. En la actualidad, los datos sobre uno mismo resultan ser lo más valioso que puede poseer una persona, puesto que pueden representar en gran parte la propia identidad. Es crucial el establecimiento de una «gobernanza de datos», de manera que solamente las personas adecuadas puedan tener acceso a información privilegiada, y cada individuo al mínimo necesario para poder llevar a cabo su función.

3.2. Proceso ETL

En secciones anteriores se ha explicado cómo el *Big Data* implica un aumento del volumen, velocidad y variedad de los datos, entre otras características. Lo cual supone también la utilización de distintas fuentes de datos, que suelen incrementar de igual manera en el tiempo y a medida que surgen más requisitos o necesidades para los proyectos.

Para llevar a cabo dicha gestión, se pone en funcionamiento un proceso *ETL*, (*Extract–Transform–Load*). Este proceso tendrá como objetivo dirigir los datos en bruto desde los orígenes de datos hasta los destinos de los mismos, transformándolos por el camino según corresponda para asegurar la obtención de información valiosa y fiable [19].

Descripción del proceso

Como bien se ha indicado, este proceso presenta tres etapas específicas y necesarias. Cada una cuenta con sus propias características y requisitos, lo que lo convierten en uno de los procesos críticos al trabajar con datos. A continuación se detalla el funcionamiento de estas etapas:

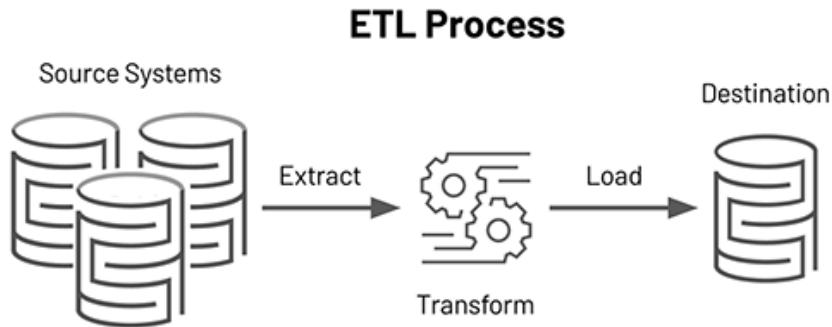


Figura 3.4: Ilustración básica de un proceso *ETL*. Fuente: [19]

- **Extracción.** La primera etapa del proceso se encarga de realizar la extracción de datos desde las fuentes de datos seleccionadas, *APIs*, bases de datos, registros de sensores, etc. Esta información extraída puede estar formada por distintos tipos de datos y estar tanto en formato estructurado como semi-estructurado o no estructurado. Además, según el funcionamiento del origen de datos, esta extracción podría ser posible realizarla de manera parcial (extrayendo únicamente registros filtrados o los modificados recientemente) o de manera total (extrayendo todos los registros, necesitando posteriormente un identificador o algún método para compararlos con los extraídos previamente y eliminar duplicados).
- **Transformación.** La segunda etapa se centra en el procesamiento de los datos en bruto mediante tareas de limpieza, transformaciones y enriquecimiento de los datos. Tras la finalización de esta parte del proceso, los datos finales obtenidos han de estar correctamente integrados y resultar fiables, almacenándose en un sistema intermedio o de *staging*. Por ello, en este punto deberán estar cumpliendo los requisitos de calidad de datos propuestos y estar disponibles para ser posteriormente utilizados o procesados y enriquecidos en mayor profundidad para fines particulares de la aplicación que los explotará en última instancia.
- **Carga.** Finalmente, la última etapa del proceso consiste en mover los datos finales, ya fiables y de calidad, al sistema o base de datos de la aplicación que los va a explotar y utilizar.

Cabe destacar también el proceso *ELT* (*Extract–Load–Transform*) [38], que actúa de manera similar al original *ETL*. En lugar de procesar los datos extraídos y mantenerlos en un área de *staging* para su posterior uso, en el proceso *ELT* los datos se cargan directamente en el sistema o base de datos destino. Por lo que cada aplicación deberá realizar sus propias transformaciones según sea necesario a partir de los datos disponibles tras realizar la carga de los mismos. El proceso *ELT* puede producir mejores resultados en conjuntos de datos no estructurados y de gran volumen.

Desafíos

La aplicación de un proceso *ETL* o *ELT* se formaliza con la creación de una *data pipeline*, un conjunto de tareas y acciones encargadas de gestionar la totalidad del proceso. El desarrollo y mantenimiento de las *data pipelines* supone numerosos retos a la hora de tratar con grandes volúmenes de datos.

Comenzando por la infraestructura necesaria para ejecutar dichos procesos, los sistemas sobre los que se desarrollan estos flujos de datos han de cumplir con las necesidades específicas de cada proyecto. Distintas tareas pueden necesitar de una serie de recursos distintos, de menor a mayor capacidad computacional o espacio de almacenamiento. Estos sistemas necesitan ser escalables para permitir la flexibilidad de mantener las *data pipelines* disponibles y en correcto estado ante cambios imprevistos en los datos o en los requisitos.

Igualmente y de manera general, al ser cada proyecto distinto y presentar diferentes requerimientos, los desarrollos realizados para un flujo de datos no tienen por qué ser extrapolables a otro proyecto. De esta manera, las aplicaciones que actúan como origen o destino de datos, y las propias herramientas que se encargan del procesamiento de los mismos, pueden estar en continuo desarrollo y evolución. Por consiguiente, surge también la necesidad de un mantenimiento continuo sobre las *data pipelines* creadas para sustentar la correcta fiabilidad de las mismas.

De igual manera, resulta importante destacar la posibilidad de cambios de contexto en los datos. Lo que puede provocar que durante cierta ventana temporal los datos extraídos puedan variar en gran medida respecto a lo establecido por defecto previamente. Esto podría no significar exclusivamente que los nuevos datos sean erróneos, sino que simplemente han modificado sus valores normales. Por ello, también será imprescindible ajustar las métricas de calidad de datos para asegurar que la información obtenida siga siendo fiable.

3.3. Big Data Knowledge Discovery

Un proyecto de esta magnitud, que resulte modular y escalable a la vez, requiere de una serie de componentes correctamente integrados e interconectados capaces de soportar grandes volúmenes de datos. Por ello, cada fase del proceso ha de estar diseñada con estos objetivos en mente, permitiendo llevar a cabo de manera eficiente las tareas de *knowledge discovery* o «descubrimiento de conocimiento» que puedan resultar necesarias para la obtención de valor.

Un antecedente sobre cómo realizar este tipo de interconexión puede darse en la arquitectura *SoMABiT* [10] (*Social Media Analysis using Big Data Technology*). Se trata de un concepto de plataforma que permite la ejecución de tareas de integración y análisis de sentimientos. En los siguientes apartados se detalla en mejor medida el concepto y la arquitectura propuesta en dicho artículo.

Concepto SoMABiT

El artículo mencionado desarrolla el concepto *SoMABiT* como una integración de distintas fuentes de datos sobre un sistema *Hadoop HDFS* que sirva como *knowledge base*, sobre el que posteriormente se posibilita la realización de consultas sobre los datos. De esta manera, se consigue crear una herramienta de ayuda a los usuarios para la toma de decisiones.

Los datos extraídos de las fuentes de datos se almacenarían en bases de datos *NoSQL* para su posterior procesamiento y enriquecimiento, persistiéndolos finalmente en la «base de conocimiento» *Hadoop*. Finalmente, los usuarios podrían realizar consultas de manera directa sobre este conocimiento, filtrándose los registros que contengan las palabras clave seleccionadas y mostrándose los datos de manera visual en un *dashboard*.

Arquitectura SoMABiT

La arquitectura básica consiste en tres componentes principales, teniendo cada uno una serie de responsabilidades y tareas concretas a llevar a cabo en el conjunto del sistema. En la [Figura 3.5](#) se puede observar el diseño empleado por los autores.

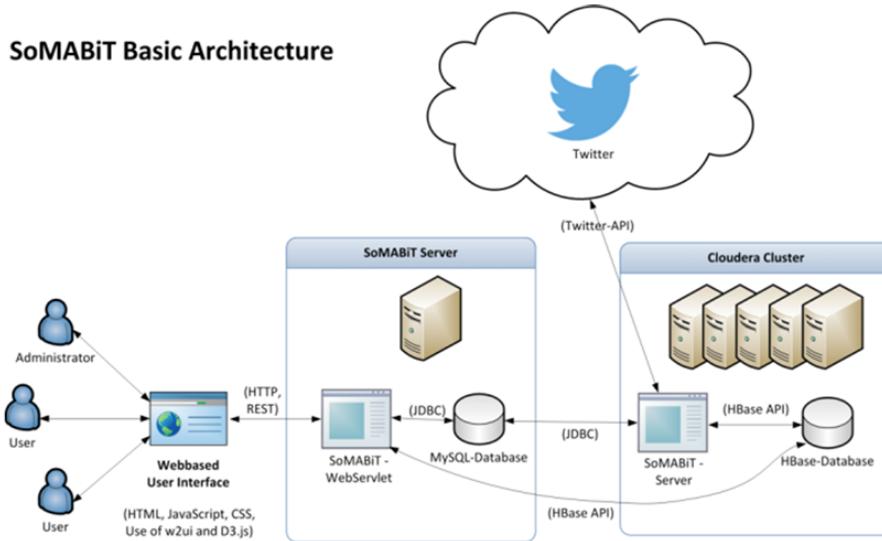


Figura 3.5: Arquitectura básica del sistema *SoMABiT*. Fuente: [10]

El sistema *SoMABiT* está formado por los siguientes componentes:

- ***SoMABiT-Server*.** Se trata de un clúster de máquinas *Cloudera* que actúan a modo de servidor para la plataforma. Este componente se encarga de recibir las configuraciones de las consultas realizadas por los usuarios y lanzar las peticiones de extracción a los distintos orígenes de datos, almacenando la información obtenida posteriormente en *HBase* y creando las tablas necesarias para ello.
- ***SoMABiT-WebServlet*.** Este servicio se encarga del procesamiento analítico de los datos y de persistir las tareas lanzadas por los usuarios en una base de datos *MySQL*, junto con información sobre la configuración de los trabajos *MapReduce* a realizar y el momento de tiempo en el que ejecutar cada uno. Se comunica con el *SoMABiT-Server* para realizar la extracción de los datos.
- ***Graphical User Interface (GUI)*.** Interfaz gráfica a la que pueden acceder los usuarios y administradores de la plataforma para interactuar con el servicio *SoMABiT-WebServlet*. Cada tipo de usuario puede realizar distintas acciones según los permisos correspondientes. Por lo que los usuarios finales puede estar accediendo únicamente a los *dashboards* de visualización de datos, mientras que usuarios más técnicos pueden estar encargándose de configurar y lanzar nuevos trabajos sobre los datos.

3.4. Procesamiento de lenguaje natural

Las técnicas *NLP* (*Natural Language Processing*) consisten en el estudio, análisis y procesamiento por máquinas del lenguaje natural. Estas tareas se llevan a cabo para que un sistema pueda comprender diversos aspectos sobre los datos y generar así un modelo de la interpretación del lenguaje con el que poder trabajar en mayor medida posteriormente.

No obstante, conseguir tal objetivo no resulta en una tarea nada trivial. Uno de los mayores desafíos en este ámbito es que el modelo empleado consiga comprender la noción del significado de una palabra en combinación con el resto de un predicado.

Por ejemplo, el siguiente enunciado se trata claramente de un sarcasmo, indicando que la persona no ha disfrutado de una obra: «¡Qué buena comedia, un poco más y me hubiese reído!». Sin embargo, algunos modelos *NLP* más simples podrían tratarlo como una buena experiencia por parte de la persona, puesto que incluye palabras como «buena», «más» y «reír» que, por lo general, se podrían asignar con sentimientos positivos.

En los siguientes apartados se indican las principales técnicas que constituyen el *state-of-the-art* en cuanto a técnicas de procesamiento de lenguaje natural.

Principales técnicas del estado del arte

Las técnicas *NLP* han ido evolucionando a lo largo del tiempo, desde tareas básicas utilizando bolsas de palabras hasta modelos más avanzados basados en *Deep Learning*.

En la [Figura 3.6](#) se puede observar la evolución histórica de estas técnicas, culminando con la tendencia en la que se está enfocando actualmente, los llamados modelos preentrenados de lenguaje o *LLM* (*Large Language Models*).

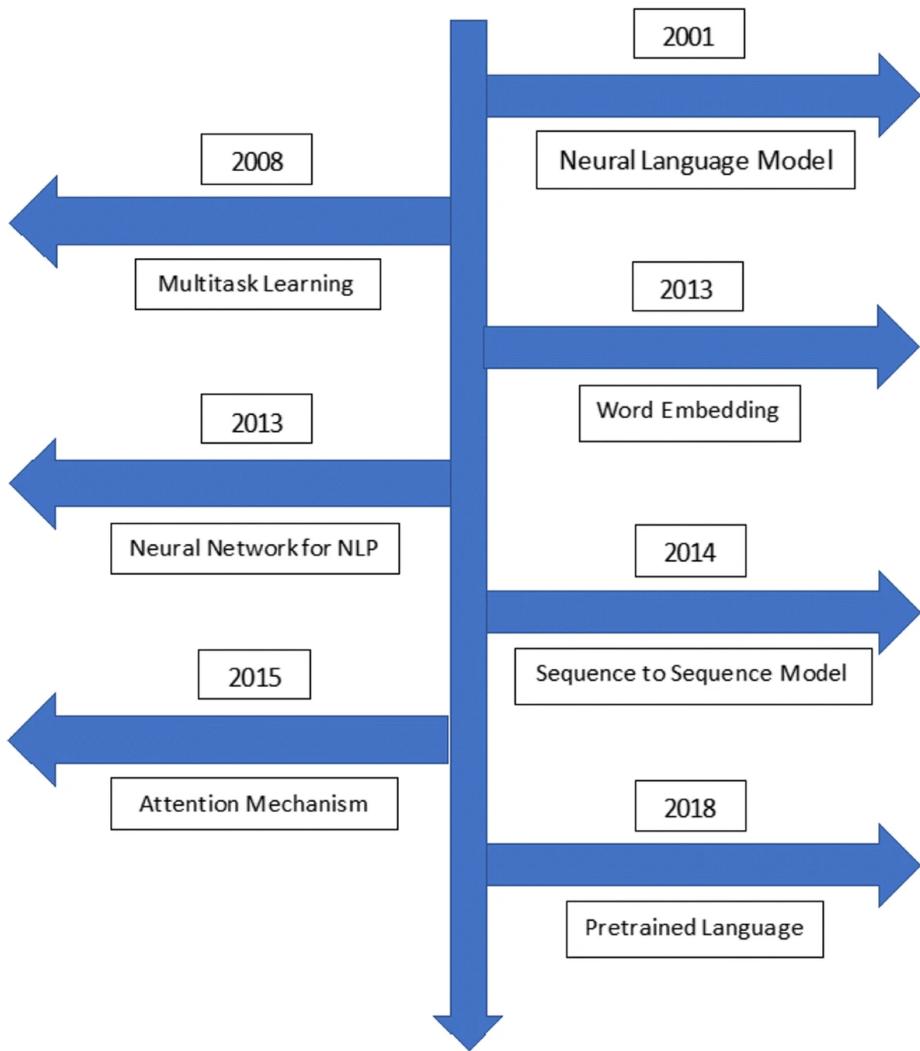


Figura 3.6: Evolución de las técnicas NLP. Fuente: [42]

A continuación, se describen algunas de las principales técnicas que han constituido el estado del arte [1]:

- ***Recurrent Neural Networks.***

Las redes neuronales recurrentes (*RNN*) son un caso particular de las redes neuronales convencionales. En estos modelos, las capas intermedias completamente conectadas presentan un segundo *output* a diferencia de las clásicas.

A parte de realizar la fase *feedforward* en la que se propagan los valores y se realiza el ajuste de pesos por las capas del modelo, la salida de

cada capa (tras pasar por las funciones de activación) vuelve a ser utilizada como entrada de la misma en la siguiente iteración. De esta manera, los resultados obtenidos de la aplicación de una *RNN* no dependen solamente de los datos de entrada actuales, sino también de los anteriormente recibidos.

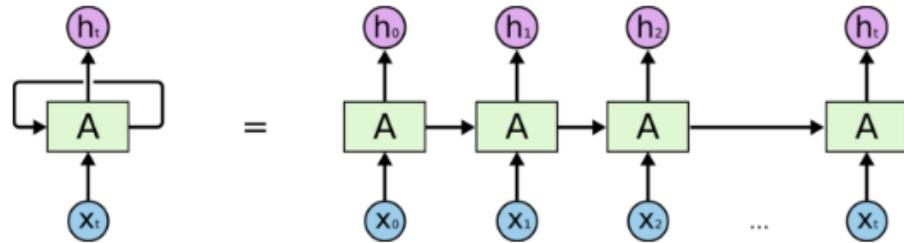


Figura 3.7: Esquema básico de neuronas recurrentes. Fuente: [1]

- ***Long Short-Term Memory Networks.***

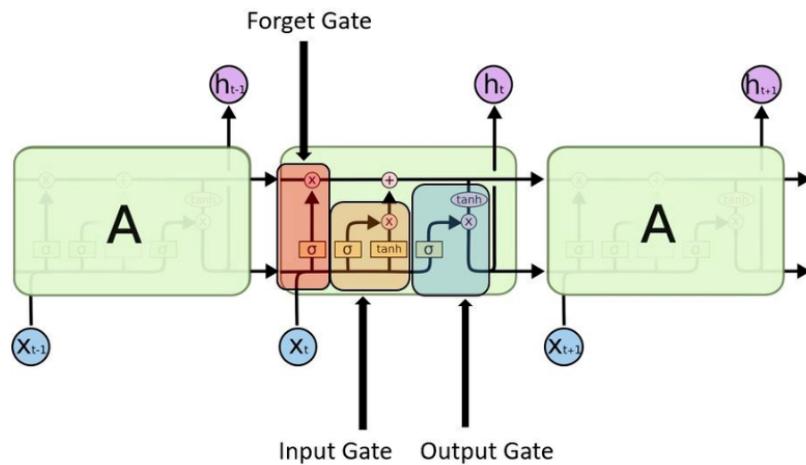


Figura 3.8: Esquema básico de neuronas recurrentes con funciones de olvido incluidas. Fuente: [1]

Estas redes neuronales son una evolución de las *RNN*. Aparte de presentar mejoras en el cálculo de los pesos, la novedad reside en un componente adicional que permite a la red «olvidar» algunos valores de lo aprendido previamente.

De este modo se consigue que el modelo siga siendo capaz de aprender de manera continua y, al no «olvidar» todo sino solamente alguna parte de lo aprendido, se obtiene también cierta capacidad de memoria.

- **Mecanismos de atención.**

Este mecanismo forma parte del modelo *Transformer* [69], que utiliza una estructura de *encoder-decoder*. El *encoder* es la parte de la red que se encarga de construir una representación continua a partir de datos de entrada en forma de símbolos para el *decoder*, y este último tiene como objetivo generar una secuencia de símbolos como respuesta a partir de la representación continua creada por el *encoder*.

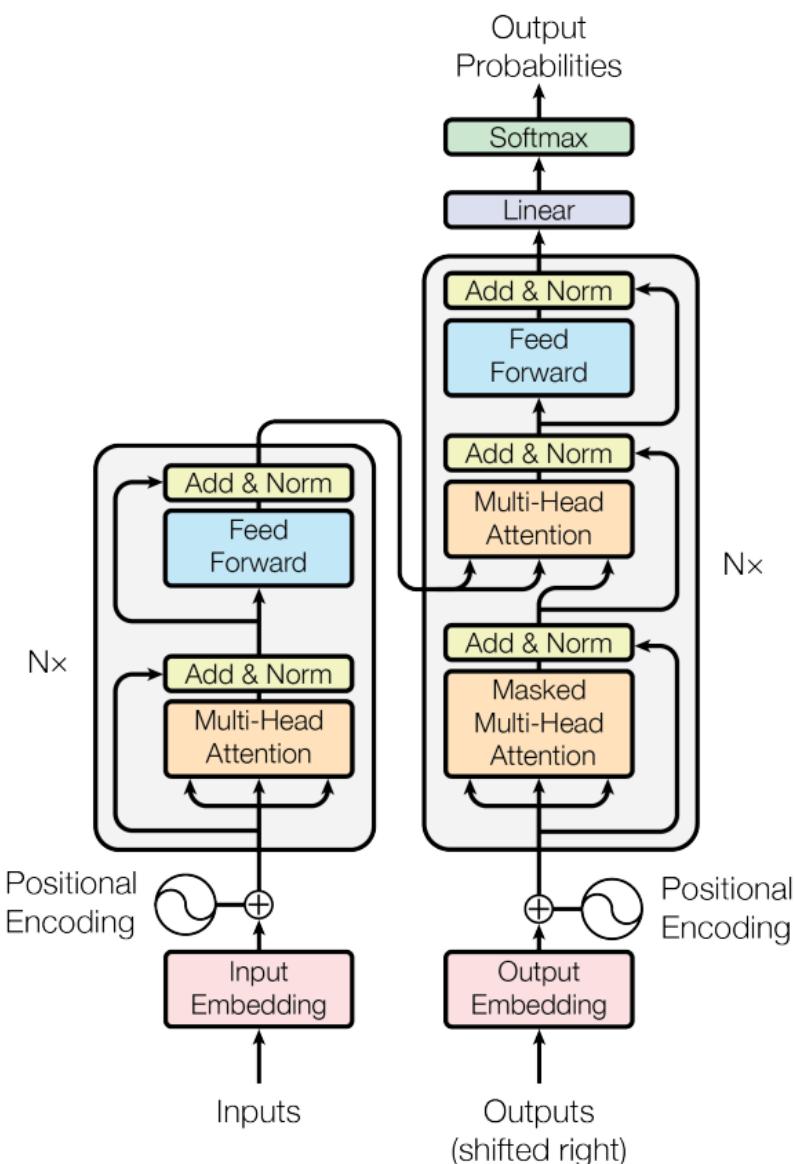


Figura 3.9: Arquitectura del modelo *Transformer*. Fuente: [69]

Como se puede ver en la [Figura 3.9](#), el modelo no utiliza *RNNs* ni *LSTMs*, pero presenta estos mecanismos de atención tanto en las capas del *encoder* como del *decoder*. El funcionamiento de estos mecanismos se basa en la idea de que, en lugar de procesar toda la secuencia de entrada de manera uniforme, se asignan pesos a diferentes partes de la secuencia.

Esto permite al modelo poner la atención en las partes más relevantes y útiles para la tarea a realizar. La ausencia de *RNNs* en el modelo también implica la eliminación de las limitaciones que estas suponían, por lo que el modelo *Transformer* se vuelve capaz de mantener una memoria mucho mayor de lo aprendido.

En la [Figura 3.10](#) se pueden observar las operaciones realizadas para calcular los pesos de atención y combinar los datos de entrada (*query*, *key*, *value*) de entrada de manera ponderada.

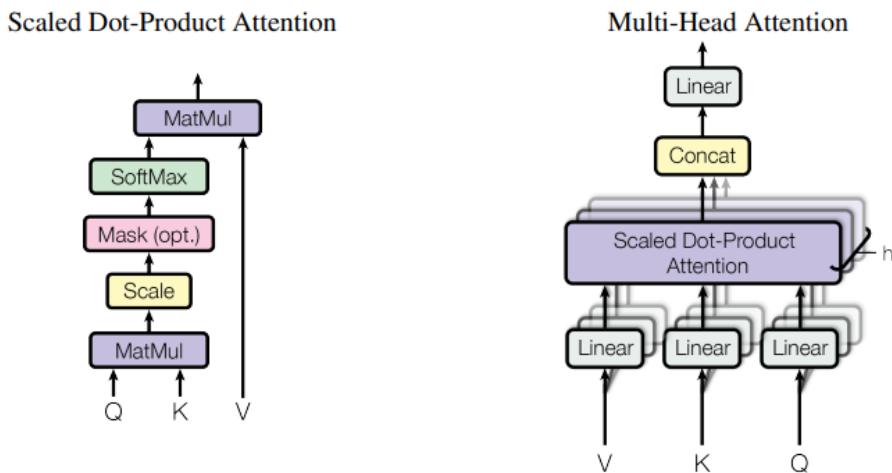


Figura 3.10: Mecanismo de atención del modelo *Transformer*. Fuente: [69]

- **BERT.**

El modelo *BERT* (*Bidirectional Encoder Representations from Transformers*) se ha convertido en la referencia base del *state-of-the-art* actual, a partir del cual se han ido desarrollando otros para tareas concretas.

Se trata de una mejora del *Transformer* en la que, en lugar de tener en cuenta solamente el contexto anterior a cada palabra, se consigue tener en cuenta el contexto posterior también, convirtiéndolo así en un modelo bidireccional.

No obstante, el mayor impacto que ha tenido *BERT* es su método de entrenamiento. Este modelo se ha entrenado de manera no supervisada intentando predecir de manera correcta la siguiente palabra de una frase.

Esto se ha conseguido «enmascarando» una palabra de cada enunciado del conjunto de datos (por ejemplo, «He entrado en la «*/MASK/*» del banco.»). Esto permite que el modelo aprenda tanto las representaciones de las palabras como las relaciones entre ellas y su significado en el contexto de cada enunciado.

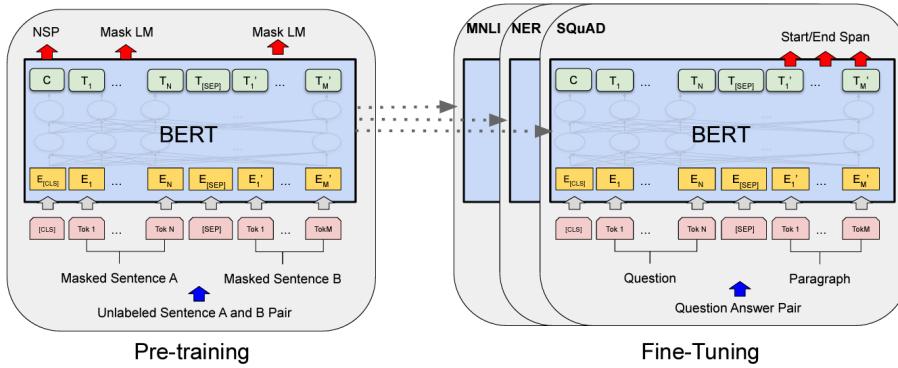


Figura 3.11: Entrenamiento y *fine-tuning* del modelo *BERT*. El pre-entrenamiento del *LLM* permite su posterior adaptación mediante transferencia de aprendizaje a nuevas tareas o conjuntos de datos. Fuente: [23]

Por estas razones, se considera un *LLM* (*Large Language Model*) debido a su entrenamiento sobre grandes volúmenes de datos y a su arquitectura, puesto que el modelo *BERT* base cuenta con 12 capas, y 12 mecanismos de atención y 110 millones de parámetros.

Este entrenamiento previo intensivo es de gran ayuda al realizar *transfer learning*, puesto que se puede adaptar a nuevas tareas simplemente añadiendo nuevas capas y realizando un entrenamiento posterior con datos etiquetados para dicha tarea.

Aplicaciones

Las técnicas de procesamiento de lenguaje natural tienen un amplio catálogo de aplicaciones en el que se pueden utilizar. Estos casos de uso han ido mejorando con el tiempo a medida que han evolucionado también las técnicas mencionadas anteriormente. No obstante, debido a las restricciones

tecnológicas o computacionales de algunos proyectos, no siempre resulta posible utilizar los últimos avances en este campo.

A continuación, se listan algunas de las aplicaciones de las técnicas *NLP*:

- **Clasificación de texto.** Consiste en categorizar el texto mediante etiquetas preestablecidas. Algunas de estas tareas pueden ser la clasificación de tópicos de un texto o el análisis de sentimientos, que se centra en determinar aspectos como el sentimiento (positivo, neutro o negativo), el sarcasmo o la actitud emocional (alegre, triste, enfadado) de la persona que ha proporcionado un enunciado [27, 43].
- **Traducción automática.** Trata la conversión de un texto en un idioma origen a un idioma destino [72].
- **Filtros de *spam*.** Identificación de mensajes y correos electrónicos no deseados [58].
- **Extracción de información.** Selección automática de información específica y extracción de fragmentos relevantes [44].
- **Elaboración de resúmenes.** Generación de la síntesis de un texto de manera concisa y coherente [56].
- **Sistemas de diálogo.** Como pueden ser los asistentes virtuales o los *chatbots*, que utilizan el intercambio de lenguaje natural entre máquinas y humanos para realizar acciones específicas [15].
- **Medicina.** Análisis de registros clínicos, información sanitaria y datos médicos [16].

Técnicas y herramientas

Esta sección de la memoria tiene como objetivo presentar las técnicas metodológicas y las herramientas de desarrollo que se han utilizado para llevar a cabo el proyecto. En el caso de algunas de estas herramientas se estudiarán diferentes alternativas, en las que se incluirán comparativas entre las distintas opciones y una justificación de la elección realizada.

4.1. Técnicas

En este apartado se hará una breve descripción sobre las técnicas empleadas a lo largo del proyecto.

SCRUM

Es un proceso de desarrollo software ubicado dentro de las metodologías ágiles. Consiste en segmentar un proyecto en varios requisitos que se han de cumplir y posteriormente subdividir estos en tareas. El desarrollo se realiza mediante *sprints*, iteraciones incrementales de normalmente dos semanas de duración, en los que se planifican las tareas a realizar durante dicho periodo.

Procesamiento de Lenguaje Natural

El término *NLP* (*Natural Language Processing*) se refiere al conjunto de métodos, dentro de la inteligencia artificial, que trabajan con recursos textuales o sonoros. Se ponen en práctica metodologías de estadística, lingüística y *machine learning* para permitir crear programas que puedan interpretar dicho tipo de información [17].

Análisis de sentimientos

El *sentiment analysis* es una técnica en la que se busca identificar y extraer información subjetiva a partir de recursos textuales. Las principales maneras de realizar este tipo de análisis suelen seguir dos rutas.

La primera, utilizando reglas y diccionarios de palabras a las que se les asignan distintas puntuaciones según el sentimiento asociado a cada palabra. La segunda, y la que mejores resultados proporciona actualmente [51], emplea técnicas de *NLP* para extraer características de los datos y comprender el contexto de la información proporcionada. Esto permite realizar clasificaciones y predicciones más acertadas ya que el resultado no se limita simplemente a un subconjunto de palabras, sino al sentido que se les da a las mismas también.

4.2. Herramientas

Para llevar a cabo este proyecto, se ha utilizado el siguiente conjunto de herramientas.

GitHub

Para el *hosting* del repositorio se ha utilizado *Github*³, puesto que ya se tenía experiencia en el uso de esta plataforma. Permite realizar la gestión del control de versiones a lo largo del desarrollo del software y simplifica el seguimiento de las tareas. Posee capacidades para creación de procesos de integración continua y despliegue continuo (*CI/CD*), automatización de flujos de trabajo, seguimiento y gestión de proyectos.

ZenHub

Para facilitar el trabajo de la gestión del proyecto se ha utilizado *ZenHub*⁴. Es una plataforma centrada en mejorar la productividad de los equipos de desarrollo, que permite llevar a cabo la planificación del proyecto, realizar un seguimiento del progreso y calcular métricas de productividad mediante gráficas.

Se ha elegido esta herramienta ya que, además de permitir realizar toda la gestión del proyecto, cuenta con una extensión web desde la que se puede

³<https://github.com/>

⁴<https://www.zenhub.com/>

acceder al panel de control directamente desde el propio repositorio de GitHub. Por lo que todas las operaciones de planificación de tareas se llevan a cabo desde el mismo lugar y facilita el trabajo del desarrollador.

Entorno de desarrollo integrado

Un *Integrated Development Environment (IDE)* es, como indica el propio nombre, un conjunto de herramientas que componen un espacio de trabajo completo para desarrollar *software*. Suele estar compuesto de las herramientas necesarias para editar, compilar, ejecutar y probar código, facilitando así la labor del desarrollador.

Herramientas consideradas:

- **Spyder:** Entorno de desarrollo *open-source* especializado en la explotación de datos y el análisis científico.
- **Visual Studio:** Herramienta que permite realizar todas las tareas de programación, depuración, pruebas y desarrollo de soluciones para cualquier plataforma.
- **Visual Studio Code:** Versión más ligera y personalizable de Visual Studio.

Herramienta elegida:

- **Visual Studio Code**⁵

Es el IDE elegido para llevar a cabo el desarrollo de proyecto. Como ventajas principales, presenta un tamaño reducido de instalación respecto a las otras opciones y permite la configuración y ejecución de tareas, además de la capacidad para instalar y personalizar nuevas funcionalidades mediante sus extensiones.

Extensiones utilizadas

Se han escogido una serie de extensiones del *Marketplace* que presenta la herramienta para facilitar la calidad de vida al trabajar con este IDE.

⁵<https://code.visualstudio.com/>

- **Python:** Extensión principal para dar soporte al lenguaje de programación Python para el correcto desarrollo de código (*linting*, formato de código, exploración de variables, depuración, etc.).
- **Python Docstring Generator:** Facilita y asiste en la creación de comentarios tipo *docstring* para funciones en Python.
- **Pylance:** Servidor de lenguaje que añade soporte adicional a Python.
- **Trailing Whitespace:** Resalta y recorta los espacios en blanco sobrantes.
- **Visual Studio IntelliCode:** Emplea IA para añadir desarrollo predictivo y autocompletado de código.
- **Docker:** Facilita la creación y gestión de contenedores a través del IDE.

Editor L^AT_EX

La elaboración de esta memoria está basada en la plantilla L^AT_EX provista como ejemplo por los Coordinadores del Máster y disponible públicamente⁶. Para facilitar la edición y gestión de esta plantilla, se ha decidido utilizar una herramienta adecuada para ello.

Herramientas consideradas:

- **MiK^TE_X + T_EX:** Herramientas que realizan la traducción de L^AT_EX a *PDF* y permiten gestionar y editar este tipo de archivos, respectivamente.
- **Overleaf:** Plataforma en línea que facilita la gestión y edición de documentos con formato L^AT_EX.

Herramienta elegida:

- **Overleaf**

Overleaf es un editor en línea⁷ de L^AT_EX. Para utilizarlo no es necesario realizar la instalación de ningún componente, tiene documentación integrada

⁶https://github.com/bbaruque/plantillaTFM_MUINBDES.git

⁷<https://es.overleaf.com/>

para L^AT_EX y permite la visualización de los cambios realizados en tiempo real, además de contar ya con los paquetes más utilizados.

También resulta más cómodo al tratarse de una plataforma *online*, ya que tan solo hace falta disponer de un navegador y conexión a Internet para poder trabajar con ella desde cualquier equipo. Otra de las mejores funcionalidades que ofrece es la posibilidad de comprobar el histórico de los archivos modificados y realizar un *rollback* de los mismos.

Se ha utilizado esta herramienta para elaborar la memoria y los anexos en L^AT_EX.

Joplin

A lo largo de la duración del proyecto hará falta tomar notas de diversos temas. Para facilitar esta tarea, se ha utilizado *Joplin*⁸. Es una plataforma de código abierto que permite gestionar apuntes y notas en forma de *notebooks*.

Entre las principales características que ofrece se encuentra la total privacidad de los datos, la sencilla interfaz que presenta, la facilidad de uso gracias al lenguaje *Markdown* y la sincronización de contenido entre diversos equipos.

Se utilizará principalmente para dejar constancia de los temas comentados durante las reuniones y apuntar información relevante para el proyecto que se vaya encontrando a medida que se desarrolle este trabajo.

Super Productivity

La gestión del tiempo dedicado se ha llevado a cabo mediante la herramienta de código abierto *Super Productivity*⁹. Sus principales funciones consisten en realizar la planificación, seguimiento y gestión de tareas. Permite distribuir tareas a lo largo de diversos proyectos, la asignación de etiquetas personalizadas y tener constancia del tiempo estimado y dedicado para cada una.

Presenta una interfaz sencilla de utilizar y amigable para el usuario que agiliza el trabajo gracias a la utilización de atajos de teclado. Otra de las características más importantes que tiene esta herramienta es la integración con varias plataformas para la importación de tareas. Por lo que la planificación realizada en GitHub y ZenHub se puede extraer a esta

⁸<https://joplinapp.org/>

⁹<https://super-productivity.com/>

herramienta y realizar un mejor seguimiento del tiempo empleado en cada una de ellas.

Docker

El despliegue de todos los componentes del proyecto se ha llevado a cabo en su totalidad mediante contenedores *docker*¹⁰. Se trata de una plataforma de desarrollo, distribución y despliegue que permite separar las aplicaciones de la infraestructura en la que se ejecutan. Para facilitar la gestión de los contenedores, se ha empleado el programa *Docker Desktop*¹¹.

Para ello utiliza contenedores o *dockers*, que son entornos «autocontenidos» que pueden ser ejecutados en paralelo y aislados del resto de procesos del sistema. Cada *docker* tiene una imagen asociada, que se puede distribuir, ejecutar y replicar de manera sencilla para un despliegue rápido y exacto.

Postman

Es una plataforma¹² para construir y utilizar *APIs* que simplifica el desarrollo y la colaboración. Cuenta con una versión web y una aplicación de escritorio, además de un repositorio público de colecciones de *APIs* y documentación sobre los posibles tipos de llamadas que se les puede realizar.

Esta herramienta será la utilizada para llevar a cabo una inspección inicial de los distintos *endpoints* que presentan las *APIs* investigadas en la sección anterior. Un ejemplo concreto de ello podría ser el siguiente *workspace* de Twitter [62], en el que se presentan documentadas las posibles llamadas a realizar.

Herramienta de extracción de datos

Para facilitar la labor de ejecución de consultas contra las *APIs* de las distintas plataformas web, se ha decidido emplear librerías de código ya habilitadas para ello.

API Wrappers

Las interfaces estudiadas en la sección anterior tienen un gran número de usuarios, lo que ha conllevado a la creación de distintas librerías o paquetes

¹⁰<https://docs.docker.com/>

¹¹<https://www.docker.com/products/docker-desktop/>

¹²<https://www.postman.com/>

en algunos lenguajes de programación que «envuelven» y «atacan» los *endpoints* de dichas *APIs*. Estos paquetes tienen el objetivo de facilitar al usuario la tarea de crear las consultas y consumir los recursos provenientes de dichas peticiones.

Estas librerías iban a ser utilizadas inicialmente para construir un primer prototipo para esta etapa de extracción de datos. Más concretamente, se utilizarían las siguientes:

- **Python Twitter Search API.** Cliente en lenguaje *Python* enfocado en utilizar los *endpoints* de búsqueda de *tweets* [64].
- **Python-Facebook.** Una librería simple de *Python* que simplifica el uso de la *Graph API* de Meta, dando soporte tanto para Facebook como para Instagram [39].
- **PRAW: The Python Reddit API Wrapper.** Paquete de *Python* que facilita el acceso a la *API* de Reddit [9].

No obstante, tras comenzar a trabajar con ellas se observaron fallos de dependencias y partes *deprecadas*. Esto originó inconsistencias entre las versiones de las *APIs* actuales y el código de dichas librerías, además de provocar contratiempos en la evaluación del *sprint* correspondiente. Por estas razones, se terminó descartando esta opción y utilizando la que se propone a continuación.

Herramienta elegida: Airbyte

Esta plataforma¹³ permite realizar la creación de un *pipeline* de extracción y guardado de datos de forma sencilla y rápida. Presenta una versión de código abierto y distribuida en contenedores *docker*, junto a una interfaz web que simplifica el proceso de gestión.

Permite la creación, definición y configuración tanto de fuentes de datos como de destinos de los mismos. Actualmente cuenta con más de 300 conectores disponibles para distintas aplicaciones e interfaces web [4].

Esta herramienta es la que se ha terminado utilizando en la fase de prototipado para la etapa de ingestión de datos de la herramienta final desarrollada.

¹³<https://airbyte.com/>

Herramienta de carga de datos

Los datos adquiridos mediante la herramienta de extracción de datos han de ser cargados en alguna herramienta y poder ser consultados posteriormente cuando sea necesario. Además, el sistema seleccionado para esta tarea no puede utilizar un esquema de datos estricto ya que se está trabajando con datos no estructurados.

Para cumplir con estos requisitos, se ha investigado viabilidad de las siguientes opciones.

Herramientas consideradas:

- **Apache HDFS (Hadoop Distributed File System).** Sistema de ficheros distribuido escalable horizontalmente, tolerante a fallos y de alto rendimiento para procesar grandes conjuntos de datos. Complejo de configurar correctamente para conseguir la eficiencia óptima [29].
- **Apache Hudi.** Plataforma *data lakehouse open-source* de procesamiento de datos tanto en *streaming* como en *batch*. Permite ingestión de datos eficiente, versionado y actualización de datos, con soporte para transacciones *ACID*. Actualmente aún presenta poca documentación y soporte, además de requerir bastantes recursos *hardware* para conseguir un rendimiento óptimo [30].
- **Apache Cassandra.** Base de datos *NoSQL* columnar enfocada al procesamiento de grandes conjuntos de datos con alto rendimiento y tolerante a fallos, con gran eficiencia para escritura y replicación de datos. Presenta soporte limitado para realizar transacciones complejas y requiere definir previamente el esquema de los datos a cargar para conseguir el rendimiento óptimo [31].
- **MongoDB.** Base de datos *NoSQL* orientada a documentos escalable y flexible que permite la ingestión y consulta eficientes de estructuras de datos complejas en formato *JSON*. Presenta soporte limitado para modelado de datos relacional y para transacciones complejas [53].

Herramienta elegida:

- **MongoDB¹⁴**

Se ha seleccionado esta herramienta por la sencilla integración y facilidad de uso que presenta para el caso de uso concreto de este proyecto. Los datos extraídos procedentes de las *APIs* seleccionadas se presentan en su totalidad en formato *JSON*, por lo que se pueden cargar directamente en esta base de datos sin realizar ninguna transformación intermedia.

MongoDB Compass

Esta herramienta [52] sirve de interfaz gráfica de usuario que permite explorar, analizar y manipular los datos almacenados en la base de datos no relacional y orientada a datos MongoDB. MongoDB Compass¹⁵ Permite visualizar la estructura y el contenido de las colecciones, crear índices, realizar consultas, modificar documentos, entre otras tareas. Además, facilita el trabajo al evitar tener que utilizar la *shell* propia de la herramienta.

Ha resultado de gran utilidad para la exploración de los datos en bruto de este proyecto ya que los datos obtenidos mediante la herramienta de extracción están en formato *JSON*, lo cual se asimila a la perfección con el formato *BSON* empleado por MongoDB. Ofrece una interfaz intuitiva para la realización de consultas sobre los datos.

Apache Spark

Apache Spark¹⁶ es un sistema de procesamiento de datos distribuido y de código abierto [35]. Spark es capaz de procesar grandes volúmenes de datos de manera rápida, eficiente y escalable.

Permite procesar datos en memoria, lo que lo hace más rápido que otros sistemas de procesamiento de datos como Hadoop MapReduce, que requieren que los datos se escriban en disco entre las operaciones. Sus principales características se centran en el rendimiento, escalabilidad, facilidad de uso y flexibilidad.

¹⁴<https://www.mongodb.com/>

¹⁵<https://www.mongodb.com/products/compass>

¹⁶<https://spark.apache.org/>

HuggingFace Transformers

*HuggingFace Transformers*¹⁷ es una librería del lenguaje de programación *Python*. Esta facilita el uso y entrenamiento de modelos pre-entrenados pertenecientes al *state-of-the-art* de las técnicas *NLP* (*Natural Language Processing*) [26].

Este paquete proporciona una interfaz unificada y sencilla para cargar, entrenar, evaluar y desplegar estos *LLM* (*Large Language Models*). El uso que se le dará a esta herramienta consistirá en la carga e inferencia de diversos modelos *NLP* pre-entrenados para tareas de análisis de sentimiento y de emoción, clasificación de tópicos y entidades.

Base de datos para procesamiento analítico de datos en línea

Los datos extraídos y los inferidos a través de los modelos de procesamiento de lenguaje natural empleados han de ser persistidos posteriormente para su consiguiente explotación de manera visual. Por tanto, para llevar a cabo esta tarea de forma óptima, resulta necesario emplear una base de datos enfocada al procesamiento analítico en línea (*OLAP*) de los datos, en lugar de emplear modelos enfocados al procesamiento de transacciones en línea (*OLTP*) comunes.

Herramientas consideradas:

- **ClickHouse.** Sistema gestor de base de datos *OLAP*¹⁸ basado en arquitectura *share-nothing*, compresión de datos y posibilidad de consultas en lenguaje SQL nativo [18]. Utiliza un concepto de vistas materializadas que se gestionan automáticamente para asegurar un rendimiento óptimo de las consultas en cualquier momento.
- **Apache Druid.** Base de datos *OLAP*¹⁹ distribuida, escalable y auto-balanceada [28]. Al contrario que ClickHouse, presenta una arquitectura modular en la que las consultas, datos y nodos de almacenamiento se encuentran separados unos de otros. A pesar de esto, realiza una gestión óptima de datos en *streaming* y permite priorizar consultas.

¹⁷<https://huggingface.co/docs/transformers/>

¹⁸<https://clickhouse.com/>

¹⁹<https://druid.apache.org/>

Herramienta elegida:

- ***ClickHouse***

La herramienta *ClickHouse* se ha elegido teniendo en cuenta los requisitos de este proyecto concreto. De esta manera, permitirá una baja latencia y un buen rendimiento para las consultas realizadas. Además, la gestión y configuración no resultará tan complicada como la exigida por *Apache Druid*, que necesita de varios servicios y componentes para funcionar correctamente. Por lo que el coste computacional y de mantenimiento sería mayor en comparación con la herramienta elegida.

Herramienta analítica de visualización de datos

Una vez que se tiene todos los datos extraídos, procesados, transformados e inferidos persistidos en el sistema *OLAP*, se encuentran ya disponibles para su explotación visual. Para ello, es necesario utilizar una herramienta de visualización compatible con el sistema *OLAP* empleado, capaz de explotar visualmente y de manera eficaz los datos obtenidos para asegurar un buen entendimiento de los mismos.

Herramientas consideradas:

- ***Metabase*.** Herramienta de exploración y visualización de datos²⁰ rápida, ligera y con una interfaz sencilla. Permite la creación de gráficos simples a través de la formulación de preguntas interactivas. Posee numerosos recursos de visualización de datos y conectores para las principales bases de datos.
- ***Apache Superset*.** Plataforma de visualización y exploración de datos moderna²¹ e intuitiva que permite la gestión de *dashboards*, roles y consultas asíncronas. Ofrece gran cantidad de recursos tanto para visualización de datos como conectores a bases de datos.

²⁰<https://www.metabase.com/>

²¹<https://superset.apache.org/>

Herramienta elegida:

- ***Apache Superset***

La herramienta *Apache Superset* [37] presenta todas las características necesarias para una correcta gestión y visualización de los datos. Permite la asignación de roles y permisos de usuario, gestión de *dashboards* y gráficos individuales, construcción de consultas tanto en lenguaje *SQL* nativo como de manera interactiva y visual, más de 40 tipos de visualizaciones distintas y más de 30 conectores para distintas bases de datos.

Herramienta de orquestación de procesos

Debido a la compleja arquitectura que presenta este proyecto, es necesario un método para gestionar todo el flujo de acciones que se lleva a cabo a través de los distintos componentes. Para ello, es necesario utilizar una herramienta que permita la orquestación de todos los procesos a ejecutar.

Herramientas consideradas:

- ***Apache Airflow***. Plataforma de orquestación²² que permite autorizar, programar y monitorizar flujos de trabajo de manera programática. Utiliza una gestión de flujos de trabajo mediante grafos acíclicos dirigidos (*DAGs*) que se pueden observar en funcionamiento a través de una sencilla interfaz web. Presenta un amplio y extensible catálogo de integraciones con distintos tipos de tareas preestablecidas.
- ***Luigi***. Paquete de Python²³ capaz de construir flujos de trabajo complejos con datos en *batch*. Gestiona la resolución de dependencias, *workflows*, visualización y fallos. Tiene soporte para el ecosistema *Hadoop* y funcionalidades para integrar diversas tareas en un solo *pipeline*.

²²<https://airflow.apache.org/>

²³<https://github.com/spotify/luigi>

Herramienta elegida:**■ *Apache Airflow***

La herramienta *Apache Airflow* [36] presenta una arquitectura modular y una cola de mensajes para la gestión de los nodos, lo que facilita su escalabilidad en situaciones necesarias. Además, permite la creación de *data pipelines* de manera dinámica y programática, lo que mejora la definición de flujos de trabajo y su parametrización. La aplicación web que presenta es otro punto a favor, ya que permite monitorizar, programar y gestionar *pipelines* de una manera sencilla.

Aspectos relevantes del desarrollo del proyecto

Este apartado pretende recoger los aspectos más interesantes del desarrollo del proyecto, además de la experiencia práctica adquirida durante la realización del mismo con las diversas tecnologías empleadas.

5.1. Extracción de datos

Comenzando con la primera etapa del proceso *ETL*, el objetivo de la extracción de datos consiste en investigar y explotar los posibles recursos disponibles para recoger toda la información necesaria para el proyecto. Consecuentemente, los requisitos fundamentales de esta etapa consistirán en localizar las fuentes de datos a utilizar y emplear las herramientas necesarias para extraer dichos datos.

Fuentes de datos

La información necesaria para conseguir el objetivo de este proyecto está formada por opiniones públicas de personas sobre algún tema o temas en concreto. La manera más sencilla de obtener estos datos es empleando recursos web como foros, *blogs* y redes sociales. Más concretamente, se ha optado por investigar la disponibilidad de *APIs* públicas de los principales sitios web donde las personas publican sus opiniones.

A continuación, se realiza un pequeño resumen de la información de la que se dispone actualmente sobre las *APIs* de cada plataforma.

Twitter

En 2006 se abrió al público la *API REST* [63] de Twitter, que actualmente se encuentra ya en su versión **v2**, aunque coexiste a su vez con algunas partes de la misma aún en la versión **v1.1** y otras de pago (*Premium v1.1* o *Enterprise*). Está basada en *GraphQL*²⁴ y devuelve los resultados en formato *JSON*.

Los permisos que se deben asignar son solo de lectura o escritura de contenido. Mientras que el número de peticiones varía en función del *endpoint*, la ventana temporal de restricción se limita a tan solo 15 minutos [65].

Ofrece acceso de lectura, escritura, modificación y borrado de una amplia variedad de recursos, como puede verse en la [Tabla 5.1](#).

Recurso	Versión	Descripción
Tweets	v2	Operaciones <i>CRUD</i> .
	v1.1	
	<i>Premium</i>	
	<i>Enterprise</i>	
Users	v2	Gestión y búsqueda de usuarios
	v1.1	y relaciones entre los mismos.
	<i>Premium</i>	
	<i>Enterprise</i>	
Spaces	v2	Búsqueda de espacios y participantes.
Direct Messages	v1.1	Envío y respuesta a mensajes directos.
Lists	v2	Gestión de listas de contactos.
	v1.1	
Trends	v1.1	Identificar tendencias por zonas geográficas.
Media	v1.1	Cargar archivos multimedia.
Places	v1.1	Búsqueda de lugares.

Tabla 5.1: Recursos disponibles a través de la *API* de Twitter. Fuente: [66]

²⁴Lenguaje de consultas para *APIs* que facilita la gestión de datos y peticiones (<https://graphql.org/>).

Facebook

La *API* de Facebook originalmente utilizaba *FQL* (*Facebook Query Language*) como lenguaje de consulta, parecido a *SQL*. Sin embargo, en 2010 comenzó la migración hacia *Graph API* [46], actualmente en su versión **v16.0**. Se organiza en función de colecciones, nodos y campos. Un nodo es un objeto único que representa una clase del diccionario de datos en concreto, mientras que los campos son atributos del mismo y una colección comprende un conjunto de nodos. Toda esta información es presentada en formato *JSON*.

Presenta una gran cantidad de permisos [49] que son requeridos para realizar las acciones de gestión, algunos de los cuales es necesario que sean aprobados por Facebook para su uso. Además, el número de peticiones que se pueden realizar se limita a 200 por hora por cada usuario [50].

La lista de nodos que presenta la *API* es muy extensa, aunque en la [Tabla 5.2](#) se muestran algunos de ellos que podrían resultar útiles para el desarrollo de este proyecto.

Nodo	Descripción
<i>Comment</i>	Comentarios de los objetos.
<i>Link</i>	Enlaces compartidos.
<i>Group</i>	Objeto único de tipo grupo.
<i>Likes</i>	Lista de personas que han dado <i>like</i> a un objeto.
<i>Page</i>	Información sobre páginas.
<i>User</i>	Representación de un usuario.

Tabla 5.2: Muestra de nodos disponibles en *Graph API* de Facebook. Fuente: [47]

Instagram

Lanzada originalmente en 2014 y actualmente integrada junto a la *Graph API* de Facebook. Dispone de dos versiones, una más básica enfocada solamente al consumo de contenido, y la normal, que permite realizar diversos tipos de acciones sobre la cuenta y llevar a cabo su gestión.

Se dispone de un conjunto de permisos requeridos bastante más reducido que para la *API* de Facebook, aunque el límite de peticiones es el mismo ya que funciona sobre la propia *Graph API*.

Al estar basada también en la misma tecnología, la estructura consta de los mismos elementos mencionados en el apartado anterior. La diferencia serían los nodos principales en los que se distribuye su contenido, como se observa en la [Tabla 5.3](#).

Nodo	Descripción
Comment	Comentarios de los objetos.
Hashtag	Representa un <i>hashtag</i> .
Multimedia	Referencia una foto, vídeo, historia o álbum.
User	La cuenta de un usuario.
Page	Información sobre páginas.

Tabla 5.3: Muestra de nodos disponibles en *Graph API* de Instagram.
Fuente: [\[48\]](#)

YouTube

Introducida en el año 2013, actualmente en su versión **v3**, y permite la integración de funcionalidades de la plataforma, búsqueda de contenido y análisis demográficos. Cada recurso se representa como un objeto *JSON* sobre el que se pueden ejecutar varias acciones.

Respecto a permisos requeridos, no son tan estrictos como por parte de Meta. No obstante, el número de peticiones se calcula en función de las «unidades» que consume cada tipo de petición, teniendo un total básico de 10 000 al día [\[21\]](#).

Cada recurso se representa como un objeto de datos con identificador único. Entre ellos, los más representativos para la realización de este proyecto podrían ser los expuestos en la [Tabla 5.4](#).

Recurso	Descripción
Caption	Representa los subtítulos de un vídeo.
Comment	Comentarios de los objetos.
Playlist	Colección de vídeos accesibles de forma secuencial.
Search result	Información de una búsqueda que apunta a un objeto.
Video	Objeto representativo para un vídeo.

Tabla 5.4: Recursos disponibles a través de la *API* de YouTube. Fuente: [\[22\]](#)

Reddit

Esta *API* fue lanzada en 2011, proporcionando acceso y gestión sobre todas las acciones disponibles desde su interfaz web. También la menos restrictiva de las estudiadas en esta sección, aunque no por ello menos trabajada.

Como ventaja respecto al resto, permite realizar hasta 60 peticiones por minuto [70] a través de todos sus *endpoints*.

Los recursos se representan como objetos tipo *JSON*. En la [Tabla 5.5](#) se pueden observar las principales estructuras de datos.

Recurso	Descripción
<i>Comment</i>	Comentario de las demás estructuras de datos.
<i>Subreddit</i>	Representación de un subforo.
<i>Message</i>	Información sobre mensajes.
<i>Account</i>	Datos de la cuenta de un usuario.

Tabla 5.5: Recursos disponibles a través de la *API* de Reddit. Fuente: [40]

Método de extracción

Para realizar la extracción de los datos se comenzó a realizar un prototipo inicial empleando los *API wrappers* mencionados anteriormente (véase la [Sección 4.2](#)). No obstante, debido a las razones ya explicadas en dicho apartado, finalmente se ha optado por utilizar la herramienta Airbyte para realizar esta tarea.

Esta plataforma de código abierto se ha instalado en la máquina local mediante contenedores *docker*, lo que ha facilitado en gran medida su despliegue ya que está compuesta por una arquitectura compleja con varios servicios interconectados entre sí. Cuenta con una interfaz web sencilla que permite realizar la gestión y configuración de fuentes de datos, destinos de datos y conexiones. En la [Figura 5.12](#) se puede observar una visión general de la arquitectura de esta herramienta.

También presenta una *API* propia [2] desde la que es posible gestionar las configuraciones de dichos recursos sin necesidad de acceder a su interfaz web.

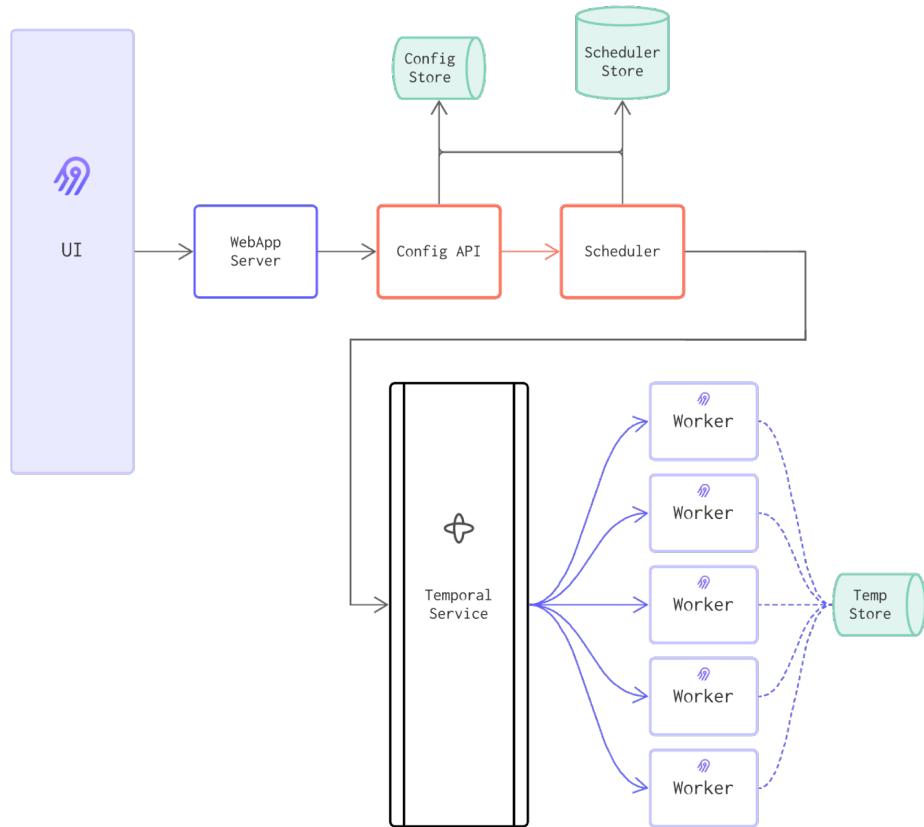


Figura 5.12: Visión general de la arquitectura de *Airbyte*. Fuente: [3]

Inicialmente se comenzó utilizando el conector básico para Twitter que ya presentaba esta herramienta. Tras comprobar su funcionamiento y las posibilidades de extracción de datos que ofrecía, resultó no ofrecer los datos suficientes que se esperaba.

Por ello, al tratarse de una herramienta *open-source*, se procedió a realizar un desarrollo propio y modificar el código base de dicho conector. Se ampliaron así las posibilidades de parametrización y extracción de datos que ofrecía, mejorando así su facilidad de uso y extensibilidad de opciones. Dicho conector está disponible para su uso como una imagen *Docker* que se puede agregar como un nuevo conector en Airbyte, disponible en el siguiente enlace: <https://hub.docker.com/r/liviujv/airbyte-source-twitter/tags>.

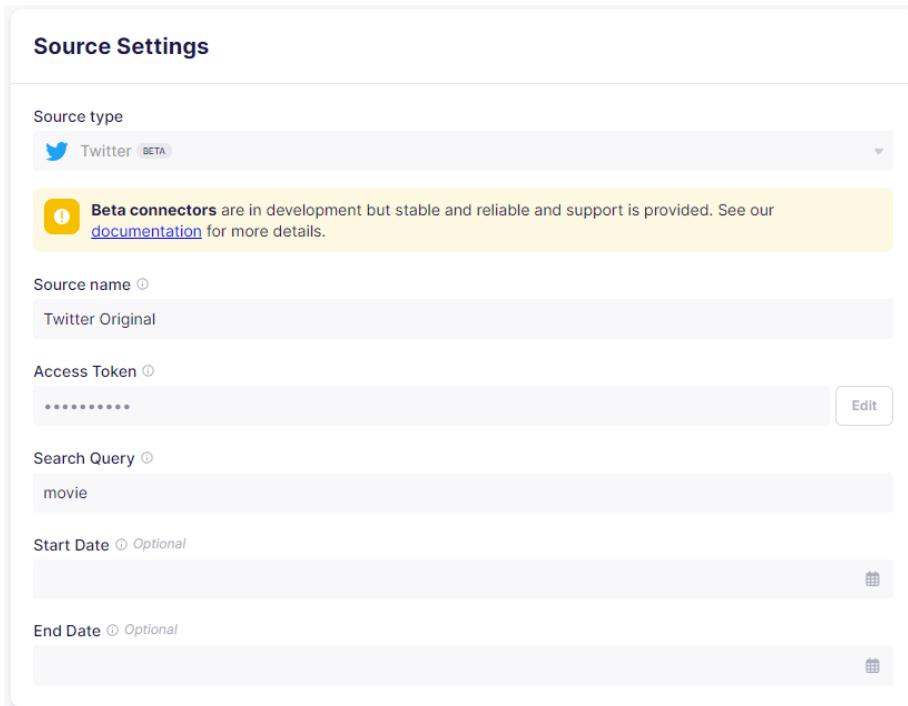


Figura 5.13: Configuración básica del conector original de Airbyte para Twitter

En la Figura 5.13 se pueden observar las opciones básicas de configuración del conector, mientras que en la Figura 5.14 se puede comprobar la cantidad de opciones extendidas que se han implementado.

Además, se ha conservado el flujo de datos básico que ya presentaba el conector y se ha añadido un flujo de datos avanzado, en el que se permite la extracción de todas las opciones de configuración documentadas en la *API* de Twitter para la ejecución de consultas y peticiones.

El código de dicho desarrollo se puede comprobar en el *Pull Request* realizado al repositorio oficial de la herramienta Airbyte y su correspondiente *issue* documentada, accediendo al siguiente enlace: <https://github.com/airbytehq/airbyte/pull/25534>.

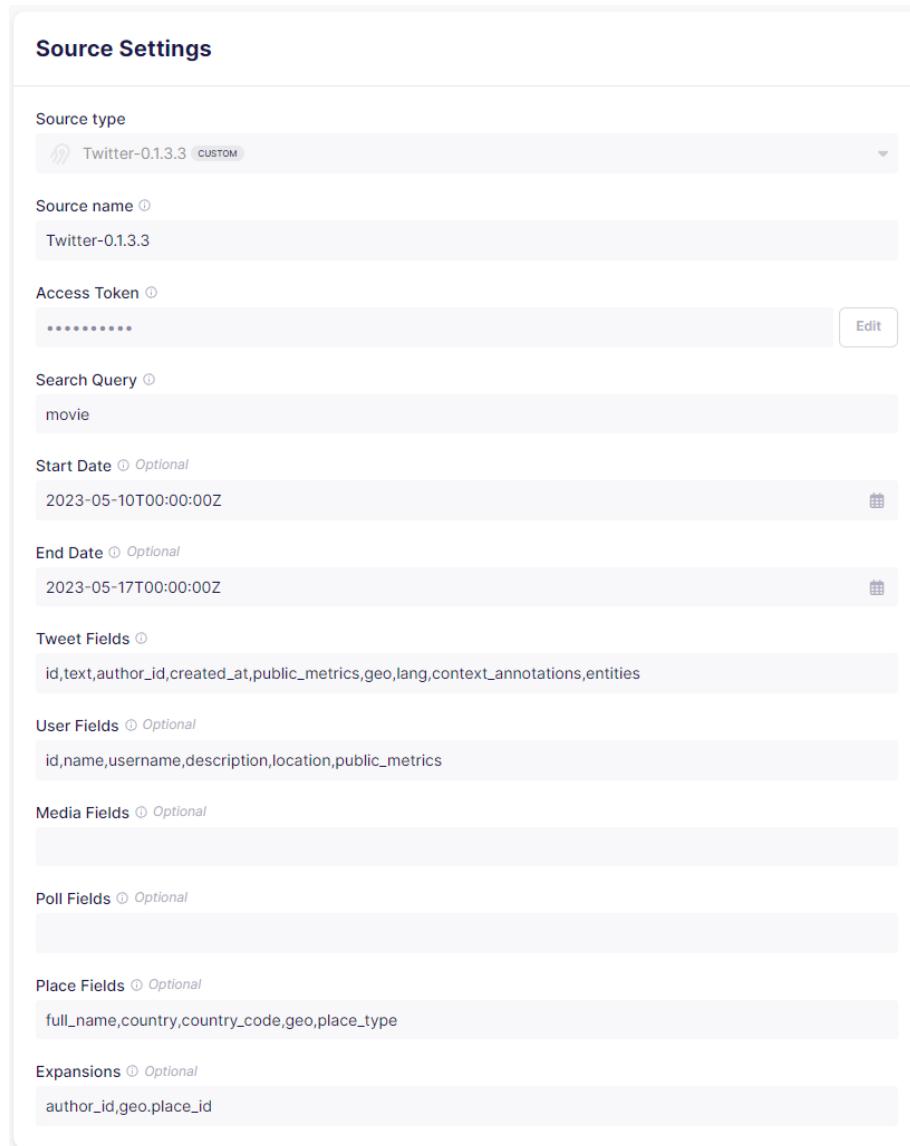


Figura 5.14: Configuración ampliada del conector mejorado de Airbyte para Twitter

Cambios críticos en las APIs investigadas

Al comienzo del proyecto se decidió utilizar inicialmente la *API* de Twitter para realizar la extracción de datos. Las razones tras esta decisión se basaron en que presenta una mayor facilidad de uso que el resto de las *APIs* mencionadas, además de que la herramienta de extracción Airbyte ya disponía de un conector básico para ello. Consecuentemente, la siguiente

fuente de datos que se había planteado utilizar para el proyecto sería la *API* de Reddit, por presentar mayor facilidad de extracción de datos enfocados a temas concretos que Facebook o Instagram.

Después de terminar las tareas de integración y mejora del conector de Airbyte para Twitter se dirigió el enfoque hacia las demás etapas del proyecto, dejando así por finalizada esta parte. Tras la inclusión, despliegue y configuración de la herramienta de orquestación de procesos Apache Airflow, se procedió a la realización de pruebas mediante la ejecución completa de la *pipeline ETL* para comprobar el correcto funcionamiento del proyecto, surgiendo así problemas en la etapa de extracción de datos.

Al comprobar el origen de los errores, se descubrieron los cambios críticos que había sufrido recientemente la *API* de Twitter [8, 59]. El acceso básico sin coste que existía anteriormente permitía realizar hasta 500 000 peticiones de manera mensual a una multitud de diversos *endpoints*. Actualmente, el acceso de dicho plan (*Free*) sin coste ha quedado altamente restringido, permitiendo tan solo realizar publicaciones en la propia cuenta del desarrollador. El siguiente plan (*Basic*), que permite el acceso al *endpoint* de búsqueda de *tweets* presenta un coste de \$100 mensuales, limitando a 10 000 el número de peticiones de lectura que se pueden realizar. El siguiente plan que se aproxima al número original de peticiones es el *Pro*, con un coste de \$5 000 mensuales y restringiendo el acceso a 300 000 peticiones de búsqueda.

Al comprobar la siguiente opción investigada inicialmente a utilizar como segunda fuente de datos, se descubrió que la *API* de Reddit también está sufriendo cambios muy restrictivos [71]. Los usuarios y desarrolladores de la plataforma han organizado numerosas protestas [41] para intentar evitar estos hechos. El nuevo plan básico y sin coste ofrece entre 10 y 100 peticiones por minuto, dependiendo del tipo de cuenta y acceso concedido.

Estos cambios críticos en las *APIs* fueron detectados **a lo largo del Sprint 10**. Por lo que fue necesaria la toma de una decisión rápida sobre la manera de proseguir con esta parte del proyecto.

Conjunto de datos de demostración

Por las razones explicadas en la sección anterior, las mejoras implementadas en el conector de Airbyte para Twitter se han vuelto poco usables. También se ha descartado el desarrollo de un nuevo conector para Reddit que dependa de la baja cadencia de peticiones posibles a realizar.

Teniendo esto en cuenta y el poco margen temporal restante para la finalización del proyecto, se ha decidido investigar el uso de un posible

conjunto de datos a modo de demostración de uso del proyecto, ya que resulta poco viable utilizar las *APIs* investigadas.

El conjunto de datos seleccionado finalmente ha sido *Game of Thrones S8 (Twitter)* [20], de la plataforma de *data science* Kaggle²⁵. Se trata de un *dataset* sobre la serie de televisión del mismo nombre, que el autor recolectó mediante un *script* en el lenguaje R de manera diaria a lo largo de un mes durante el estreno de su octava temporada.

Este conjunto de datos en formato *CSV* presenta más de 760 000 registros y 88 atributos distintos, con información tanto sobre los usuarios, como las publicaciones realizadas por los mismos. Lo que resulta de gran utilidad ya que se puede reconstruir una estructura de datos similar a la del primer *data pipeline* realizado mediante la *API* de Twitter.

No obstante, dichos datos son del año 2019 mientras que los obtenidos mediante la *API* de Twitter datan de este año 2023. Al momento de visualizar los datos habrá grandes discrepancias en las gráficas que incluyan un eje temporal (véase la Figura 5.15), por lo que estos datos extraídos de la *API* de Twitter no van a dejarse disponibles para su uso, empleando solamente este conjunto de datos investigado a modo de demostración.

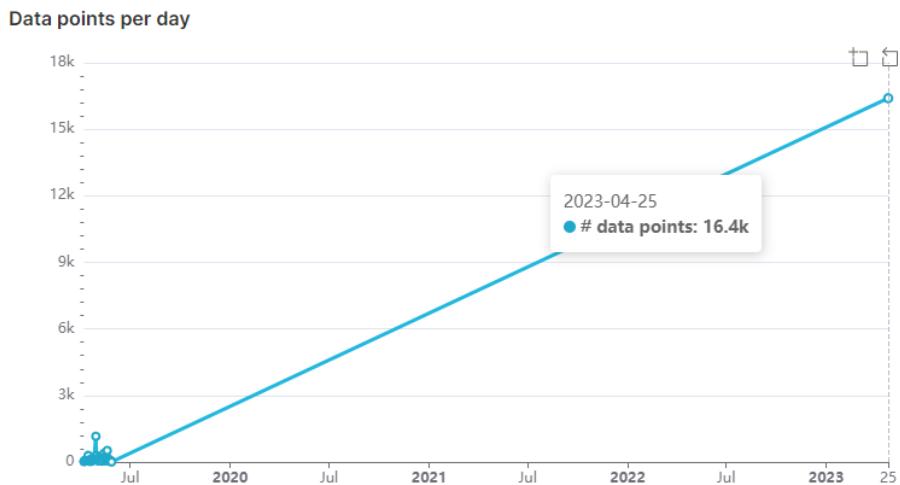


Figura 5.15: Discrepancia en el eje temporal al utilizar los datos recientes de la *API* de TWitter y los del conjunto de datos de demostración

²⁵<https://www.kaggle.com>

Para comprobar de mejor manera los datos disponibles y la usabilidad de los mismos se ha realizado un análisis exploratorio, al que se puede acceder a través del *Jupyter Notebook*²⁶ que se deja disponible.

La primera parte de dicho análisis consiste en comprobar la posibilidad de crear una estructura de datos parecida a la diseñada originalmente para el flujo de datos de la *API* de Twitter, consiguiendo completar dicho esquema de datos en una gran medida.

La segunda parte se centra en buscar palabras clave que puedan servir a modo de tópico o tema de interés sobre el que se pueda obtener más información mediante el análisis de sentimientos. Para ello, se comprueba el interés de los usuarios sobre diversos personajes y temas, de los que se han escogido los siguientes para formar particiones más pequeñas y manejables del conjuntos de datos total:

- **dataset_movie.** Partición compuesta de 6 996 registros que contienen la palabra clave «*movie*» en la publicación.
- **dataset_got.** Partición compuesta de 414 955 registros que contienen la palabra clave «*Game of Thrones*» en la publicación.
- **dataset_season8.** Partición compuesta de 33 663 registros que contienen la palabra clave «*season 8*» en la publicación.
- **dataset_daenerys.** Partición compuesta de 8 830 registros que contienen la palabra clave «*Daenerys*» en la publicación.
- **dataset_jon.** Partición compuesta de 12 222 registros que contienen la palabra clave «*Jon*» en la publicación.

5.2. Carga de datos

La carga de los datos se ha realizado en dos bases de datos distintas para diferentes propósitos. La primera se ha utilizado a modo de *Data Lake* del proyecto para guardar los datos en bruto extraídos, mientras que la segunda se ha empleado como un *Data Warehouse* para realizar la agregación de datos de distintos orígenes.

²⁶https://colab.research.google.com/drive/1d_obU9idFqjsDi7ezeFs1C0RxTUAgH7V#offline=true&sandboxMode=true

MongoDB

Una vez completada la extracción de los datos, resulta necesario persistirlos para su posterior uso. La herramienta seleccionada para realizar esta tarea es *MongoDB* [53].

Esta base de datos no relacional basada en documentos almacena los datos en un *JSON* optimizado llamado *BSON*. Teniendo en cuenta que los datos extraídos mediante las *APIs* que se han detallado en el apartado anterior se encuentran en su totalidad en formato *JSON*, su carga en esta base de datos resulta íntegra y directa, eliminando cualquier necesidad de transformación intermedia para ser ajustados a un esquema concreto.

MongoDB facilita el desarrollo al ofrecer una alta flexibilidad de almacenamiento de documentos no estructurados con diferentes tipos de datos en una misma colección. Esta característica resulta de gran importancia para el caso de uso del proyecto, debido a que las consultas realizadas a los distintos servicios web no siempre van a poder encontrar toda la información solicitada en los parámetros de la petición.

Presenta también capacidades de alta disponibilidad y escalabilidad gracias a la replicación y particionamiento de los datos (*sharding*), cumpliendo con las partes *C* (Consistencia) y *P* (Tolerante a particiones) del *Teorema CAP*.

Para más información sobre los esquemas de datos utilizados, consultar el apartado correspondiente de la [Sección C.2](#).

ClickHouse

Tras la ejecución de los modelos de *NLP* sobre los datos procesados, estos resultados son almacenados y agregados en el sistema *OLAP* (*On-Line Analytical Processing*) ClickHouse [18]. Este sistema gestor de bases de datos (SGBD) analítico y columnar permite realizar consultas rápidas sobre grandes volúmenes de datos.

Los datos con los que se trabaja en el proyecto son peticiones *API* cuya respuesta contiene una serie de campos que vienen o no completos según la disponibilidad de los datos. En la mayoría de los casos, algunos de los campos presentan poca densidad de información, por lo que en gran parte de registros estos campos estarán vacíos. El modelo de datos columnar que emplea ClickHouse es perfecto para este tipo de casos en los que algunos campos puedan presentar poca densidad.

Al realizar las consultas a la base de datos, se filtran únicamente los campos o columnas necesarias para resolver la petición en lugar de iterar sobre todos los campos de cada fila como en una base de datos tradicional. Por consiguiente, ClickHouse ofrece un alto rendimiento para este tipo de casos ya que se consigue aumentar notablemente la velocidad de las consultas ejecutadas al seleccionar únicamente los campos que intervienen en ellas.

ClickHouse también presenta escalabilidad horizontal, permitiendo así adaptarse ante cargas de trabajo con *Big Data*. Además cuenta con una alta flexibilidad, admitiendo datos en diversos formatos y empleando motores de almacenamiento con funciones específicas por cada tabla, según resulte necesario.

Otro punto ventajoso a la hora de trabajar con este SGBD es la capacidad de integración que presenta. Ofrece interfaz directa con la herramienta de visualización empleada Apache Superset y presenta una sintaxis *SQL* nativa, por lo que resulta de gran facilidad a la hora de lanzar consultas contra los datos.

Para más información sobre los esquemas de datos utilizados, consultar el apartado correspondiente de la [Sección C.2](#).

5.3. Transformación de los datos

Una vez finalizada la carga inicial de los datos en *MongoDB*, se procede con su procesamiento. Para esta labor, se han empleado dos herramientas descritas en los siguientes apartados.

Apache Spark

Para la primera fase de la etapa de transformación de datos se ha utilizado *Apache Spark* [35] con el lenguaje de programación *Scala*, ya que permite una gran velocidad de cómputo ideal al trabajar con grandes cantidades de datos. Además presenta las características perfectas para *Big Data*, su arquitectura es escalable y el procesamiento se puede realizar de manera distribuida.

Las tareas de procesamiento contemplan la lectura de los datos en bruto y una posterior limpieza inicial de los mismos. De esta manera, se recogen únicamente los datos necesarios y se elimina la compleja estructura de datos de la que se han extraído. A continuación, se seleccionan los campos que serán exportados nuevamente a una colección de *MongoDB* de datos limpios.

Los esquemas de datos antes y después de este procesamiento se pueden comprobar en la [Sección C.2](#).

Estos datos procesados estarán ya listos para servir de entrada a los algoritmos de *sentiment analysis* que serán ejecutados a continuación.

HuggingFace Transformers

Para la segunda fase de la etapa de transformación de datos se ha utilizado la librería *HuggingFace Transformers* [26] del lenguaje de programación *Python*. Esta librería ha facilitado la ejecución de las técnicas de procesamiento de lenguaje natural gracias a su interfaz unificada para la carga y ejecución de *LLM* (*Large Language Models*) pre-entrenados.

Más concretamente, se han enriquecido los datos procesados gracias a la inferencia de varias tareas mediante variaciones del modelo *BERT* ya explicado en la [Sección 3.4](#). A continuación, se describen las distintas tareas *NLP* empleadas y sus respectivos modelos empleados:

- **Sentiment Analysis.** Consiste en identificar y extraer la opinión o actitud de una persona o entidad hacia un tema. Las posibles categorías de sentimiento son: positivo, neutro o negativo. El modelo empleado: *Twitter-roBERTa-base for Sentiment Analysis* [14, 11, 45].
- **Emotion Analysis.** Consiste en reconocer y clasificar las emociones expresadas por una persona en un texto. Las posibles categorías de emoción son: optimismo, alegría, tristeza, enfado. El modelo empleado: *Twitter-roBERTa-base for Emotion Recognition* [13, 11, 45]
- **Topic Classification.** Consiste en clasificar en una o más categorías a un enunciado según su contenido y contexto. El modelo empleado: *tweet-topic-21-multi* [12, 6]. Las posibles categorías son las mostradas a continuación:

0: arts_&_culture	1: business_&_entrepreneurs
2: celebrity_&_pop_culture	3: diaries_&_daily_life
4: family	5: fashion_&_style
6: film_tv_&_video	7: fitness_&_health
8: food_&_dining	9: gaming
10: learning_&_educational	11: music
12: news_&_social_concern	13: other_hobbies
14: relationships	15: science_&_technology

```
16: sports           17: travel_&_adventure  
18: youth_&_student_life
```

- **Named Entity Recognition.** Consiste en localizar y etiquetar las entidades nombradas que aparecen en un texto, como personas, lugares, organizaciones, etc. El modelo empleado:
`tner/twitter-roberta-base-dec2021-tweetner7-all [61, 67, 68]`. Las posibles categorías de entidad son las mostradas a continuación:

0: corporation	1: creative_work	2: event
3: group	4: location	5: person
6: product		

5.4. Visualización de los datos

En esta sección se detallarán los aspectos relevantes de la herramienta seleccionada para la visualización de los datos. Apache Superset [37] permite a la exploración y visualización de datos, además de compartir información de forma interactiva y colaborativa.

Presenta capacidades de creación de usuarios y permisos, por lo que es posible designar roles específicos para cada usuario según las necesidades del mismo. De esta manera, se consigue implementar cierta seguridad en los datos, de modo que ciertos usuarios podrían interactuar solamente con ciertos *dashboards* según el nivel de acceso que tengan, en caso de trabajar con datos sensibles.

Además, cuenta con una sección llamada «*SQL Lab*» en la que es posible realizar consultas en formato *SQL* nativo sobre los distintos conjuntos de datos cargados. Una vez ejecutadas las consultas, estas se pueden exportar a un *dataset* virtual que se puede tomar como base para la creación de nuevas visualizaciones.

En este proyecto se ha desarrollado un cuadro de mando integral que comprende 5 vistas a distinto nivel de detalle sobre los datos recogidos. Para ello, se han empleado las visualizaciones pertinentes junto a una paleta de colores elegida intencionalmente para representar el análisis de sentimientos.

Las vistas del *dashboard* permiten la obtención de *insights* no solo en el ámbito general mediante métricas sobre los datos extraídos, sino también

sobre las estadísticas representadas de las publicaciones realizadas y los usuarios de las mismas, junto a la información enriquecida añadida mediante las técnicas *NLP*.

Para utilizar los datos almacenados en el sistema *OLAP* ClickHouse ha hecho falta la instalación de un conector adicional. Esta configuración, así como el cuadro de mando desarrollado y la importación de los *datasets* desde ClickHouse, se realiza mediante un *script* automático para facilitar el despliegue para el usuario.

El diseño e implementación de las vistas para el *dashboard* se pueden comprobar en mayor detalle en la [Sección C.2](#).

5.5. Orquestación de los procesos

La arquitectura implementada en este proyecto resulta compleja, no solamente a nivel de integración de las herramientas empleadas, sino también de la comunicación realizada entre ellas y los procesos que intervienen entre cada una.

Debido a estas razones, surge la necesidad de emplear una herramienta capaz de gestionar todo el flujo de acciones que se lleva a cabo entre los distintos componentes del proyecto. Para lograr este objetivo, se ha seleccionado *Apache Airflow* [36] como orquestador de procesos.

En las siguientes secciones se desarrollan los aspectos que mayor influencia han tenido para la integración de esta etapa.

Configuración y despliegue

La documentación de *Apache Airflow* indica de la disponibilidad de un «entorno dockerizado listo para producción», aunque dicha afirmación no resulta del todo cierta. Para la correcta configuración y despliegue de esta herramienta ha resultado necesaria la creación de un *script* que automatice de la manera más abstracta posible para el usuario la parametrización y despliegue iniciales de la plataforma.

Este *script* se encarga de la declaración de las variables y conexiones a las demás herramientas necesarias para la ejecución de las *data pipelines* creadas. También crea y configura la clave de encriptación *Fernet* [32] y realiza la instalación de los *plugins* empleados (`apache-airflow-providers-docker` [34] y `apache-airflow-providers-airbyte` [33]), que se han utilizado para,

respectivamente, realizar la integración con el servicio *Docker* y ejecutar la sincronización de datos en Airbyte.

Acceso seguro al *Docker socket*

Teniendo en cuenta los objetivos del proyecto sobre la creación de una plataforma segura y autocontenido, se ha realizado el despliegue completo de la plataforma en contenedores *Docker*.

Debido a esto, las etapas de procesamiento e inferencia de los datos se realizan en contenedores «desechables», en el sentido de que solamente resulta necesario que estén activos durante el tiempo necesario para realizar sus operaciones. De esta manera, esta parte del flujo de datos se vuelve más dinámica y eficiente, utilizando únicamente los recursos necesarios durante el tiempo requerido y liberándolos nuevamente tras finalizar dichas tareas.

Para realizar dicho despliegue y borrado dinámico de contenedores, *Apache Airflow* necesita acceso al *Docker socket*. No obstante, conceder acceso a este *socket* significa otorgar permisos de administrador local sobre la propia máquina en la que se despliega. Para solventar este problema y securizar el *socket*, se ha desplegado un *proxy* [60] también *dockerizado* y configurable que permite el paso de peticiones benignas y bloquea las malignas.

Integración y flujos de datos

La orquestación de los procesos y comunicación entre los numerosos componentes del proyecto ha requerido la creación de distintos flujos de datos utilizando diversas tareas y configuraciones.

Teniendo en cuenta la cantidad de fuentes de datos seleccionadas finalmente para el proyecto, se han diseñado dos *data pipelines*. Un flujo de datos con un *DAG* (*Directed Acyclic Graph*) normal para los datos provenientes de la *API* de Twitter y otro flujo de datos que genera de manera dinámica los *DAGs* necesarios para todos los subconjuntos de datos del *dataset* explicado en anteriores apartados. No obstante, en ambos flujos de datos han intervenido las mismas tareas y en el mismo orden de ejecución, como se puede observar en la [Figura 5.16](#).

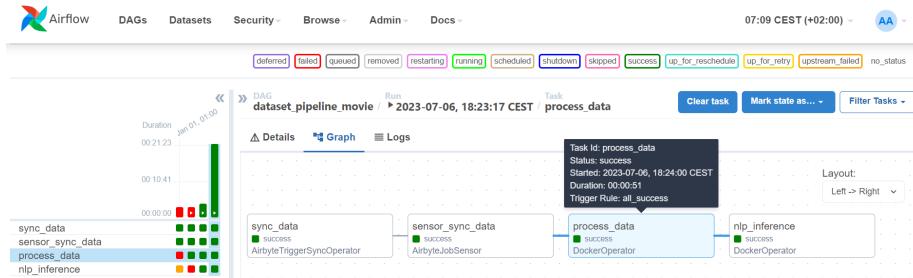


Figura 5.16: Tareas de la *data pipeline* sobre el *dataset* de demostración

- La primera tarea (`sync_data`) es un actuador que se encarga de ejecutar la sincronización de los datos en Airbyte. Para ello, es necesario establecer previamente la conexión con la herramienta de extracción de datos desde la configuración de Airflow y especificar el identificador de la conexión a ejecutar.
- La segunda tarea (`sensor_sync_data`) es un sensor que se encarga de recibir la señal con el estado de la sincronización cuando Airbyte finaliza la extracción de datos. Según el estado de la señal, finalización satisfactoria o no, termina la *pipeline* o continúa con la siguiente tarea.
- La tercera tarea (`process_data`) es un actuador que se encarga de realizar el procesamiento de los datos mediante Apache Spark. Los parámetros de configuración necesitan los datos de conexión a la base de datos MongoDB, junto al término de búsqueda empleado para las consultas de extracción y en qué colección se están almacenando estos datos en bruto. Esta tarea levanta un contenedor *Docker* con Apache Spark para realizar sus operaciones, que posteriormente es eliminado para liberar los recursos ocupados. Durante la ejecución de esta fase se puede acceder a la interfaz web de monitorización que provee Spark.
- La cuarta tarea (`nlp_inference`) es otro actuador que se encarga de realizar la inferencia de los modelos *NLP* (*Natural Language Processing*). Los parámetros de configuración necesitan los datos de conexión a las bases de datos MongoDB y ClickHouse, junto al término de búsqueda empleado para las consultas de extracción. Esta tarea levanta un contenedor *Docker* con la librería *HuggingFace Transformers* y emplea los modelos descritos en la sección anterior para realizar la inferencia de los datos. Tras su finalización, los datos son agregados en ClickHouse y el contenedor eliminado para liberar los recursos utilizados.

5.6. Interfaz web de acceso centralizado

Como se ha explicado en los apartados anteriores, la mayoría de los componentes utilizados para crear esta plataforma poseen una interfaz web que permite monitorizar y gestionar el servicio correspondiente.

La modularidad que ofrece esta arquitectura invita a posibles extensiones de funcionalidades o a la incorporación de distintas herramientas o nuevos servicios. Esto implica también la posibilidad de que haya más interfaces web a las que acceder según con qué parte de la plataforma se quiera trabajar, en caso de que se tenga los permisos suficientes para ello.

Por estas razones, se ha decidido crear un punto único de acceso centralizado a todas las interfaces de gestión y monitorización en forma de página web. De esta manera, se agiliza el acceso al resto de interfaces que ofrece la plataforma actualmente en caso de que se necesite trabajar de manera paralela en varios puntos de la misma.

El desarrollo de la página web (véase la [Figura 5.17](#)) se ha realizado mediante el *framework* *Flask* con el objetivo de que esta interfaz sea una aplicación web ligera con las funcionalidades básicas y necesarias. El diseño se ha realizado a medida, *responsive* (adaptable a cualquier dispositivo) y teniendo en cuenta un estilo minimalista acorde a las necesidades de un punto web de acceso centralizado.

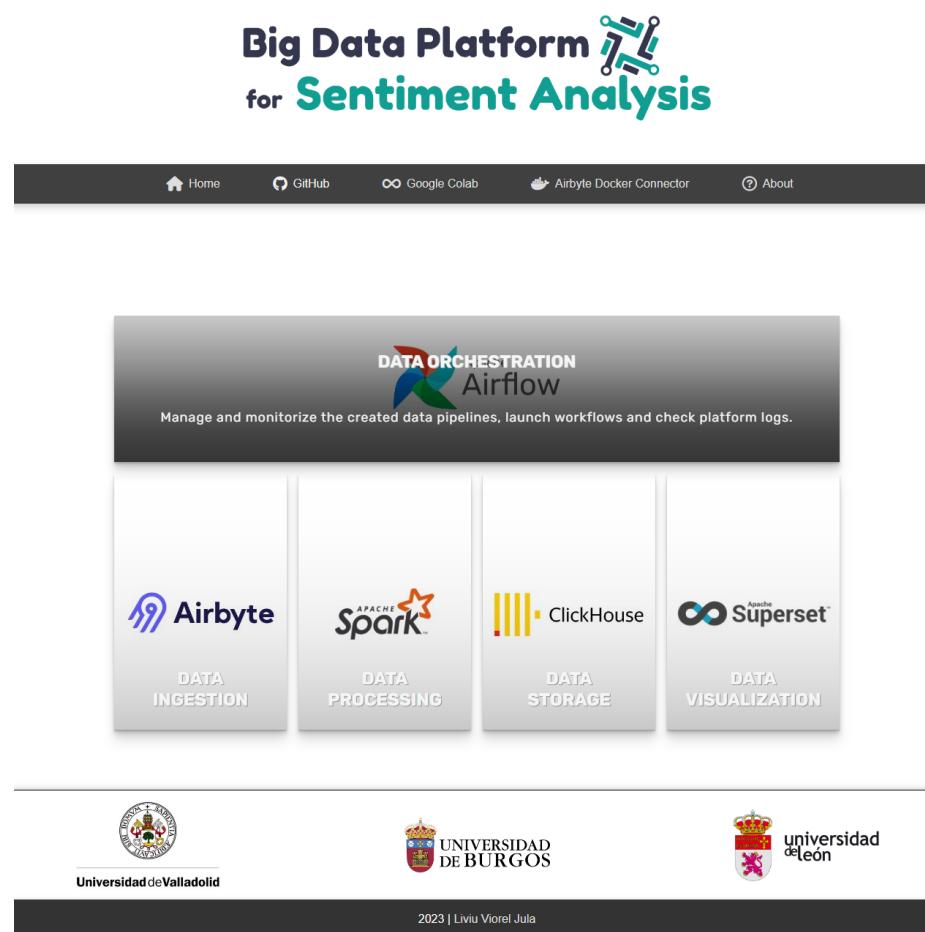


Figura 5.17: Aplicación web desarrollada como punto de acceso centralizado

Trabajos relacionados

En este apartado se describirán otras herramientas similares ya existentes que cumplen un propósito similar al planteado en este proyecto. También se escribirá sobre los principales artículos científicos que comprenden el *state-of-the-art* relacionado con las técnicas de procesamiento de lenguaje natural utilizadas.

6.1. Herramientas similares

A continuación se detallan las características de las principales plataformas que existen actualmente que cumplen con funciones similares a las planteadas en este proyecto. Se han categorizado según supongan o no algún coste económico para su utilización.

Herramientas de pago

Comenzando con las opciones de pago, por ser más establecidas y conocidas que las gratuitas.

Brand24

Es una plataforma²⁷ que monitoriza las menciones sobre la marca del cliente tanto en la web como en redes sociales. Utiliza técnicas NLP para analizar en tiempo real los datos de diversas fuentes como blogs, foros, redes sociales, vídeos...

²⁷<https://brand24.com/>

Una de las ventajas competitivas que ofrece es su capacidad de mostrar la influencia que ha tenido cada mención. Como desventaja, cabe destacar el limitado número de menciones que permite monitorizar en sus servicios de suscripción. El rango de precios comprende desde los \$49 mensuales del paquete básico hasta los \$348 del paquete ejecutivo.

MonkeyLearn

Es un conjunto de herramientas de análisis de texto que permite crear modelos propios de *machine learning* sobre los datos introducidos, empleando la propia interfaz gráfica de la plataforma.

Como ventaja principal, provee unos modelos ya entrenados que se pueden utilizar en la mayoría de las situaciones, pero permite también entrenarlos sobre los datos específicos que interesen al cliente. Como desventajas, se podrían incluir la manera de establecer la conexión con los datos, puesto que necesita acceso directo a la base de datos del cliente, además de requerir una suscripción mensual de \$299.

Repustate

Es una herramienta de análisis²⁸ de sentimientos que analiza de manera sintáctica los datos introducidos para poder evaluar de mejor manera la intención de cada texto. También es capaz de analizar *emojis* según el contexto en el que se utilicen y provee una API que da soporte a 23 idiomas distintos.

Las principales ventajas que ofrece son la gran cantidad de idiomas que soporta y la posibilidad de especificar distintos significados de palabras concretas para mejorar el análisis que realiza. Como principal desventaja, la utilización de este servicio requiere una suscripción mensual de \$199 para su plan *Standard* o \$499 para el *Premium*.

Herramientas gratuitas

A continuación, las opciones que no requieren realizar gasto económico alguno para utilizar sus funcionalidades básicas.

²⁸<https://www.repustate.com/>

Social Searcher

Es una herramienta sencilla²⁹ que ofrece búsqueda por palabras clave, etiquetas o usuarios y muestra unos análisis básicos sobre los resultados obtenidos. Muestra un *dashboard* con varias pestañas en las que se realizan distintos tipos de análisis, además de gráficos diversos que categorizan las menciones en temas y clasifican las opiniones de los usuarios.

La principal ventaja de esta herramienta es que permite aprovechar sus servicios de manera gratuita y sin límite de consultas, aunque tenga también planes de pago. Como desventaja, las funcionalidades que ofrece la versión gratuita son bastante básicas.

Tweet Sentiment Viz

Esta herramienta es la más básica³⁰ de la lista. Muestra una serie de gráficos exploratorios (temas, mapas de calor, nubes de palabras, etc.) sobre los datos buscados en tiempo real en función de palabras clave.

Como principal ventaja, es que funciona bastante bien dentro de unos límites preestablecidos. Entre sus desventajas, esta herramienta analiza únicamente datos de la plataforma Twitter, además de emplear técnicas de bolsas de palabras. Por lo que tendrá dificultades a la hora de interpretar cualquier palabra utilizada que no esté dentro de dichos diccionarios.

6.2. Artículos científicos

A continuación, se detallan los artículos científicos que más relevancia han tenido en relación a los objetivos establecidos para este proyecto.

BERT: Bidirectional Encoder Representations from Transformers

Se trata de un modelo [23] que utiliza una red neuronal ya entrenada para generar *word embeddings* que son utilizadas posteriormente como características en modelos *NLP*.

BERT se basa en *transformers* (mecanismos de atención que «aprenden» correlaciones entre las palabras de un texto). Estos *transformers* presentan dos componentes, un *encoder* que procesa los datos de entrada y un *decoder*

²⁹<https://www.social-searcher.com/>

³⁰https://www.csc2.ncsu.edu/faculty/healey/tweet_viz/tweet_app/

que se encarga de realizar las predicciones correspondientes. Sin embargo, como el objetivo es construir un modelo de lenguaje, tan solo hace falta la primera parte de estos, el codificador.

Mientras que los modelos hasta el momento tomaban una dirección de lectura secuencial de los datos (bien de izquierda a derecha o bien al revés), el codificador del *transformer* es capaz de leer cada palabra del texto a la vez. Esto permite al modelo analizar el contexto general en el que se presenta cada palabra y no teniendo en cuenta solamente una dirección. De esta manera, se considera un modelo «bidireccional», aunque en realidad no tenga una dirección como tal.

Generalmente, los modelos de lenguaje se entrena intentando predecir una secuencia de palabras dentro de un texto, lo que los convierte en unidireccionales. Por ello, *BERT* emplea dos estrategias para mantener su habilidad bidireccional:

- ***Masked LM (MLM)***. La primera estrategia que se utiliza es ocultar, mediante un *token*, a forma de máscara aproximadamente un 15 % de las palabras del texto de entrada del codificador. Posteriormente, el modelo intentará predecir las palabras que faltan basándose en el contexto que las rodea.
- ***Next Sentence Prediction (NSP)***. La segunda estrategia consiste en entrenar el modelo mediante pares de frases. La mitad de los datos de entrada se divide de tal manera que la segunda frase de cada par es la que va a continuación de la primera frase en el texto original. Mientras que en la otra mitad de los datos, la segunda frase se escoge al azar del texto original. De esta manera, se asume que el modelo será capaz de distinguir correctamente qué frase tiene sentido a continuación de otra. Se utilizan una serie de *tokens* para indicar el inicio y final de cada frase.

Ambas estrategias se ponen en práctica y se entrena a la vez para conseguir minimizar la «función de pérdida» o *loss function* del modelo.

Conclusiones y Líneas de trabajo futuras

En esta sección final se desarrollarán las conclusiones derivadas del desarrollo de este proyecto, así como los resultados obtenidos en comparación con los objetivos planteados inicialmente. También se incluye un listado de posibles mejoras a implementar de cara a líneas de trabajo futuras.

7.1. Conclusiones

Este proyecto ha sido el resultado de los conocimientos aprendidos hasta el momento y de los obtenidos a medida que se estaba desarrollando esta «Plataforma *Big Data* para *Sentiment Analysis*».

A continuación, se destacan las conclusiones obtenidas tras la realización del proyecto:

- En general, se han podido cumplir la gran mayoría de los objetivos propuestos inicialmente para el proyecto. Se ha desarrollado e integrado una plataforma modular, escalable y capaz de trabajar con *Big Data* utilizando únicamente tecnologías de código abierto.
- Se han diseñado e implementado diversas *data pipelines* mediante el proceso *ETL* (*Extract, Transform, Load*) que han necesitado de la correcta cooperación entre diversas tecnologías. Para ello, ha hecho falta la correcta integración y configuración de los diversos componentes que han intervenido en estos flujos de datos.
- Respecto a la herramienta de extracción de datos Airbyte, se ha conseguido mejorar y extender las funcionalidades que presentaba el

conector base de Twitter. El desarrollo de la mejora de este conector se ha hecho público para la comunidad en el repositorio oficial de la herramienta.

- Para la utilización de las técnicas de procesamiento de lenguaje natural (*NLP*) se han empleado modelos que forman parte del *state-of-the-art* en este campo, más concretamente, las distintas variantes del modelo *BERT*. La investigación realizada sobre dichos modelos ha ayudado a comprender mejor el funcionamiento de los modelos *Transformer* y de las tendencias actuales del *Deep Learning* en general.
- Se ha conseguido el despliegue de la plataforma completa mediante contenedores *docker*. Esto la convierte en una solución integral para el análisis de sentimientos en *Big Data*, además de una solución modular. Por consiguiente, se posibilita la agregación de nuevos componentes para extender las funcionalidades de la misma y el intercambio de unos componentes por otros, en caso de necesitar emplear tecnologías diferentes o adaptarse a distintos casos de uso.
- Se ha llevado a cabo el diseño e implementación de un *dashboard* que comprende 5 vistas distintas en las que se muestra la información procesada a través de la *pipeline ETL* a diferente nivel de detalle. El cuadro de mando creado permite la obtención de nuevos *insights* mediante varias visualizaciones, que muestran los sentimientos de los usuarios respecto al tema inquirido.
- El desarrollo de esta plataforma empleando las tecnologías seleccionadas ha supuesto un grado de aprendizaje nada trivial. Cada herramienta presentaba numerosos conceptos necesarios para utilizarlas correctamente, además de la complejidad inherente que podía darse en cada una de ellas. No obstante, se ha obtenido una gran satisfacción por todos los conocimientos aprendidos, ya que resultarán útiles para futuros proyectos.
- Este proyecto ha resultado de gran complejidad por la cantidad de tecnologías empleadas en su desarrollo, ya que el número de componentes utilizados se podría haber reducido. Esta decisión se ha tomado con el objetivo de afianzar los conocimientos obtenidos hasta el momento sobre las distintas áreas que ha cubierto el Máster y, al mismo tiempo, aprender nuevas tecnologías modernas capaces de cubrir todo el proceso *ETL*. Con los resultados finales obtenidos, este proceso de aprendizaje ha resultado satisfactorio.

- Finalmente, cabe destacar que la realización de este Trabajo de Fin de Máster se ha realizado compaginando tanto la jornada laboral de trabajo como el estudio de las asignaturas del propio Máster. Esto ha supuesto una gran carga de trabajo y limitaciones de tiempo, lo que ha necesitado de una capacidad de organización para la priorización de ciertas tareas y partes del proyecto con el objetivo de finalizar este trabajo final de manera satisfactoria.

7.2. Líneas de trabajo futuras

A continuación, se listan una serie de posibles líneas de mejora o aspectos con los que se podría continuar el desarrollo de este proyecto:

- Actualmente, la plataforma desarrollada se ha enfocado en trabajar con datos en *batch*. Por ello, la ejecución del proceso *ETL*, en especial la etapa de Extracción y la fase de inferencia mediante *NLP* de la etapa de Transformación, conllevan cierta cantidad de tiempo para su finalización.

Una de las posibles mejoras futuras para esta plataforma sería implementar la posibilidad de trabajar también con datos en *streaming*, para los casos de uso en los que se quiera obtener la información de inmediato en lugar de esperar a la ejecución periódica de los *data pipelines*.

- El funcionamiento desarrollado para la fase de inferencia mediante técnicas *NLP* se realiza en las *data pipelines* de manera consecutiva, ejecutando todas las tareas de procesamiento de lenguaje natural una a una sobre los datos. La herramienta Apache Airflow permite ejecución de tareas en paralelo mediante su programación en *DAGs (Directed Acyclic Graph)*. El código desarrollado para las *data pipelines* se ha diseñado de tal manera que permita la inclusión de estos cambios para su generación de manera dinámica.

Por lo que otra posible línea de mejora consistiría en realizar la ejecución cada tarea *NLP* en paralelo, necesitando posteriormente la agregación de los datos inferidos de cada tarea con sus respectivos registros de datos. Esto conllevaría un aumento notable en la velocidad de ejecución de las *data pipelines*, debido a que esta fase es la que supone una mayor carga de trabajo en la plataforma.

- Como se ha mencionado en la anterior sección, la arquitectura de la plataforma está compuesta por numerosas tecnologías. Este diseño se podría reducir y emplear una menor cantidad de componentes en caso de que se quiera aligerar la carga de trabajo en el servidor de despliegue, por ejemplo, eliminando el uso de algunos componentes y unificando su funcionalidad en una sola tecnología o herramienta.
- En contraste con el punto anterior, se podría también sugerir añadir un componente más. Al trabajar con *Big Data*, y más concretamente con opiniones de personas, resulta necesario mantener la seguridad de los datos. Para ello, se podría sugerir la implementación de una herramienta para llevar a cabo la «gobernanza» de los datos y asignar los permisos necesarios a cada usuario según el uso que se necesite dar a los datos.
- Finalmente, otro posible punto de mejora en el ámbito de la seguridad. La mayoría de las herramientas empleadas se han protegido con acceso restringido mediante credenciales básicos de autenticación. Sin embargo, y ya que las herramientas utilizadas lo permiten, se podría ir un paso más en esta línea y realizar la integración de estas tecnologías con un sistema *LDAP* (Lightweight Directory Access Protocol), mejorando así la seguridad de autenticación en la plataforma.

Apéndices

Apéndice A

Plan de Proyecto Software

A.1. Introducción

En las siguientes secciones se realizará un estudio de la planificación temporal seguida durante el desarrollo de este proyecto, además de la viabilidad tanto económica como legal que podría llegar a suponer este trabajo.

Debido a la naturaleza inherente del proyecto, al no tratarse de un *software* típicamente tradicional sino más bien centrado hacia la investigación e implementación de modelos de *machine learning*, no ha resultado sencillo llevar a cabo algunas de las buenas prácticas y conceptos normales de acuerdo a un «Plan de Proyecto Software» tradicional.

A.2. Planificación temporal

La planificación del proyecto se ha llevado a cabo mediante la metodología de desarrollo ágil *Scrum*. A continuación se realiza un desglose de los distintos *Sprints* llevados a cabo.

Inicialmente, se presentan las tareas correspondientes a cada iteración del trabajo y su duración inicial estimada. Posteriormente, se realiza una comparación entre el tiempo total estimado y el real empleado mediante la ilustración de gráficos *burn-down*.

Sprint 0 (01/02/2023 – 15/02/2023)

Este *Sprint* inicial se dedicará a la preparación del entorno de trabajo para el proyecto. Se elegirán las herramientas con las que se trabajará en algunas de las etapas del proyecto, se investigarán técnicas y librerías a utilizar, se realizarán unas pruebas concepto iniciales y se comenzará la labor de documentación.

- **Gestión del Sprint (4h).** Se realizará el planteamiento de las tareas a llevar a cabo a lo largo de este sprint y se documentarán en la [Sección A.2](#) del [Apéndice A](#) de los anexos del proyecto.
- **Elegir IDE (2h).** Para la realización de este proyecto será necesaria la utilización de diversos lenguajes de programación, por lo que la elección de un entorno de desarrollo integrado adecuado resultará de gran ayuda.
- **Estudiar guía L^AT_EX (2h).** Como objetivo para la generación de la memoria del proyecto, se va estudiar una guía sobre L^AT_EX con el fin de recordar los conocimientos necesarios para poder crear la documentación correspondiente.
- **Documentación de la memoria - Técnicas y herramientas (4h).** Comenzar con la documentación de la memoria del proyecto, con la sección «Técnicas y herramientas». De manera inicial, se documentará lo siguiente:
 - **Técnicas**
 - Scrum
 - Natural Language Processing
 - Sentiment Analysis
 - **Herramientas**
 - GitHub
 - ZenHub
 - Overleaf
 - Joplin
 - Super Productivity
- **Documentación de la memoria - Trabajos relacionados (4h).** La siguiente parte de la memoria que se va a redactar será el [Capítulo 5.6](#). En este apartado se describirán otras herramientas similares

ya existentes que cumplen un propósito similar al planteado en este proyecto.

También se escribirá sobre los principales artículos científicos que comprenden el *state-of-the-art* relacionado con las técnicas de procesamiento de lenguaje natural que serán utilizadas.

- **Investigar y probar recursos NLP ya existentes (8h).** Ya que inicialmente no se prevé el desarrollo de un algoritmo NLP propio, se investigará el *state-of-the-art* sobre análisis de sentimientos y se comprobará si existen recursos ya implementados para utilizar en el proyecto.

Sprint 1 (15/02/2023 – 01/03/2023)

Durante la duración de este *Sprint* se investigarán las *APIs* de las posibles plataformas de las que se va a extraer la información textual y las herramientas disponibles para realizar la primera etapa del proceso *ETL*. También se comenzará a realizar una primera prueba concepto utilizando los recursos investigados.

- **Gestión del Sprint (2h).** Se realizará el planteamiento de las tareas a llevar a cabo a lo largo de este sprint y se comenzará a documentar el [Capítulo 4.2](#).
- **Elegir IDE (2h).** Para la realización de este proyecto será necesaria la utilización de diversos lenguajes de programación, por lo que la elección de un entorno de desarrollo integrado adecuado resultará de gran ayuda.
- **Investigar posibles APIs a utilizar (4h).** Como se ha especificado en el objetivo de este proyecto, se necesita información y opiniones públicas de las que poder obtener conocimiento sobre temas concretos. Para ello, se investigará la existencia de *APIs* públicas de los principales sitios web en los que la gente suele expresar sus opiniones de manera general, siendo estos los foros, blogs y redes sociales.
- **Escoger tema inicial con alta polaridad (2h).** Para comprobar el correcto funcionamiento de los recursos *NLP* investigados en el *Sprint* anterior, se escogerá un tema con alta polaridad sobre el que se centrarán las pruebas de dichos recursos. De esta manera, será más sencillo de visualizar el correcto funcionamiento de estos y el análisis de sentimientos mediante ejemplos claros.

- **Investigar herramientas para realizar la extracción de datos (8h).** En este punto se comenzarán a investigar las posibles herramientas para realizar la etapa de extracción de datos del proyecto. Para ello, se compararán las principales alternativas disponibles para realizar la recogida de información de los recursos web descubiertos en este mismo *Sprint*.
- **Crear prototipo inicial para la etapa de extracción de datos (8h).** Para realizar una mejor comparación de las herramientas investigadas en la tarea anterior, se creará un pequeño prototipo para esta primera etapa de extracción de datos sobre las *APIs* seleccionadas, empleando para ello las tecnologías escogidas más relevantes.
- **Documentar los procesos del sprint actual (8h).** A lo largo de este sprint se va a realizar la investigación de varios recursos que deberán ser correctamente documentados, ya que las siguientes etapas del proyecto dependerán de la calidad de la información proporcionada inicialmente.

En este *sprint* tuvo lugar un error de cálculo a la hora de planificar las tareas a realizar. La investigación inicial sobre los recursos de extracción de datos indicó como viable la utilización de los *API wrappers* mencionados en la [Sección 5.1](#) cuando resultó no ser así. Por ello, la estimación inicial de crear un prototipo para la etapa de extracción de datos durante este *sprint* se vio afectada, teniendo que completar su creación durante el siguiente *sprint*.

Sprint 2 (01/03/2023 – 15/03/2023)

Durante la duración de este *Sprint* se investigará la documentación de la herramienta de extracción de datos elegida y se terminará la creación del prototipo planteado inicialmente en el anterior *sprint*.

- **Gestión del *Sprint* (2h).** Se realizará el planteamiento de las tareas a llevar a cabo a lo largo de este sprint y se comenzará a documentar el [Capítulo 4.2](#).
- **Crear prototipo inicial para la etapa de extracción de datos (8h).** Para realizar una mejor comparación de las herramientas investigadas en la tarea anterior, se creará un pequeño prototipo para esta primera etapa de extracción de datos sobre las *APIs* seleccionadas, empleando para ello las tecnologías escogidas más relevantes.

- **Crear cuenta de desarrollador para la API de Twitter (2h).** Para poder utilizar la *API* de Twitter es necesario crear una cuenta de desarrollador para obtener los *tokens* de acceso. Se creará una cuenta dedicada al proyecto para realizar las peticiones correspondientes.
- **Corregir memoria del proyecto (2h).** Se procederá a implementar las correcciones provistas a modo de *feedback* por el tutor en los comentarios de las tareas.
- **Investigar documentación de la herramienta de extracción de datos elegida (4h).** Las herramientas de extracción de datos elegidas inicialmente para realizar esta labor no resultaron del todo óptimas como se ha mencionado. No obstante, otra de las alternativas que se planteaba utilizar más adelante parece resultar más adecuada. Por ello, se procederá a investigar la documentación disponible sobre la herramienta Airbyte [5].
- **Documentar prototipo de extracción de datos (4h).** Tras la creación del prototipo de extracción de datos, será necesario documentar el procedimiento también en la memoria del proyecto para que quede constancia del funcionamiento del mismo.

Sprint 3 (15/03/2023 – 29/03/2023)

Durante la duración de este *Sprint* se investigarán las posibles herramientas a utilizar para la carga de los datos extraídos previamente en la anterior etapa y se continuará con el desarrollo del prototipo planteado.

- **Gestión del Sprint (2h).** Se realizará el planteamiento de las tareas a llevar a cabo a lo largo de este sprint.
- **Comentarios menores en la documentación (2h).** Se procederá a implementar el *feedback* del tutor.
- **Investigar herramienta para realizar la carga de los datos (8h).** En este punto se comenzará a investigar las posibles herramientas para realizar la etapa de carga de datos del proyecto. Para ello, se compararán las principales alternativas disponibles para persistir los datos extraídos.
- **Investigar documentación de la herramienta de carga de datos elegida (4h).** Tras la selección de la herramienta a utilizar para esta etapa del proyecto, se procederá a investigar su documentación para

poder realizar un despliegue correcto de la misma e integrarla junto a los demás componentes del proyecto.

- **Desplegar e integrar la herramienta de carga de datos (8h).** Tras consultar la documentación necesaria, se procederá a realizar el despliegue y configuración de la herramienta para su correcta integración junto a los demás componentes del proyecto.
- **Documentar los procesos del *sprint* actual (8h).** A lo largo de este *sprint* se va a realizar la investigación de la herramienta a emplear para la carga de datos. Se procederá a documentar el despliegue e integración de dicha herramienta con los demás componentes del proyecto.

Sprint 4 (29/03/2023 – 12/04/2023)

Durante la duración de este *Sprint* se realizará el procesamiento del conjunto de datos principalmente y se mejorará la extracción de datos del prototipo creado inicialmente.

- **Gestión del *Sprint* (2h).** Se realizará el planteamiento de las tareas a llevar a cabo a lo largo de este *sprint*.
- **Modificar consulta para extracción de datos (4h).** El método actual para realizar la extracción de datos de la *API* de Twitter presenta limitaciones en cuanto a la posibilidad de los parámetros a especificar. Se va a investigar cómo realizar dicha consulta de otra manera para poder recuperar la información adicional necesaria.
- **Modificación del conector base de Airbyte (8h).** El conector base utilizado para la extracción de datos de Twitter solamente permite realizar consultas simples a su *API*. Para el desarrollo del proyecto y la información requerida en la consulta mejorada, es necesario desarrollar y mejorar el código fuente de este conector para permitir especificar los parámetros necesarios para las consultas correspondientes.
- **Procesar conjunto de datos (8h).** Tras la extracción y carga inicial de los datos, se va a proceder a realizar la limpieza correspondiente de los mismos con el objetivo de prepararlos para su futura explotación.
- **Documentar los procesos del *sprint* actual (8h).** A lo largo de este *sprint* se va a realizar la mejora del método de extracción de datos y el preprocesado de los mismos. Será necesaria la documentación

de estos procesos para tener constancia de las modificaciones que se hayan realizado sobre el conjunto de datos extraído.

Sprint 5 (12/04/2023 – 26/04/2023)

Durante la duración de este *Sprint* se concluirá la parte del procesamiento de datos y se comenzará la inferencia de los modelos de Procesamiento de Lenguaje Natural.

- **Gestión del Sprint (2h).** Se realizará el planteamiento de las tareas a llevar a cabo a lo largo de este *sprint*.
- **Mejora del conector base de Airbyte (8h).** Previamente se modificó el conector base de Twitter para adaptarlo a la consulta realizada. Ahora se van a implementar unas mejoras adicionales que permitan mejorar su integración con el código base de Airbyte.
- **Mejora del procesamiento del conjunto de datos (8h).** Previamente se comenzó con el procesamiento del conjunto de datos. Ahora se van a implementar mejoras sobre esta funcionalidad para permitir adaptarse a los nuevos cambios desarrollados para el conector.
- **Abrir PR al repositorio oficial de Airbyte con las mejoras desarrolladas (8h).** Las mejoras desarrolladas para el conector de Twitter pueden resultar de utilidad para los demás miembros de la comunidad que utilizan la herramienta Airbyte. Por ello, se va a crear un *Pull Request* con los cambios realizados para que sean añadidos al repositorio oficial.
- **Comenzar con la inferencia de los modelos NLP (2h).** Una vez se han procesado y limpiado los datos correspondientes, se puede comenzar con la inferencia de los modelos NLP sobre los mismos.

Sprint 6 (26/04/2023 – 10/05/2023)

Durante la duración de este *Sprint* se continuará con la inferencia de los modelos de Procesamiento de Lenguaje Natural.

- **Gestión del Sprint (2h).** Se realizará el planteamiento de las tareas a llevar a cabo a lo largo de este *sprint*.

- **Continuación de la mejora del procesamiento de datos (4h).** Siguiendo con lo comentado en el *sprint* anterior, se va a incluir el procesamiento de algunos datos adicionales.
- **Tarea NLP: *Sentiment analysis* (4h).** Como continuación de lo comenzado en el *sprint* anterior, se va a proceder con una de las tareas de NLP planteadas inicialmente: el análisis de sentimiento.
- **Tarea NLP: *Topic classification* (4h).** Otra de las principales tareas NLP planteadas inicialmente será la tratada a continuación, la clasificación de temas.
- **Tarea NLP: *Emotion classification* (4h).** Otra tarea NLP adicional que podría resultar interesante será la clasificación de emociones, que se diferencia de la clasificación de sentimientos en que la primera indica la emoción (alegre, triste, enfadado) de un texto y la segunda solamente la positividad o negatividad del texto.
- **Tarea NLP: *Named entity recognition* (4h).** Otra tarea NLP que puede resultar de gran interés para el proyecto es el reconocimiento de entidades (*NER*).

Sprint 7 (10/05/2023 – 24/05/2023)

Durante la duración de este *Sprint* se mejorarán las tareas de inferencia de los modelos de Procesamiento de Lenguaje Natural y se investigará la manera óptima para persistir los datos obtenidos hasta el momento.

- **Gestión del *Sprint* (2h).** Se realizará el planteamiento de las tareas a llevar a cabo a lo largo de este *sprint*.
- **Mejora de la etapa de transformación (8h).** Se van a desarrollar unas mejoras para la etapa de procesamiento de los datos que permitirán obtener una mayor calidad en la información final.
- **Mejora de tarea NLP: *Sentiment analysis* (4h).** Se va a proceder a desarrollar mejoras para la tarea NLP que se encarga del análisis de sentimientos.
- **Mejora de tarea NLP: *Topic classification* (4h).** Se va a proceder a desarrollar mejoras para la tarea NLP que se encarga de la clasificación de temas.

- **Mejora de tarea NLP: *Emotion classification* (4h).** Se va a proceder a desarrollar mejoras para la tarea NLP que se encarga de la clasificación de emociones.
- **Mejora de tarea NLP: *Named entity recognition* (4h).** Se va a proceder a desarrollar mejoras para la tarea NLP que se encarga del reconocimiento de entidades.
- **Investigar base de datos *OLAP* (4h).** Los datos extraídos durante las fases anteriores y los inferidos a través de los modelos NLP empleados han de ser persistidos nuevamente para su posterior explotación de manera visual. Por tanto, para llevar a cabo esta tarea de forma óptima, será necesario investigar una base de datos enfocada al procesamiento analítico en línea (*OLAP*) de los datos, en lugar de emplear modelos enfocados al procesamiento de transacciones en línea (*OLPT*) comunes.

Sprint 8 (24/05/2023 – 07/06/2023)

Durante la duración de este *Sprint*, el objetivo principal será investigar, desplegar e integrar el sistema óptimo a utilizar para persistir los datos obtenidos hasta el momento en una base de datos *OLAP*. Se investigará también una herramienta de visualización compatible que utilizar posteriormente.

- **Gestión del *Sprint* (2h).** Se realizará el planteamiento de las tareas a llevar a cabo a lo largo de este *sprint*.
- **Investigar documentación para el sistema *OLAP* elegido (8h).** Tras la selección de la base de datos *OLAP* a utilizar para las tareas de análisis visual, se procederá a investigar su documentación para poder realizar un despliegue correcto de este sistema e integrarlo junto a los demás componentes del proyecto.
- **Desplegar e integrar el sistema *OLAP* (8h).** Tras consultar la documentación necesaria para poner en marcha el sistema *OLAP*, se procederá a realizar el despliegue y configuración de la base de datos para su correcta integración junto a los demás componentes del proyecto.
- **Investigar herramienta de visualización (4h).** Una vez que se tienen todos los datos extraídos, procesados, transformados e inferidos persistidos en el sistema *OLAP*, se encuentran listos para su puesta

en explotación. Para ello, se ha de investigar una herramienta de visualización compatible con la tecnología empleada y acorde a los requisitos de uso, que pueda explotar visualmente y de manera eficaz los datos obtenidos.

- **Investigar documentación para la herramienta de visualización elegida (8h).** Tras la selección de una herramienta de visualización compatible con el sistema *OLAP* elegido para las tareas de análisis visual, se procederá a investigar su documentación para poder realizar la integración y despliegue correctos, además de comprobar las posibilidades de configuración para las visualizaciones disponibles.

Sprint 9 (07/06/2023 – 21/06/2023)

Durante la duración de este *Sprint*, el objetivo principal será desplegar e integrar la herramienta escogida para la visualización de los datos obtenidos hasta el momento, además de la realización de algunas mejoras para los procesos de las anteriores etapas.

- **Gestión del *Sprint* (2h).** Se realizará el planteamiento de las tareas a llevar a cabo a lo largo de este *sprint*.
- **Aplicar correcciones sobre la memoria del proyecto (2h).** Se procederá a implementar las correcciones provistas a modo de *feedback* por el tutor.
- **Mejorar integración del sistema *OLAP* (4h).** Tras realizar la integración inicial con la herramienta *OLAP ClickHouse*, se procederá a mejorar la configuración de las interacciones con esta base de datos para expandir sus posibilidades de integración con los demás componentes del proyecto.
- **Mejora y automatización de pipeline *NLP* (8h).** Actualmente, el flujo de ejecución de las tareas *NLP* se realiza de manera manual. Para poder ser utilizado de manera óptima, se va a proceder a mejorar este pipeline para permitir automatizar su ejecución. Además, el *docker* correspondiente solamente necesita estar en ejecución mientras se ejecute esta parte del flujo, por lo que se va a asegurar que en cuanto se termine esta etapa se finalice también la ejecución del *docker*.
- **Desplegar e integrar herramienta de visualización (8h).** Tras consultar la documentación necesaria para poner en marcha la herramienta de visualización seleccionada, se procederá a realizar el

despliegue y configuración de *Apache Superset* para su correcta integración junto a los demás componentes del proyecto.

- **Crear diseño inicial de los dashboards (8h).** Tras desplegar y configurar *Apache Superset*, será necesario diseñar unos *dashboards* para poder explotar los datos obtenidos. Para conseguir este objetivo, se comenzará a bocetar un diseño inicial de la forma que podrían adoptar los datos para interpretarlos y obtener información de calidad a partir de los mismos.

Sprint 10 (21/06/2023 – 05/07/2023)

Durante la duración de este *Sprint*, el principal objetivo será el despliegue e integración de la herramienta *Apache Airflow* para gestionar la orquestación de procesos. Además de la realización de algunas mejoras para los procesos de las anteriores etapas para facilitar la tarea anterior, también se integrarán nuevas fuentes de datos.

- **Gestión del *Sprint* (2h).** Se realizará el planteamiento de las tareas a llevar a cabo a lo largo de este *sprint*.
- **Implementar *dashboards* diseñados (4h).** Tras realizar el diseño de los *dashboards* anteriormente, se procederá a realizar su implementación en la herramienta *Apache Superset*. Al realizar este proceso, es posible que se realicen ciertas modificaciones sobre los diseños iniciales para adecuarse mejor a las posibilidades de la herramienta y de los datos.
- **Investigar herramienta de orquestación de procesos (4h).** Debido a la compleja arquitectura que presenta este proyecto, es necesario un método para gestionar todo el flujo de acciones que se lleva a cabo a través de los distintos componentes. Para ello, se investigará una herramienta que permita la orquestación de todos los procesos a ejecutar.
- **Investigar documentación para la herramienta de orquestación de procesos (8h).** Tras la selección de la herramienta de orquestación de procesos, se procederá a investigar su documentación para poder realizar la integración y despliegue correctos de la misma junto a los demás componentes del sistema.
- **Implementar ajustes en tareas *Spark* (4h).** Para realizar la correcta integración y despliegue de *Apache Airflow* es necesario realizar

certas modificaciones en el funcionamiento del procesamiento de datos mediante *Apache Spark*.

- **Implementar ajustes en tareas *NLP* (4h).** Para realizar la correcta integración y despliegue de *Apache Airflow* es necesario realizar ciertas modificaciones en el funcionamiento de la inferencia mediante *NLP*.
- **Implementar ajustes en despliegue de *MongoDB* y *ClickHouse* (4h).** Para realizar la correcta integración y despliegue de *Apache Airflow* es necesario realizar ciertas modificaciones en el despliegue de *MongoDB* y del sistema *OLAP ClickHouse*.
- **Despliegue e integración de la herramienta de orquestación de procesos (8h).** Tras consultar la documentación pertinente de *Apache Airflow*, se procederá a realizar el despliegue y configuración del orquestador para su correcta integración junto a los demás componentes del proyecto.
- **Investigar nueva fuente de datos (4h).** Debido a los recientes cambios en las *APIs* de Twitter y Reddit, su uso en el proyecto se ha vuelto poco viable. Por ello, se investigará otra fuente de datos como alternativa a las planteadas inicialmente.
- **Análisis exploratorio de nueva fuente de datos (4h).** Al haber seleccionado un conjunto de datos ya existente como la nueva fuente de datos, será necesario realizar un análisis exploratorio para ver las características de los datos disponibles y su usabilidad.
- **Aplicar correcciones sobre la memoria del proyecto (2h).** Se procederá a implementar las correcciones provistas a modo de *feedback* por el tutor.
- **Integrar nueva fuente de datos (8h).** Tras investigar una nueva fuente de datos para el proyecto, será necesario proceder con su integración en las correspondientes etapas de extracción, procesamiento, inferencia, carga y visualización de los datos.
- **Documentar integración y despliegue de *Apache Airflow* (4h).** Tras realizar la correcta integración y despliegue de esta herramienta de orquestación de procesos, se procederá a documentar los aspectos relevantes de lo trabajado en esta parte.

A.3. Estudio de viabilidad

Viabilidad económica

Viabilidad legal

Apéndice B

Especificación de Requisitos

- B.1. Introducción
- B.2. Objetivos generales
- B.3. Catalogo de requisitos
- B.4. Especificación de requisitos

Apéndice C

Especificación de diseño

C.1. Introducción

En las siguientes secciones se detalla el diseño de los datos, además de los diseños procedimentales de la plataforma y los esquemas arquitectónicos empleados.

C.2. Diseño de datos

En los siguientes apartados se detallará el diseño de los modelos de datos utilizados en cada parte del proyecto y el proceso seguido para elaborar los cuadros de mando en los que se visualiza la información obtenida finalmente.

Diseño de los modelos de datos

En esta sección se explicarán los modelos de datos empleados entre los distintos componentes del proyecto. El esquema general utilizado para este proyecto ha sido un modelo de datos en estrella. Esta decisión de diseño se ha tomado por facilitar posteriormente en la herramienta de visualización el filtrado de datos, además de resultar en un modelo de datos que cumple las necesidades del proyecto sin necesidad de utilizar otro más complejo.

De esta manera, se toman las publicaciones como la tabla central de «hechos» y todas las demás tablas se han tomado como «dimensiones». Las dimensiones se relacionan con las publicaciones mediante dos tipos de claves «`tweet_id`» o «`user_id`».

Modelo de datos en MongoDB

La base de datos no relacional MongoDB ha efectuado el rol de *data lakehouse* en este proyecto. Al utilizar un modelo de datos basado en documentos *BSON*, ha sido perfecta para persistir los datos en brutos que llegan mediante la fase de extracción del proceso *ETL*. Esto se debe a que las *APIs* investigadas devolvían las respuestas en formato *JSON* con gran cantidad de campos anidados. Además, el conjunto de datos utilizado a modo de demostración también ha encajado a la perfección con la estructura orientada a documentos de MongoDB.

Sin embargo, esta base de datos no se ha utilizado para almacenar únicamente los datos en bruto, sino también los obtenidos tras la etapa de transformación del proceso *ETL*. De esta manera, se persisten en una colección¹ específica los datos ya procesados y limpios dentro de la misma base de datos.

Los datos almacenados en MongoDB se han almacenado siguiendo los siguientes criterios:

- **Cada fuente de datos cuenta con su propia base de datos.** Por lo que, al estar trabajando con dos orígenes de datos distintos, se han creado las bases de datos «*raw_dataset*» y «*raw_twitter*». De esta manera, se añade el prefijo «*raw_*» al nombre de cada origen de datos, que en este caso son el *dataset* de demostración utilizado y los datos provenientes de la *API* de TWitter.
- **Cada flujo de datos configurado en los conectores de Airbyte tiene como destino una colección.** Al configurar la conexión se elige el nombre que tendrá la tabla en el sistema destino (MongoDB en este caso). Para indicar que los datos se encuentran en un estado en bruto, se ha añadido en la configuración el prefijo «*airbyte_raw_*». De esta manera, los nombres de las colecciones han quedado «*airbyte_raw_movie*», «*airbyte_raw_season8*», etc.
- **Los datos ya procesados se almacenan en la misma base de datos en la que se han originado.** La agregación de los mismos entre distintas fuentes de datos se realiza posteriormente en ClickHouse. Sin embargo, al tener varias colecciones en bruto a partir del mismo origen de datos (los distintos conjuntos de datos particionados), los datos

¹Término que utiliza MongoDB para agrupar un conjunto de documentos, similar a las tablas en las bases de datos relacionales.

procesados se persistirán de manera común en las colecciones de datos procesados. Para indicar el estado de estos datos ya procesados se ha utilizado el prefijo «*clean_*». Por lo que las colecciones originadas tras la etapa de transformación se guarda en las colecciones «*clean_tweets*», «*clean_users*», etc.

En la [Figura C.1](#) se puede observar la estructura descrita con las relaciones entre las bases de datos y las colecciones, junto a la nomenclatura utilizada para designar cada uno de estos elementos.

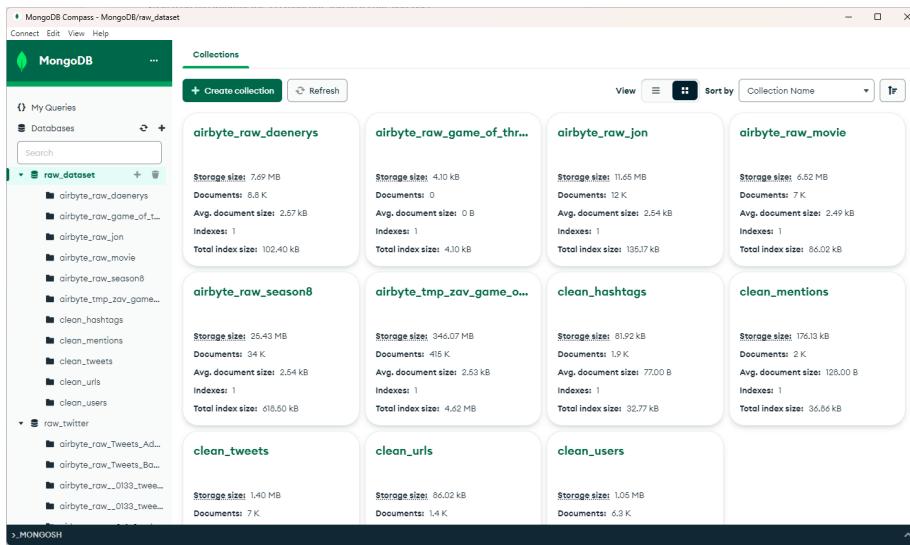


Figura C.1: Estructura de datos en MongoDB y nomenclatura de las colecciones

A continuación, se muestran las estructuras de datos empleadas en las colecciones de cada base de datos junto a los tipos de datos utilizados para cada atributo.

Nombre del campo	Tipo	Nullable
_airbyte_data	struct	true
- urls_expanded_url	string	true
- retweet_count	integer	true
- retweet_favorite_count	void	true
- ext_media_expanded_url	string	true
- country_code	string	true
- user_id	string	true
- bbox_coords	string	true
- quoted_favorite_count	void	true
- media_t.co	string	true
- quoted_source	void	true
- retweet_followers_count	void	true
- reply_to_status_id	string	true
- reply_to_user_id	string	true
- place_type	string	true
- retweet_friends_count	void	true
- coords_coords	string	true
- hashtags	string	true
- source	string	true
- place_full_name	string	true
- retweet_statuses_count	void	true
- retweet_created_at	void	true
- statuses_count	integer	true
- is_quote	boolean	true
- quoted_retweet_count	void	true
- retweet_name	void	true
- geo_coords	string	true

continúa en la página siguiente

continúa desde la página anterior

Nombre del campo	Tipo	Nullable
– media_expanded_url	string	true
– friends_count	integer	true
– text	string	true
– urls_url	string	true
– retweet_screen_name	void	true
– location	string	true
– protected	boolean	true
– screen_name	string	true
– profile_background_url	string	true
– quoted_text	void	true
– profile_expanded_url	string	true
– quoted_description	void	true
– retweet_location	void	true
– retweet_verified	void	true
– listed_count	integer	true
– quoted_name	void	true
– display_text_width	integer	true
– profile_url	string	true
– ext_media_t.co	string	true
– urls_t.co	string	true
– quoted_screen_name	void	true
– quoted_status_id	void	true
– ext_media_type	void	true
– profile_banner_url	string	true
– retweet_user_id	void	true
– quoted_user_id	void	true
– status_id	string	true
– media_url	string	true
– quoted_friends_count	void	true

continúa en la página siguiente

continúa desde la página anterior

Nombre del campo	Tipo	Nullable
– name	string	true
– profile_image_url	string	true
– retweet_description	void	true
– retweet_status_id	void	true
– quoted_created_at	void	true
– status_url	string	true
– quoted_followers_count	void	true
– account_created_at	string	true
– is_retweet	boolean	true
– reply_to_screen_name	string	true
– quoted_statuses_count	void	true
– verified	boolean	true
– url	string	true
– lang	string	true
– description	string	true
– created_at	string	true
– retweet_source	void	true
– account_lang	string	true
– country	string	true
– mentions_screen_name	string	true
– retweet_retweet_count	void	true
– retweet_text	void	true
– mentions_user_id	string	true
– quoted_location	void	true
– favourites_count	integer	true
– ext_media_url	string	true
– place_name	string	true
– quoted_verified	void	true
– media_type	string	true

continúa en la página siguiente

continúa desde la página anterior

Nombre del campo	Tipo	Nullable
– place_url	string	true
– favorite_count	integer	true
– followers_count	integer	true
_airbyte_data_hash	string	true
_airbyte_emitted_at	string	true
_id	struct	true
– oid	string	true

Tabla C.1: Esquema de datos para la base de datos «*dataset*», colecciones «*airbyte_raw_**»

Nombre del campo	Tipo	Nullable
_airbyte_data	struct	true
– data	struct	true
– author_id	string	true
– context_annotations	array	true
– domain	struct	true
– – description	string	true
– – id	string	true
– – name	string	true
– entity	struct	true
– – description	string	true
– – id	string	true
– – name	string	true
– created_at	string	true
– edit_history_tweet_ids	array	true
– entities	struct	true

continúa en la página siguiente

continúa desde la página anterior

Nombre del campo	Tipo	Nullable
– annotations	array	true
– start	integer	true
– end	integer	true
– probability	double	true
– type	string	true
– normalized_text	string	true
– cashtags	array	true
– start	integer	true
– end	integer	true
– tag	string	true
– hashtags	array	true
– start	integer	true
– end	integer	true
– tag	string	true
– mentions	array	true
– start	integer	true
– end	integer	true
– username	string	true
– id	string	true
– urls	array	true
– description	string	true
– display_url	string	true
– end	integer	true
– expanded_url	string	true
– images	array	true
– url	string	true
– width	integer	true
– height	integer	true
– media_key	string	true

continúa en la página siguiente

continúa desde la página anterior

Nombre del campo	Tipo	Nullable
– start	integer	true
– status	integer	true
– title	string	true
– unwound_url	string	true
– url	string	true
– geo	struct	true
– coordinates	struct	true
– type	string	true
– coordinates	array	true
– place_id	string	true
– id	string	true
– lang	string	true
– public_metrics	struct	true
– retweet_count	integer	true
– reply_count	integer	true
– like_count	integer	true
– quote_count	integer	true
– impression_count	integer	true
– text	string	true
– withheld	struct	true
– copyright	boolean	true
– country_codes	array	true
includes	struct	true
– places	array	true
– country	string	true
– country_code	string	true
– full_name	string	true
– geo	struct	true
– type	string	true

continúa en la página siguiente

continúa desde la página anterior

Nombre del campo	Tipo	Nullable
– bbox	array	true
– coordinates	double	true
_airbyte_data_hash	string	true
_airbyte_emitted_at	string	true
_id	struct	true
– oid	string	true

Tabla C.2: Esquema de datos para la base de datos «*raw_twitter*», colección «*airbyte_raw_movie*»

Los siguientes esquemas de datos son comunes a ambos orígenes de datos, a excepción de las colecciones «*clean_annotations*» y

«*clean_context_annotations*», que no se han podido construir a partir de la información del conjunto de datos de demostración. Los campos marcados con asterisco (*) tampoco se han podido obtener para el conjunto de datos, solamente con la información obtenida desde la *API* de Twitter.

Nombre del campo	Tipo	Nullable
tweet_id	string	true
source	string	false
search_query	string	false
user_id	string	true
created_at	string	true
language	string	true
text	string	true
retweet_count	integer	true
reply_count *	integer	true
like_count	integer	true

continúa en la página siguiente

continúa desde la página anterior		
Nombre del campo	Tipo	Nullable
quote_count *	integer	true

Tabla C.3: Esquema de datos para colección «*clean_tweets*»

Nombre del campo	Tipo	Nullable
user_id	string	true
name	string	true
username	string	true
description	string	true
location	string	true
followers_count	integer	true
following_count	integer	true
tweet_count	integer	true
listed_count	integer	true

Tabla C.4: Esquema de datos para colección «*clean_users*»

Nombre del campo	Tipo	Nullable
tweet_id	string	true
tag	string	true

Tabla C.5: Esquema de datos para colección «*clean_hashtags*»

Nombre del campo	Tipo	Nullable
tweet_id	string	true
user_id	string	true
username	string	true

Tabla C.6: Esquema de datos para colección «*clean_mentions*»

Nombre del campo	Tipo	Nullable
tweet_id	string	true
display_url	string	true

Tabla C.7: Esquema de datos para colección «*clean_urls*»

Nombre del campo	Tipo	Nullable
tweet_id	string	true
type	string	true
annotation	string	true

Tabla C.8: Esquema de datos para colección «*clean_annotations*»

Nombre del campo	Tipo	Nullable
tweet_id	string	true
domain	string	true
entity	string	true

Tabla C.9: Esquema de datos para colección «*clean_context_annotations*»

Modelo de datos en ClickHouse

La base de datos columnar ClickHouse ha efectuado el rol de *data warehouse* en este proyecto. Al utilizar un modelo de datos basado en columnas, resulta perfecta para el caso de uso que presentan las fuentes de datos utilizadas. Algunos de los campos de los distintos esquemas de datos descritos en la sección anterior tienen poca densidad. Por lo que, si se utilizara una base de datos relacional tradicional, todos esos campos estarían completamente vacíos en la mayoría de los registros, lo que repercutiría en la latencia de las consultas realizadas.

Sin embargo, al tratarse de un sistema *OLAP* (*On-Line Analytical Processing*), ClickHouse está expresamente diseñado para esta labor. El modelo de datos columnar que utiliza ayuda a reducir drásticamente el tiempo de ejecución de las consultas ya que emplea únicamente los campos necesarios para resolverlas, en lugar de iterar sobre cada uno como se haría en una base de datos relacional tradicional.

Respecto a los esquemas de datos empleados, todos los datos procesados e inferidos mediante las técnicas de *NLP* (*Natural Language Processing*) se

almacenar en una misma base de datos de nombre «*twitter*». Esta base de datos contiene toda la información relacionada con las fuentes de datos que tienen que ver con Twitter, que en el caso del proyecto son tanto los datos de la *API* como el *dataset* utilizado.

De esta manera, los registros de la colección «*clean_tweets*» de la base de datos «*raw_dataset*» que existen en MongoDB y los registros de la colección «*clean_tweets*» de la base de datos «*raw_twitter*», se vuelcan en la misma tabla «*tweets*» existente en ClickHouse. Este proceso es similar para los demás esquemas de datos y permite tener integrada en la misma tabla la información de todos los conjuntos de datos relacionados entre sí (campo «*source*») junto a los términos de consulta («*search_query*») por los que se ha extraído dicha información.

A continuación, se muestran los esquemas de datos utilizados para las tablas creadas en la base de datos ClickHouse.

Nombre del campo	Tipo de dato
tweet_id	String
source	String
search_query	String
user_id	String
created_at	DateTime
language	FixedString(2)
text	String
retweet_count	Int32
reply_count	Int32
like_count	Int32
quote_count	Int32
sentiment	String
emotion	String
topic	String
entity	String

Tabla C.10: Esquema de datos para tabla «*tweets*»

Nombre del campo	Tipo de dato
user_id	String
name	String
username	String
description	String
followers_count	Int32
following_count	Int32
tweet_count	Int32
listed_count	Int32

Tabla C.11: Esquema de datos para tabla «users»

Nombre del campo	Tipo de dato
tweet_id	String
tag	String

Tabla C.12: Esquema de datos para tabla «hashtags»

Nombre del campo	Tipo de dato
tweet_id	String
user_id	String
username	String

Tabla C.13: Esquema de datos para tabla «mentions»

Nombre del campo	Tipo de dato
tweet_id	String
display_url	String

Tabla C.14: Esquema de datos para tabla «urls»

Nombre del campo	Tipo de dato
tweet_id	String
type	String
annotation	String

Tabla C.15: Esquema de datos para tabla «*annotations*»

Nombre del campo	Tipo de dato
tweet_id	String
domain	String
entity	String

Tabla C.16: Esquema de datos para tabla «*context_annotations*»

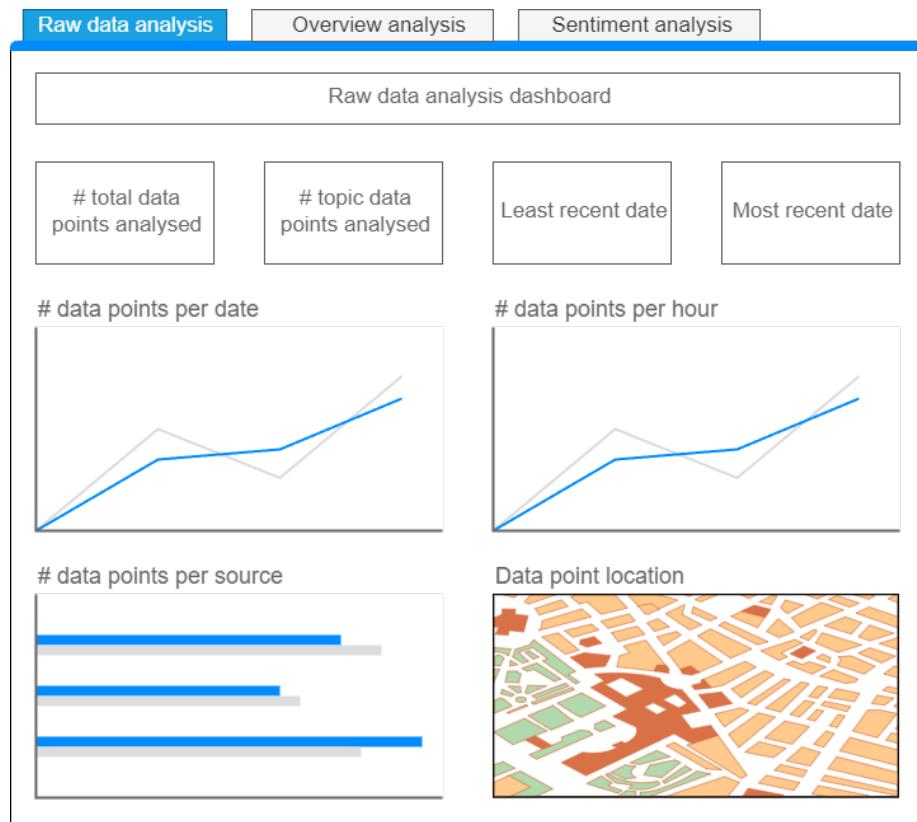
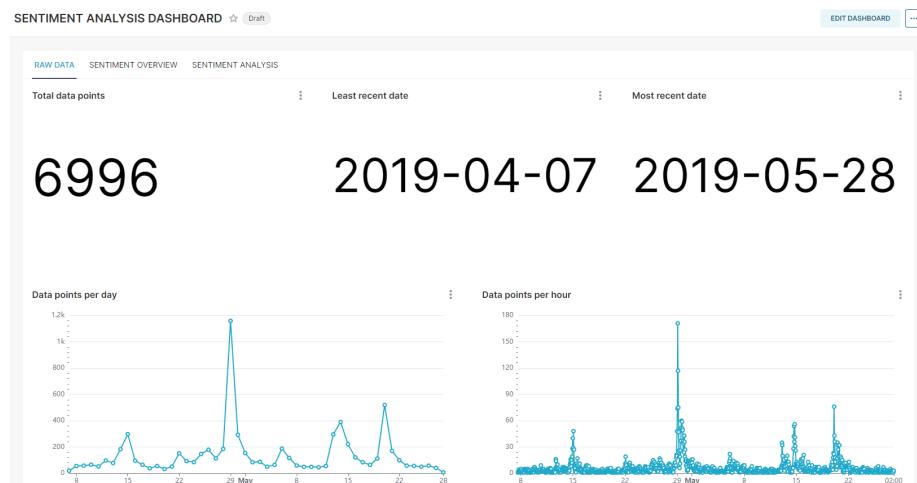
Diseños de los cuadros de mando

A continuación, se detalla la evolución seguida para el diseño de los cuadros de mando implementados en la herramienta *Apache Superset*. Se han diseñado e implementado un total de 5 vistas para el *dashboard* completo, las cuales muestran los datos procesados obtenidos a través del proceso *ETL* en diversos niveles de detalle e información.

Raw Data

La primera vista del cuadro de mando planea dar una visión general de los datos disponibles. Muestra unos indicadores con el número total de registros disponibles, así como la fecha más reciente y antigua que presentan los datos. Se plantean también dos gráficos de línea que muestran el número de registros extraídos cada día y en cada hora.

La idea inicial de esta primera pestaña del *dashboard* se puede observar en el boceto de la [Figura C.2](#), mientras que la implementación correspondiente se puede ver en la [Figura C.3](#).

Figura C.2: Diseño inicial de la pestaña *Raw data*Figura C.3: Primera iteración de la pestaña *Raw data*

Posteriormente, se hizo una segunda iteración sobre el diseño. Se mejoró la usabilidad y accesibilidad de los gráficos de línea añadiendo la posibilidad de hacer *zoom* en los datos. De esta manera, se puede observar en mayor detalle los registros en puntos específicos que puedan llamar la atención del usuario. Esta mejora se puede observar en la [Figura C.4](#).

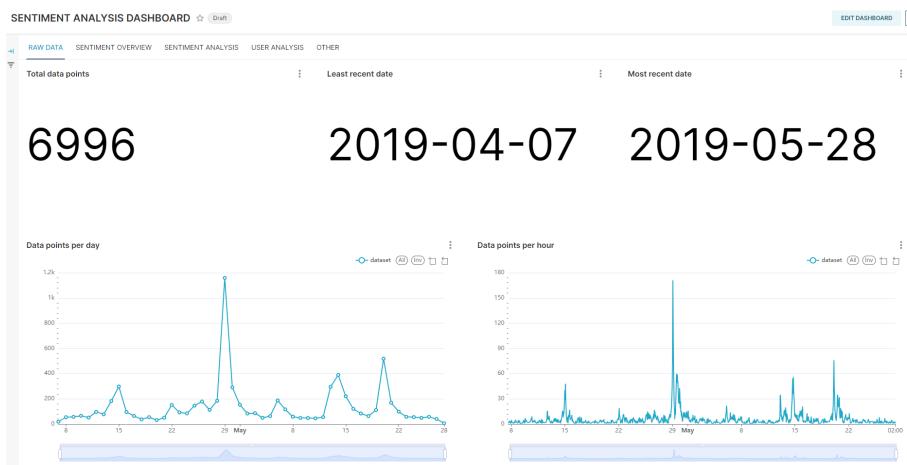


Figura C.4: Segunda iteración de la pestaña *Raw data*

Sentiment Overview

La segunda vista del cuadro de mando planea dar una visión general sobre los atributos inferidos a través de las técnicas de *NLP*. Se ideó unos indicadores para la puntuación media del sentimiento, así como el número de registros con sentimientos positivos, neutros y negativos. Debajo de estos indicadores, se mostrarían unos gráficos de barras indicando el número de registros totales asociados a cada sentimiento, emoción, tópico y entidad.

La idea inicial de esta segunda pestaña del *dashboard* se puede observar en el boceto de la [Figura C.5](#), mientras que la implementación correspondiente se puede ver en la [Figura C.6](#).



Figura C.5: Diseño inicial de la pestaña *Sentiment overview*

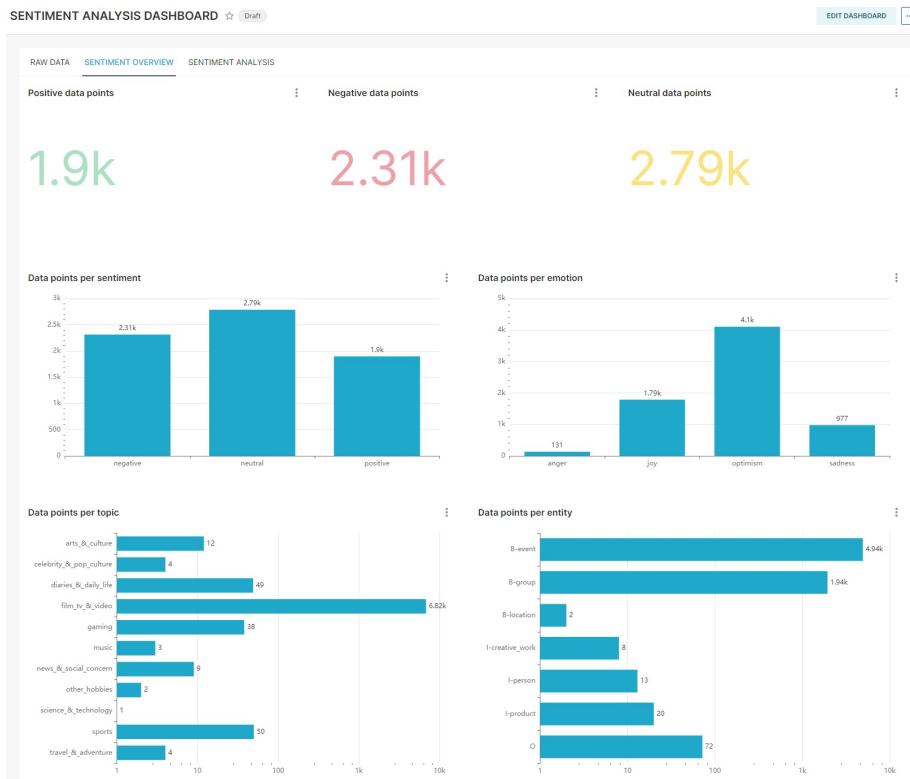


Figura C.6: Primera iteración de la pestaña *Sentiment overview*

Sentiment Analysis

La tercera vista del cuadro de mando entra a mayor detalle en el nivel de registros respecto a los sentimientos inferidos con los algoritmos de *NLP*. Se muestran unos gráficos lineales con el tipo de sentimiento (positivo, negativo o neutro) y emoción (optimismo, alegría, tristeza, enfado) asociada a nivel diario y en cuestión de la hora.

La idea inicial de esta tercera pestaña del *dashboard* se puede observar en el boceto de la [Figura C.7](#), mientras que la implementación correspondiente se puede ver en la [Figura C.8](#).

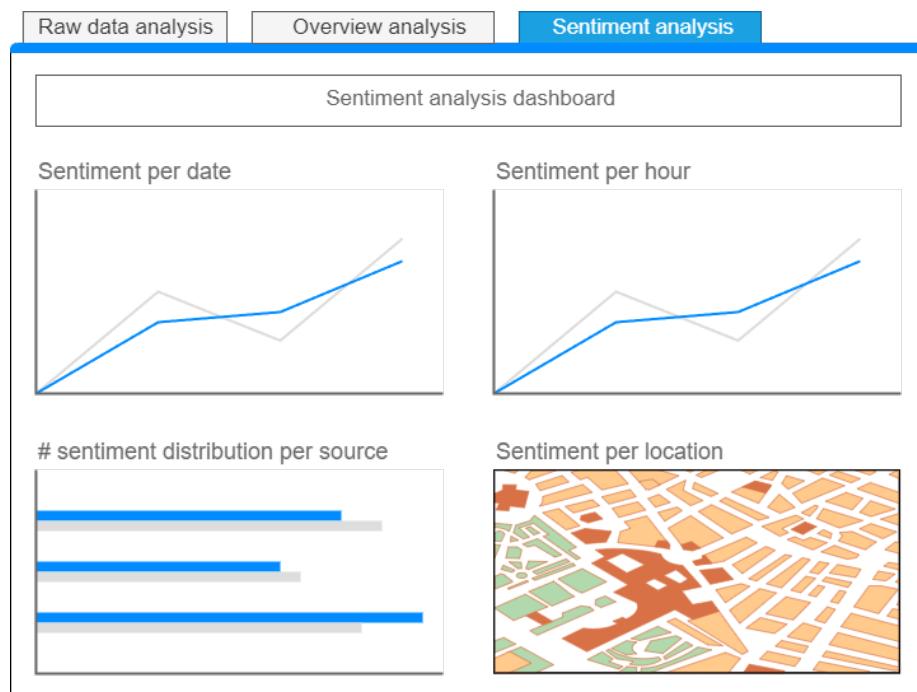


Figura C.7: Diseño inicial de la pestaña *Sentiment analysis*

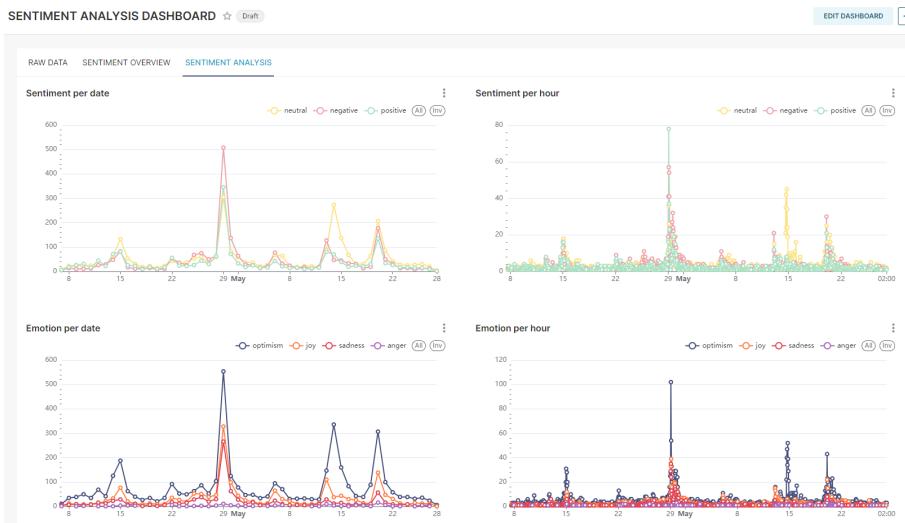


Figura C.8: Primera iteración de la pestaña *Sentiment analysis*

Posteriormente, se hizo una segunda iteración sobre este diseño (véase la [Figura C.9](#)). Se mejoró la accesibilidad de las gráficas que se encuentran en el nivel de hora añadiendo un *zoom* para poder observar en mayor detalle estos registros, además de eliminar las marcas de los puntos.

También se ha convertido la línea de puntos rígida que existía anteriormente a una un área ligeramente curvada para representar mejor los sentimientos y las emociones, puesto que estos términos representan naturalmente un flujo de combinaciones de las mismas y no se trata de algo estricto y exclusivo. Además se han modificado los color asociados a cada categoría con los que suelen estar representadas, realizando la siguiente asignación:

- Sentimiento
 - Positivo: Un color verde claro (#ACE1C4).
 - Neutro: Un color amarillo claro (#FDE380).
 - Negativo: Un color rojo claro (#EFA1AA).
- Emoción
 - Optimismo: Un color dorado (#FFD700).
 - Alegría: Un color verdoso (#90EE90).
 - Tristeza: Un color morado (#9370DB).
 - Enfado: Un color rojo intenso (#FF2400).

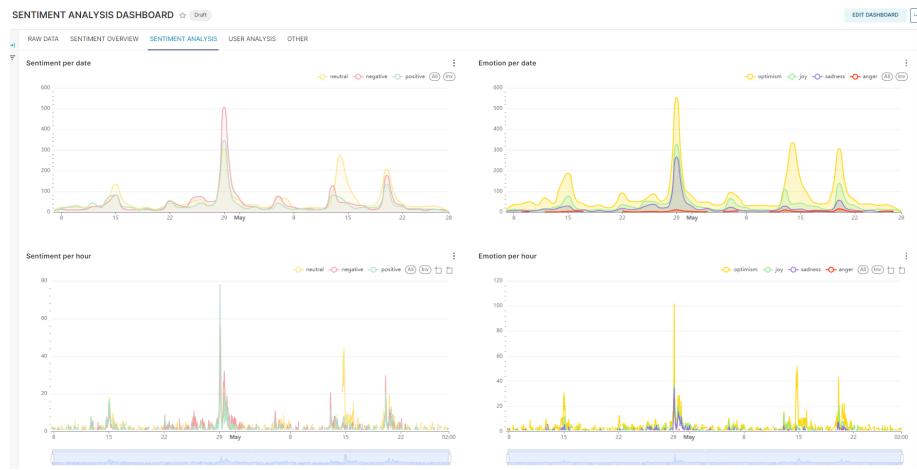
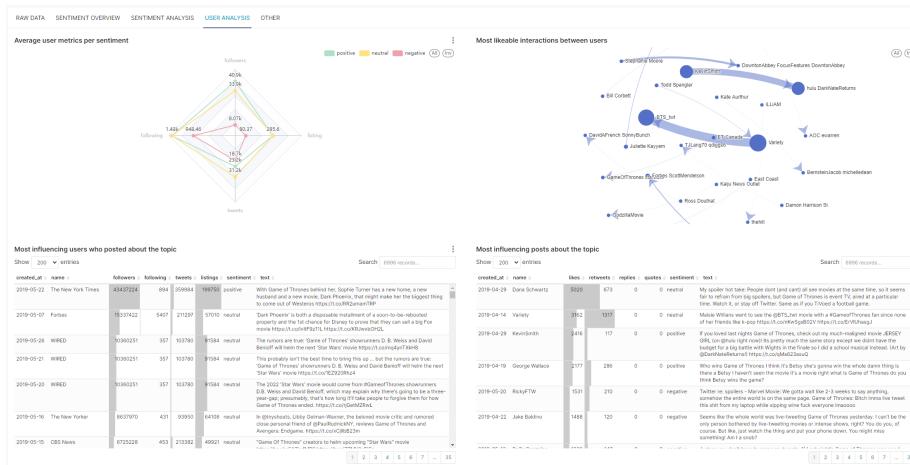


Figura C.9: Segunda iteración de la pestaña *Sentiment analysis*

User Analysis

La cuarta vista del cuadro de mando entra se centra en dar una visión a nivel de los usuarios que han realizado las publicaciones. En primer lugar, se muestran las estadísticas medias de cada usuario (número de seguidos, número de usuarios seguidos, total de publicaciones y número de listados) respecto al sentimiento con el que están mayormente asociado. De esta manera, se puede observar las diferencias entre los 3 tipos de usuarios y sus sentimientos notables respecto al tema de búsqueda.

La segunda visualización muestra un grafo con las interacciones de usuarios que más han gustado. Cada nodo origen representa al usuario que ha realizado la publicación y los nodos destino al usuario que mencionan, mientras que el tamaño de los mismos es calculado en base al número de *likes* que ha recibido la publicación. En la parte inferior del *dashboard* se han agregado dos tablas que muestran, respectivamente, los usuarios más influyentes que han publicado sobre el tema en concreto y las publicaciones más influyentes que han sido realizadas sobre el tema en concreto. En las tablas se muestran también las estadísticas de los usuarios y las publicaciones, respectivamente.

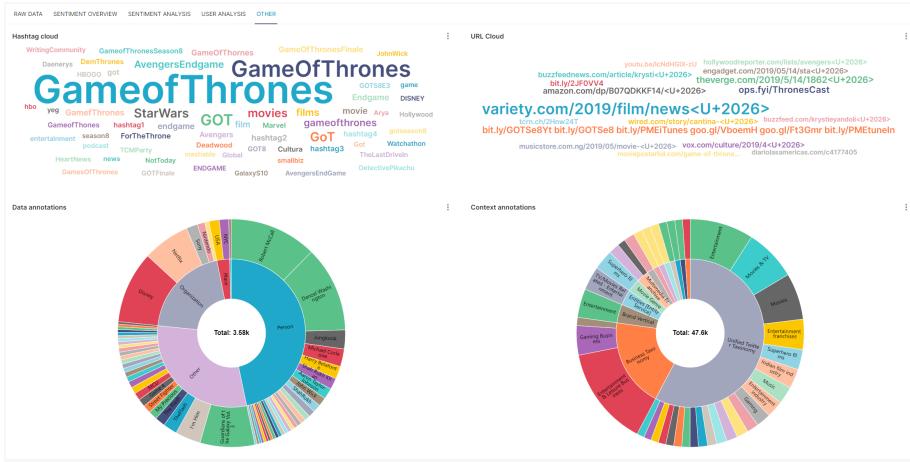
Figura C.10: Primera iteración de la pestaña *User analysis*

Other

La quinta vista del cuadro de mando se centra en las demás dimensiones no utilizadas en las anteriores vistas, como las tablas «*hashtags*», «*urls*», «*annotations*» y «*context_annotations*». Esto se debe a la disponibilidad parcial que presentan las dos últimas tablas en solamente una de las fuentes de datos, además de representar información en un menor grado de relevancia que las otras vistas diseñadas hasta el momento.

En primer lugar se muestran dos nubes de palabras, la primera con los *hashtags* más utilizados y la segunda con los enlaces más publicados. El tamaño de cada palabra es directamente proporcional al número de registros que la contienen. En la parte inferior se han creado dos gráficos tipo *sunburst* para las anotaciones y las anotaciones contextuales.

En las anotaciones se muestra en el círculo interior las categorías de las categorías principales de información detectada automáticamente en el texto (organización, persona, lugar, etc.), mientras que en el círculo exterior se muestra el nombre específico detectado. Las anotaciones contextuales siguen el mismo formato, en la parte interior se muestra la categoría de la entidad detectada y en la exterior el nombre de la subcategoría específica.

Figura C.11: Primera iteración de la pestaña *Other*

C.3. Diseño procedimental

En este apartado se describen las interacciones entre los principales componentes del proyecto. Para ello, se mostrarán una serie de diagramas de secuencias que representan el proceso *ETL*.

En la [Figura C.12](#) se muestra la interacción entre los componentes de la etapa de extracción de datos del proceso *ETL*. Esta interacción se realiza de manera automática si se programa la ejecución periódica de la *data pipeline* o cuando se ejecute de manera manual.

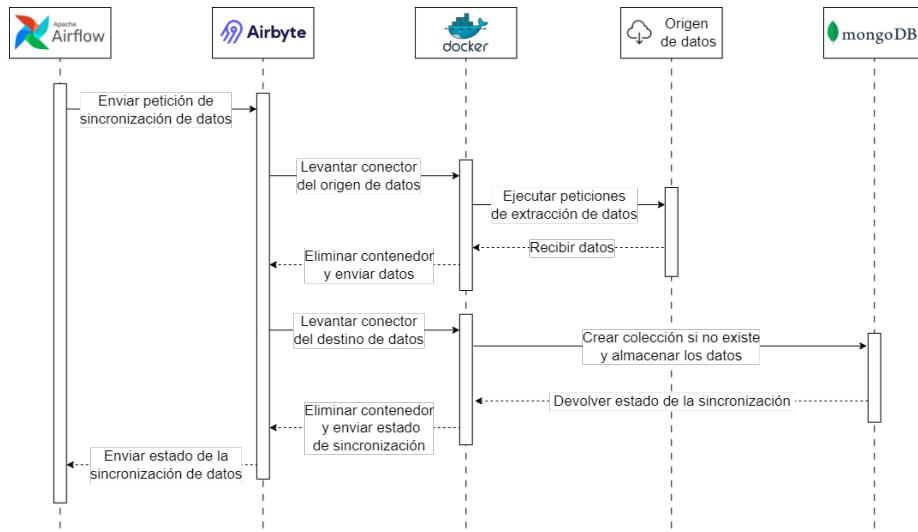


Figura C.12: Diagrama de secuencia de la interacción entre los componentes que forman parte de la etapa de extracción de datos del proceso *ETL*

En la [Figura C.13](#) y en la [Figura C.14](#) se muestra la interacción entre los componentes de la etapa de transformación del proceso *ETL*. El primer diagrama muestra el flujo de acciones que se realiza en la fase de procesamiento de los datos, mientras que el segundo diagrama muestra las acciones que se realizan en la fase de la inferencia *NLP*. Esta interacción se realiza de manera automática después de ejecutarse la etapa de extracción de datos ([Figura C.12](#)), si ha terminado satisfactoriamente.

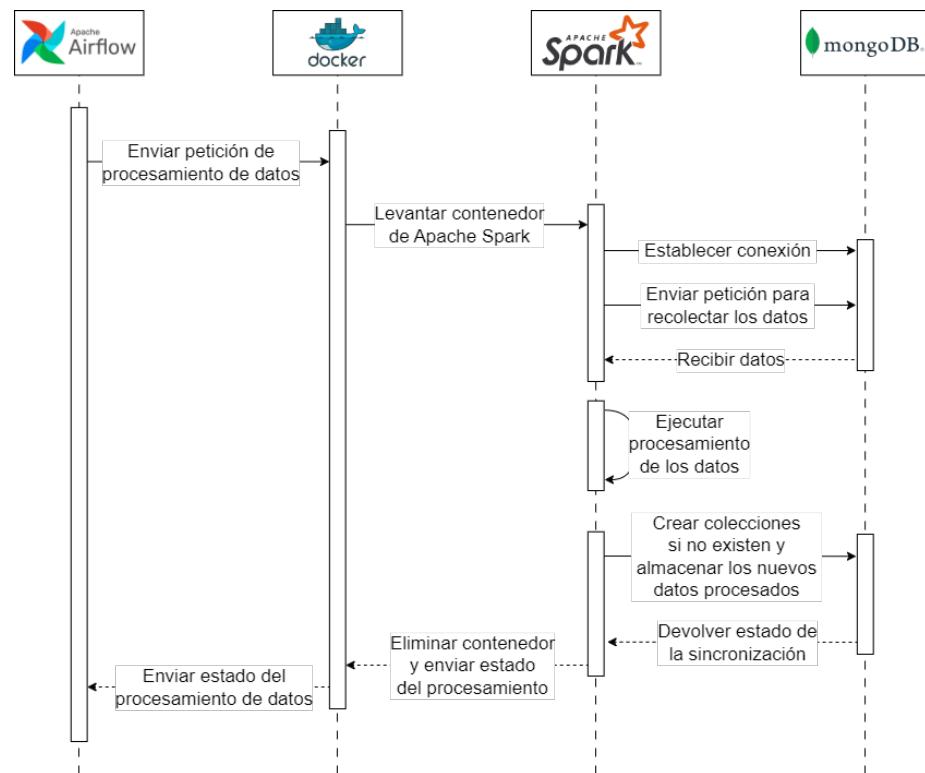


Figura C.13: Diagrama de secuencia de la interacción entre los componentes que forman parte de la fase de procesamiento de la etapa de transformación de datos del proceso *ETL*

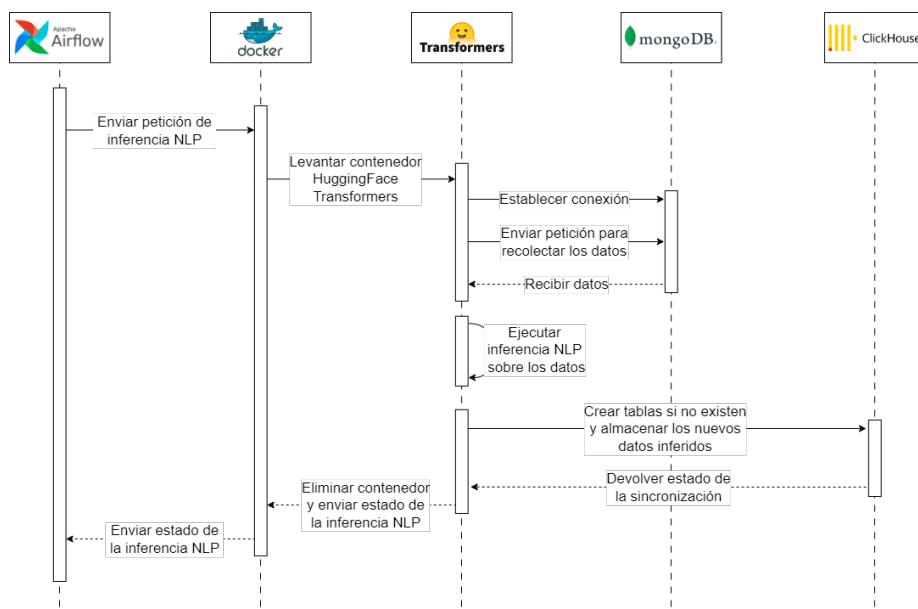


Figura C.14: Diagrama de secuencia de la interacción entre los componentes que forman parte de la fase de inferencia de la etapa de transformación de datos del proceso *ETL*

En la [Figura C.15](#) se muestra la interacción del usuario con el punto de acceso centralizado web creado para la plataforma. En este diagrama se indican las posibles acciones que se pueden tomar desde dicha interfaz web. Esta interacción se realiza en el momento que el usuario interactúa con la propia interfaz y es redirigido a las interfaces web de los demás componentes del proyecto.

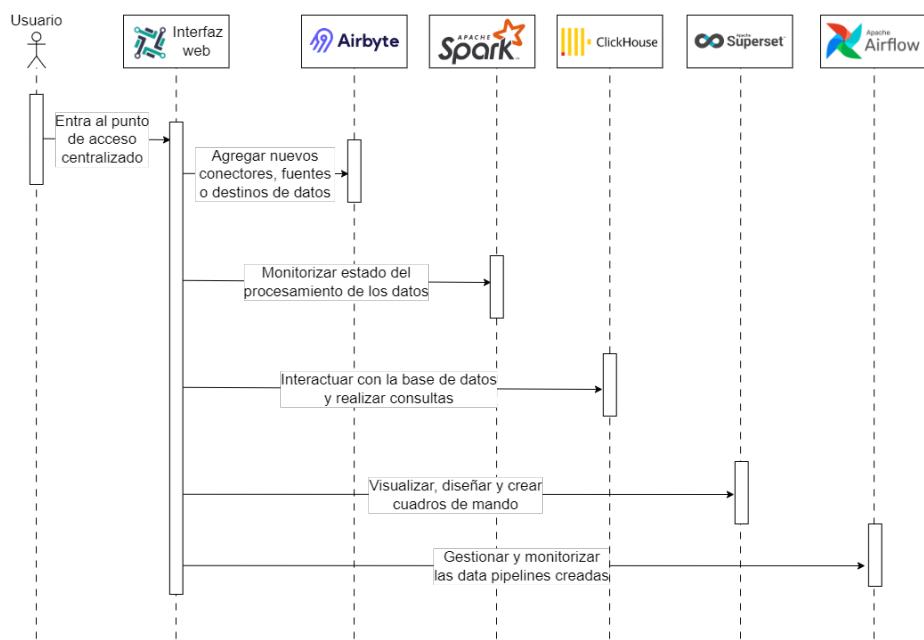


Figura C.15: Diagrama de secuencia de la interacción entre el usuario y los componentes a nivel general de la plataforma

C.4. Diseño arquitectónico

La plataforma está compuesta por diversos componentes encapsulados en contenedores *docker*, esto convierte la arquitectura de la plataforma en un diseño modular con la capacidad de intercambiar los diversos componentes sin afectar demasiado al resto.

Al tratarse de un proyecto enfocado al *Big Data*, las tecnologías utilizadas están diseñadas para ser escalables y distribuidas. No obstante, al haber realizado el desarrollo en una sola máquina, la interconexión de estos componentes entre ellos mismos se ha realizado a través de las redes *docker* creadas.

En la [Figura C.16](#) se puede observar la vista general de la arquitectura de la plataforma junto a las redes a las que pertenece cada componente.

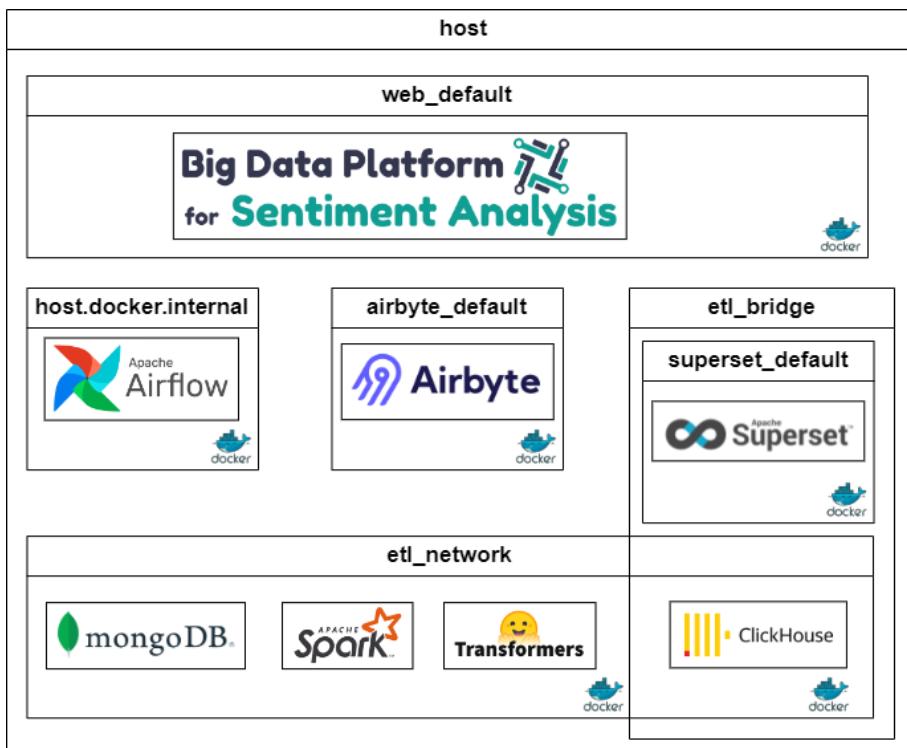


Figura C.16: Vista general de la arquitectura de la plataforma junto a las redes *docker* a las que pertenece cada componente

Las interacciones entre los componentes de la arquitectura y los flujos de datos y acciones se pueden observar en mayor detalle en la [Figura C.17](#).

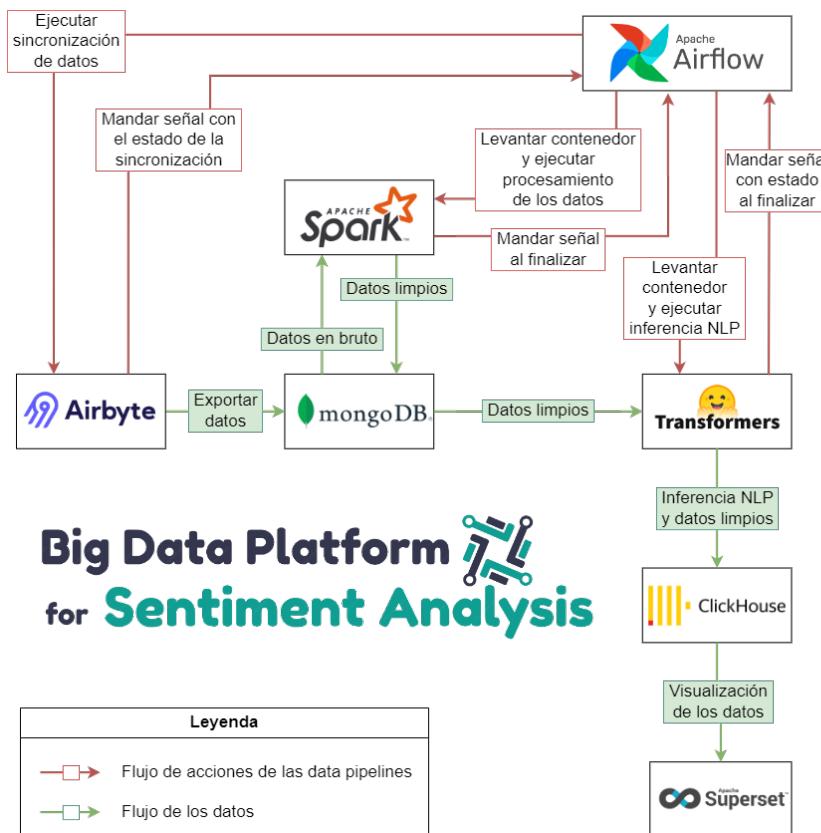


Figura C.17: Diagrama de secuencia de la interacción entre el usuario y los componentes a nivel general de la plataforma

Apéndice D

Documentación técnica de programación

D.1. Introducción

En esta sección se incluye la documentación técnica necesaria para entender la organización de directorios, la manera de instalar, configurar y ejecutar el proyecto, y los pasos a seguir para continuar con futuros desarrollos.

D.2. Estructura de directorios

La estructura del proyecto se ha organizado en la siguiente forma:

- **./** Carpeta raíz del proyecto, contiene el directorio de prototipos, las carpetas relevantes para desplegar cada componente del proyecto, la documentación del proyecto, el fichero de requisitos y los de información (*README*, licencia...).
- **./docs** Directorio principal de la documentación del proyecto, donde se encuentra la memoria y los anexos en formato L^AT_EX.
 - **./docs/img** Carpeta que contiene las imágenes utilizadas en la documentación.
 - **./docs/tex** Carpeta que contiene los ficheros *TEX* que conforman la documentación.

- **./prototypes** Directorio que contiene los desarrollos realizados a lo largo del prototipado del proyecto. La estructura de carpetas es la misma que la expuesta en los puntos siguientes.
- **./airflow** Directorio que contiene los ficheros de despliegue y configuración de Apache Airbyte, así como los archivos *dockerfile* y *docker-compose* para crear los contenedores del proyecto y levantar el *proxy* para securizar el *Docker socket*.
 - **./airflow/config** Carpeta que aparecerá una vez desplegado Apache Airflow en la que se pueden añadir configuraciones adicionales.
 - **./airflow/dags** Carpeta que contiene los archivos con las definiciones de los *DAGs* (*Directed Acyclic Graphs*) que crean las *data pipelines* implementadas y el orden de ejecución de las mismas.
 - **./airflow/logs** Carpeta que aparecerá una vez desplegado Apache Airflow en la que se guardan los registros de todas las ejecuciones de las *data pipelines* creadas.
 - **./airflow/plugins** Carpeta que aparecerá una vez desplegado Apache Airflow en la que se pueden añadir *plugins* adicionales.
- **./clickhouse** Directorio que contiene los ficheros de despliegue y configuración de ClickHouse.
- **./extraction** Directorio que contiene el fichero de descarga y configuración de la herramienta de extracción Airbyte.
- **./mongodb** Directorio que contiene los ficheros de despliegue y configuración de MongoDB.
- **./nlp** Directorio que contiene los ficheros de despliegue y configuración de *HuggingFace Transformers*.
 - **./nlp/nlp-tasks** Carpeta que contiene los ficheros Python que se encargan de la configuración y ejecución de las tareas de procesamiento de lenguaje natural (*NLP*).
- **./spark** Directorio que contiene los ficheros de despliegue y configuración de Apache Spark.
 - **./spark/spark-tasks** Carpeta que contiene los ficheros Scala que se encargan de la configuración y ejecución de las tareas de procesamiento y limpieza de datos.

- **./visualization** Directorio que contiene los ficheros de despliegue y configuración de Apache Superset.
- **./web** Directorio principal de la aplicación web *Flask* que se utiliza a modo de punto de acceso centralizado a las interfaces web de los componentes del proyecto..
 - **./app**
 - **./app/static** Contiene los ficheros estáticos utilizados para la aplicación web.
 - ◊ **./app/static/css** Carpeta con los archivos de estilos *CSS*.
 - ◊ **./app/static/js** Carpeta con los archivos de funciones *JavaScript*.
 - ◊ **./app/static/media** Carpeta con las imágenes utilizadas en aplicación web.
 - **./app/templates** Contiene las plantillas *HTML* de la aplicación web.

D.3. Manual del programador

En esta sección se describirán todos los elementos necesarios y la metodología a seguir para realizar futuros desarrollos.

Configuración de Airbyte

En este apartado se detallarán los métodos de configuración de la herramienta de extracción de datos.

Configuración del origen de datos

A continuación, se va a crear un ejemplo de fichero de configuración para una de las fuentes de datos seleccionada.

- Utilizando la API.
 1. Consultar la definición específica de la fuente de datos a configurar. Para ello es necesario consultar el *endpoint /v1/source_definitions/get*, lo que resultaría en una respuesta como la siguiente en el caso de escoger Twitter como fuente de datos (abreviada debido a su extensión real):

```
{
  "sourceDefinitionId": "...",
  "documentationUrl": "...",
  "connectionSpecification": {
    "type": "object",
    "title": "Twitter Spec",
    "$schema": "...",
    "required": [
      "api_key",
      "query"
    ],
    "properties": {...},
    ...
  },
  "jobInfo": {...}
}
```

2. **Enviar la petición de creación de fuente de datos.** Completando el *payload* de la petición hacia `/v1/sources/create` con las propiedades correspondientes obtenidas del paso anterior.
 3. **Comprobar la conexión con la fuente de datos.** Enviando una petición a `/v1/sources/check_connection`.
- **Utilizando la interfaz gráfica.** Navegando a la sección *Sources* y seleccionando el tipo de fuente a configurar, se muestra un formulario de edición que permite realizar las mismas operaciones ejecutadas anteriormente mediante la *API*. En la [Figura D.1](#) se puede observar dicho formulario.

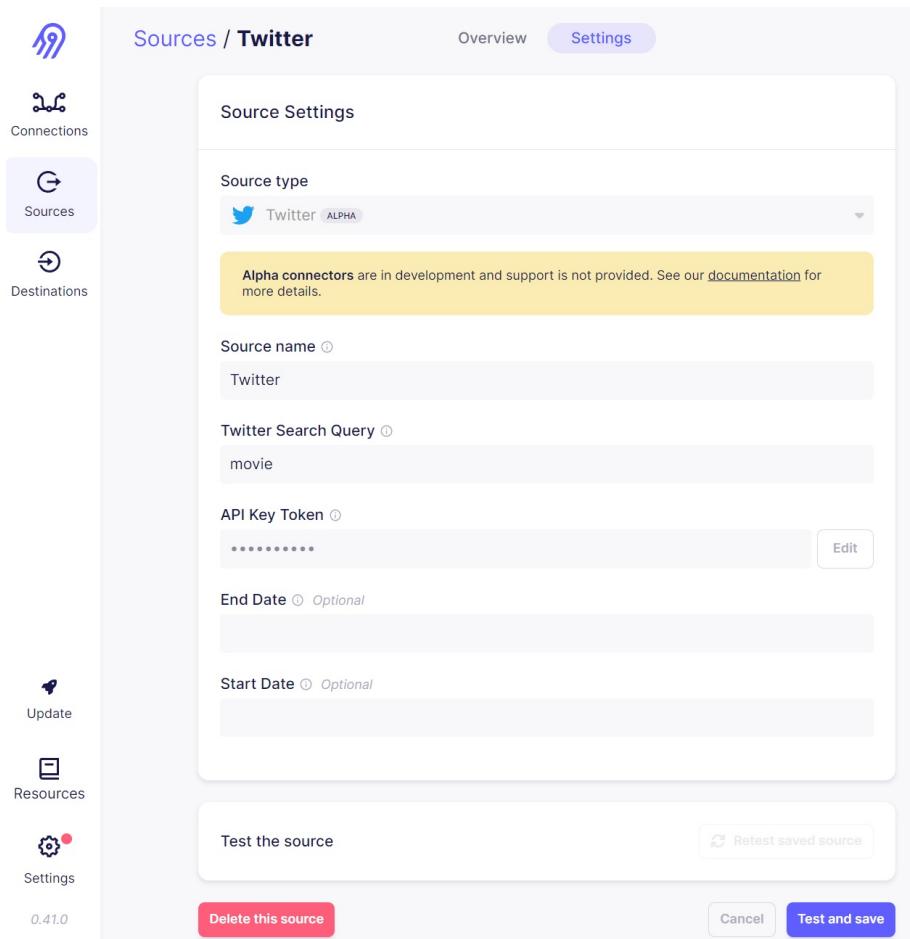


Figura D.1: Configuración del origen de datos en Airbyte: Twitter

Configuración del destino de los datos

A continuación, se va a crear un ejemplo de fichero de configuración para uno de los destinos de datos seleccionados.

- **Utilizando la API.**

1. **Consultar la definición específica del destino de datos a configurar.** Para ello es necesario consultar el *endpoint* `/v1/destination_definition_specifications/get`, lo que resultaría en una respuesta como la siguiente en el caso de escoger un fichero *CSV* local como destino de datos (abreviada debido a su extensión real):

```
{
    "destinationDefinitionId": "...",
    "documentationUrl": "...",
    "connectionSpecification": {
        "type": "object",
        "title": "CSV Destination Spec",
        "$schema": "...",
        "required": [
            "destination_path"
        ],
        "properties": {...},
        ...
    },
    "jobInfo": {...},
    "supportedDestinationSyncModes": [
        "overwrite",
        "append"
    ]
}
```

2. **Enviar la petición de creación de destino de datos.** Completando el *payload* de la petición hacia `/v1/destinations/create` con las propiedades correspondientes obtenidas del paso anterior.
 3. **Comprobar la conexión con el destino de datos.** Enviando una petición a `/v1/destinations/check_connection`.
- **Utilizando la interfaz gráfica.** Navegando a la sección *Destinations* y seleccionando el tipo de fuente a configurar, se muestra un formulario de edición que permite realizar las mismas operaciones ejecutadas anteriormente mediante la *API*. En la [Figura D.2](#) se puede observar dicho formulario.

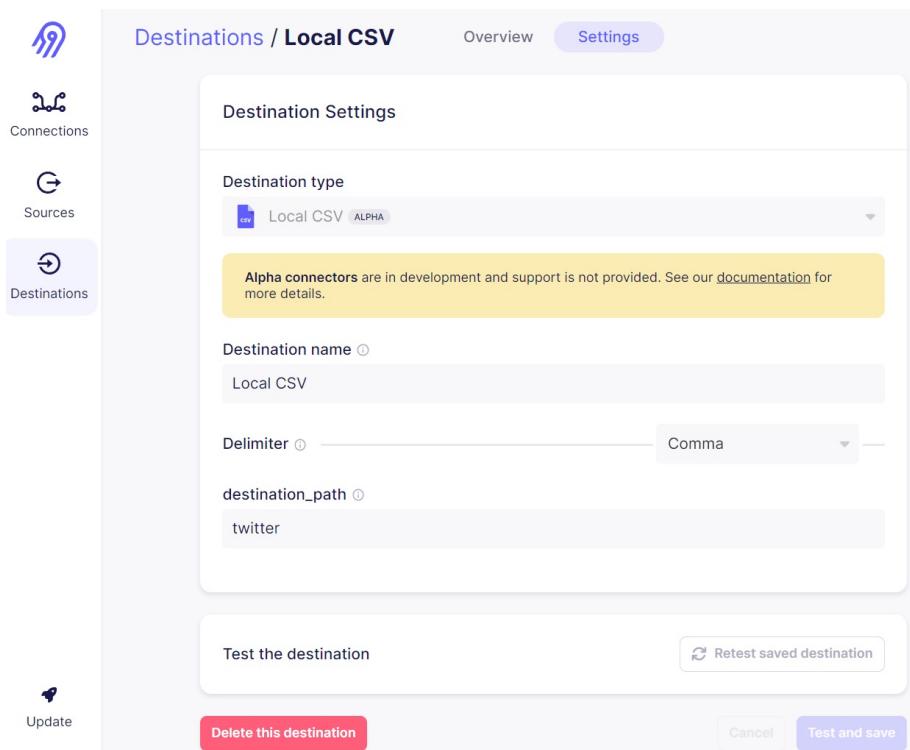


Figura D.2: Configuración del destino de datos en Airbyte: CSV Local

Configuración de la conexión entre fuente y destino

A continuación, se va a crear un ejemplo de fichero de configuración para una de las conexiones de datos seleccionada.

- Utilizando la API.

1. **Crear la conexión entre fuente y destino.** Para ello es necesario mandar una petición al endpoint `/v1/connections/create` con un *payload* como el siguiente (abreviado debido a que es bastante extenso):

```
{
  "name": "Twitter-API <> Local-CSV",
  "namespaceDefinition": "destination",
  "sourceId": "...",
  "destinationId": "...",
  "syncCatalog": {
```

```

    "streams": [
      {
        "stream": {
          "name": "...",
          "supportedSyncModes": [
            "full_refresh", "incremental"
          ]
        },
        "config": {
          "syncMode": "full_refresh",
          "destinationSyncMode": "append",
          "aliasName": "...",
          "selected": true
        }
      }
    ],
    "scheduleType": "manual",
    "status": "active",
    "geography": "auto",
    "notifySchemaChanges": true,
    "nonBreakingChangesPreference": "ignore"
  }
}

```

2. Ejecutar una sincronización manual de la conexión. Mandando una petición hacia `/v1/connections/sync`.

- Utilizando la interfaz gráfica. Navegando a la sección *Connections* y seleccionando las fuentes y destinos de datos para los que configurar la conexión. Se muestra un formulario de edición que permite realizar las mismas operaciones ejecutadas anteriormente mediante la *API*. En la [Figura D.3](#) se puede observar dicho formulario.

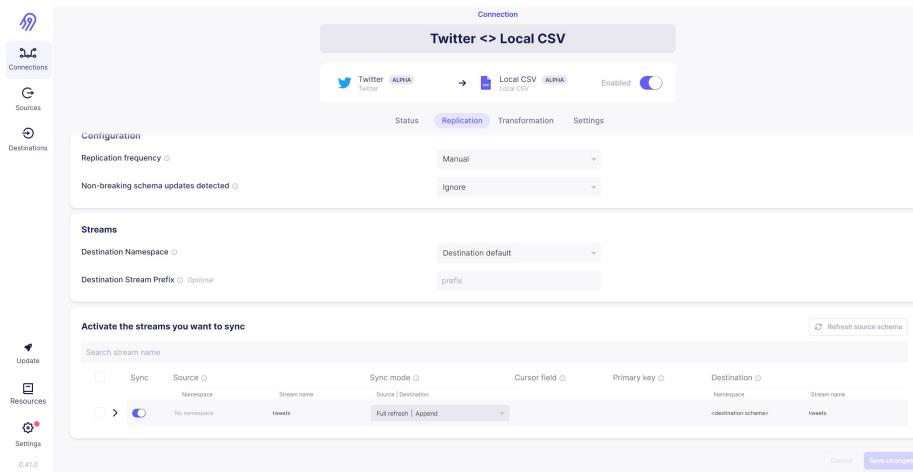


Figura D.3: Configuración de la conexión entre origen y destino de los datos en Airbyte

Creación de nuevos conectores

La herramienta de extracción Airbyte permite la creación de nuevos conectores que utilicen *APIs REST* de forma rápida y sencilla directamente desde la interfaz gráfica. Para ello, es necesario acceder a la siguiente sección de la herramienta: http://<AIRBYTE-EXAMPLE.COM>/workspaces/<WORKSPACE_ID>/connector-builder/.

En esta vista se pueden crear y configurar nuevos conectores para *APIs* bien de manera visual ([Figura D.4](#)) o mediante un esquema en formato *YAML* ([Figura D.5](#)).

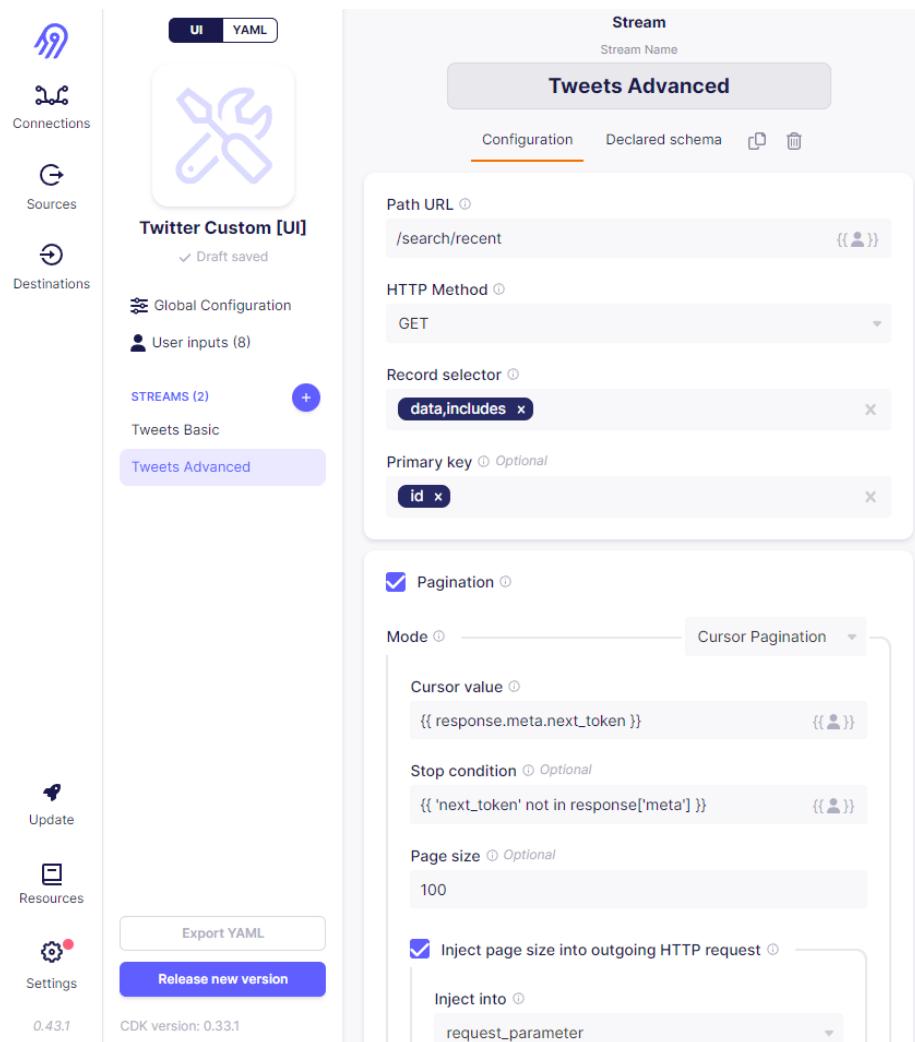
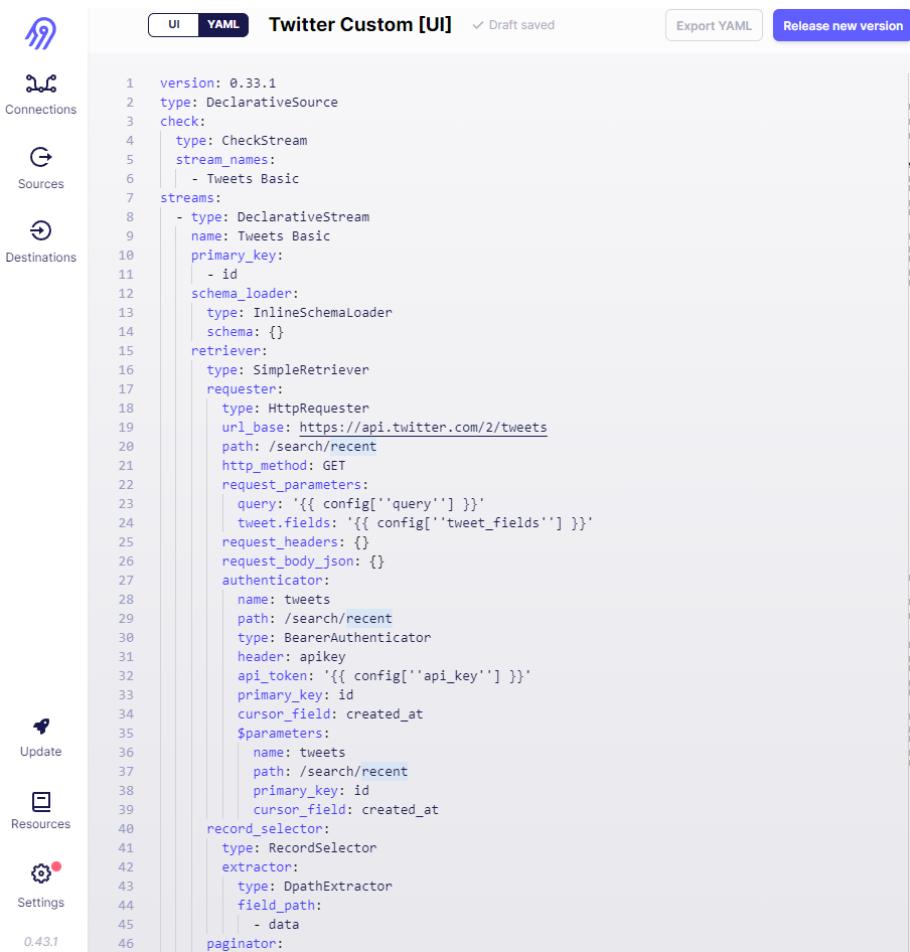


Figura D.4: Interfaz gráfica de la vista para creación y configuración de nuevos conectores para Airbyte



The screenshot shows the Airbyte UI interface for creating and configuring connectors. On the left, there's a sidebar with icons for Connections, Sources, Destinations, Update, Resources, and Settings. The main area is titled "Twitter Custom [UI]" and displays the following YAML configuration:

```
version: 0.33.1
type: DeclarativeSource
check:
  type: CheckStream
  stream_names:
    - Tweets Basic
streams:
  - type: DeclarativeStream
    name: Tweets Basic
    primary_key:
      - id
    schema_loader:
      type: InlineSchemaLoader
      schema: {}
    retriever:
      type: SimpleRetriever
      requester:
        type: HttpRequester
        url_base: https://api.twitter.com/2/tweets
        path: /search/recent
        http_method: GET
        request_parameters:
          query: '{{ config["query"] }}'
          tweet.fields: '{{ config["tweet_fields"] }}'
        request_headers: {}
        request_body_json: {}
      authenticator:
        name: tweets
        path: /search/recent
        type: BearerAuthenticator
        header: apikey
        api_token: '{{ config["api_key"] }}'
        primary_key: id
        cursor_field: created_at
      $parameters:
        name: tweets
        path: /search/recent
        primary_key: id
        cursor_field: created_at
    record_selector:
      type: RecordSelector
      extractor:
        type: DpathExtractor
        field_path:
          - data
    paginator:
```

Figura D.5: Esquema *YAML* de la vista para creación y configuración de nuevos conectores para Airbyte

Procesamiento de los datos

El procesamiento de los datos se ha realizado mediante *scripts* en lenguaje Scala para el sistema Apache Spark, por lo que será necesario tener conocimientos de estas dos herramientas para poder llevar a cabo mayores desarrollos en esta parte.

Se ha de configurar un único archivo de procesamiento a ejecutar por cada origen de datos creado, aunque se pueden incluir dependencias como funciones comunes a varios archivos. Los *scripts* pueden seguir la estructura de ejemplo en los ficheros `transform_twitter.scala` y `transform_dataset.scala`.

La ejecución de ambos archivos deberá poder realizarse mediante la `spark-shell`, a la que se le han de pasar los parámetros los parámetros de conexión para las bases de datos de origen y destino de los datos. Además, de los parámetros necesarios para identificar y filtrar los datos concretos a extraer y procesar.

Inferencia *NLP*

La inferencia mediante técnicas de procesamiento de lenguaje natural (*NLP*) se ha realizado mediante la librería *HuggingFace Transformers*. Esta librería permite la utilización de *LLMs* (*Large Language Models*) ya preentrenados para estas tareas *NLP*.

Para añadir más tareas a la *pipeline* de inferencia de datos, se han de modificar los siguientes archivos:

- El archivo `connectors.py` si se van a añadir nuevos conectores para distintos orígenes o destinos de datos.
- El archivo `tasks.py` para añadir las clases correspondientes con todos los parámetros necesarios para configurar las conexiones de entrada, salida, nombre de las tablas de los datos, etc. En el mismo archivo se añadirán nuevos métodos `task_*`, uno por cada tarea o modelo *NLP* a emplear.
- El archivo `pipeline.py` para añadir las nuevas opciones de ejecución para los nuevos orígenes o destinos de datos.

Visualización de los datos

La herramienta de visualización Apache Superset presenta una interfaz intuitiva y sencilla de utilizar para la agregación de nuevas fuentes de datos y la creación de gráficos interactivos.

En la pestaña *Datasets* se pueden configurar conexiones a nuevos orígenes de datos. Posteriormente, desde la pestaña *SQL Lab* se pueden realizar consultas en *SQL* nativo sobre los *datasets* importados.

A parte de los datos importados, los resultados de las consultas realizadas en *SQL Lab* se pueden guardar en *datasets* virtuales para poder utilizarlos posteriormente también como base para la creación de nuevos gráficos.

En la pestaña *Charts* se pueden diseñar e implementar nuevas visualizaciones que añadir posteriormente a los *dashboards* creados. El proceso presenta la selección de un conjunto de datos a utilizar para cada gráfico creado y un tipo de visualización que se puede configurar y personalizar posteriormente.

Orquestación de los datos

La orquestación de los datos se ha realizado mediante la herramienta Apache Airflow. Esta herramienta permite definir grafos acíclicos dirigidos (*DAGs*) para diseñar los flujos de trabajo a ejecutar.

En la carpeta `dags` existente dentro del directorio respectivo de la herramienta se encuentran los *DAGs* diseñados para la plataforma. Cada fuente de datos deberá tener su *DAG* propio.

Estos grafos acíclicos dirigidos se pueden crear de manera estática (véase el archivo `twitter_pipeline_dag.py`) o de manera dinámica (véase el archivo `dataset_pipeline_dag.py`). De esta manera, al necesitar crear cierto número de flujos de datos que utilicen la misma configuración base pero cambiando solamente los parámetros de entrada, se puede realizar la configuración de manera dinámica para añadir más claridad al código y facilitar la comprensión del mismo.

Los *DAGs* están formados por «actuadores», que realizan las acciones, y por «sensores», que esperan una confirmación para continuar con el flujo de trabajo. las *data pipelines* diseñadas tienen la siguiente estructura:

- Actuador Airbyte: Ejecutar sincronización de los datos.

- Sensor *Airbyte*: Esperar la confirmación con el estado de la sincronización de los datos.
- Actuador *Docker (Spark)*: Desplegar contenedor, ejecutar procesamiento de los datos y eliminar contenedor.
- Actuador *Docker (NLP)*: Desplegar contenedor, ejecutar inferencia *NLP* y eliminar el contenedor.

Cada tipo de tarea tiene su propia actuador o sensor. Airflow presenta actuadores listo para utilizar para algunas de las acciones más comunes como ejecutar comandos *bash*, enviar correos electrónicos, realizar peticiones *HTTP*, ejecutar código Python, etc.

D.4. Compilación, instalación y ejecución del proyecto

En los siguientes apartados se desarrollan los requisitos necesarios para la instalación y despliegue del proyecto.

Especificaciones técnicas recomendadas

Esta «Plataforma *Big data* para *Sentiment Analysis*» se ha desarrollado en un entorno local y empleando una sola máquina. No obstante, al ser una plataforma modular, escalable y distribuida, en un entorno de producción real se debería realizar el despliegue de cada componente en una máquina o clúster dedicado. De esta manera, se evitaría la sobrecarga del sistema, se aumentaría la seguridad de la plataforma y se eliminarían las limitaciones en el uso de recursos por cada componente.

En caso de realizar el despliegue en una sola máquina, y teniendo en cuenta las consideraciones anteriores, se recomiendan las siguientes especificaciones técnicas para la misma:

- **Sistema operativo:** Windows 11 (22H2) / Ubuntu 22.04.2 LTS
- **CPU:** Intel(R) Core(TM) i5-13600K 3.50 GHz
- **Memoria RAM:** 32GB DDR4 3.2GHz
- **GPU:** NVIDIA GeForce RTX 3060 12GB

El punto más notable de esta plataforma sería los requisitos de memoria *RAM* que presenta. La gran cantidad de componentes que se despliegan en contenedores *Docker* establece el consumo en reposo de la plataforma en unos 13 GB (véase Figura D.6). Este consumo aumenta hasta los 18GB al momento de ejecutar la etapa de transformación del proceso *ETL* (*Extract, Transform, Load*), por lo que sería recomendable cumplir esta recomendación para evitar caídas del sistema ante altos niveles de carga.

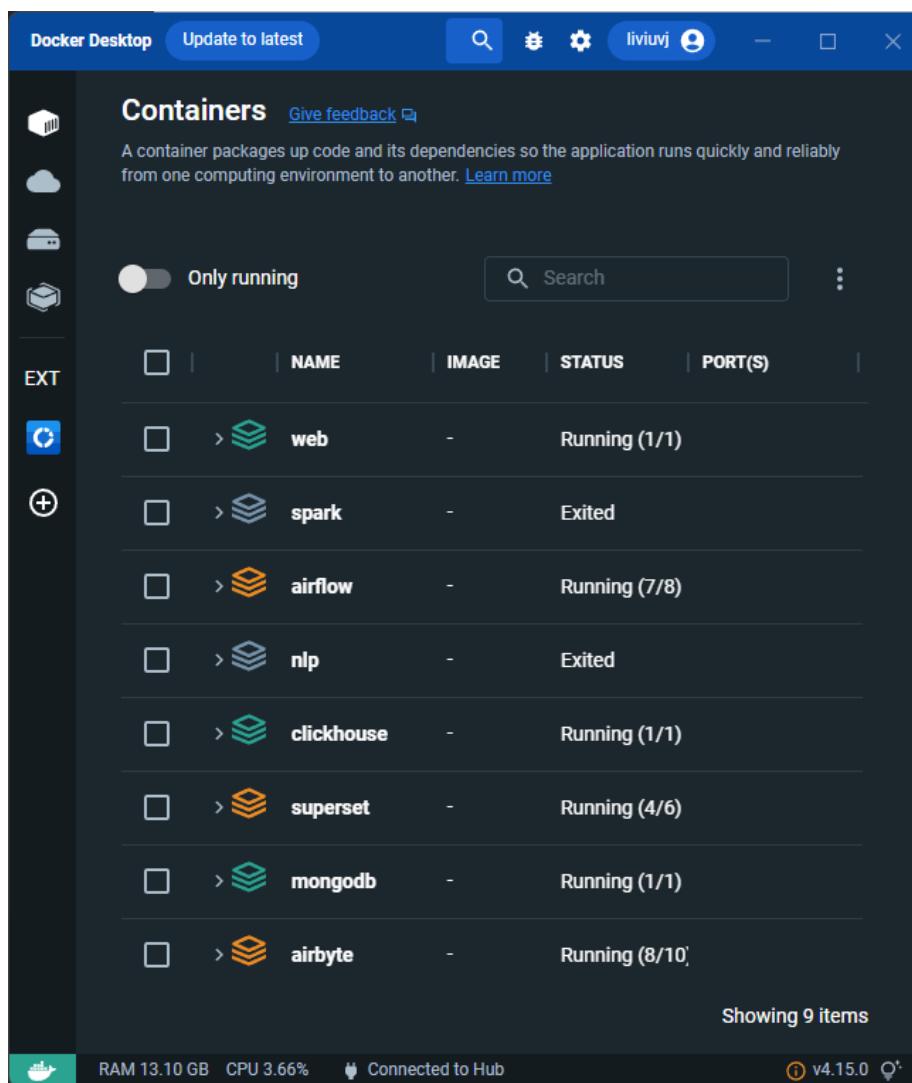


Figura D.6: Consumo de memoria RAM con la plataforma «en reposo»

Respecto a la GPU, realmente no se necesita para la ejecución normal de la inferencia *NLP* ya que actualmente dichos modelos se ejecutan en la

CPU. No obstante, realizar estas tareas en una GPU de alta gama permitiría reducir notablemente los tiempos de ejecución de esta fase del proceso *ETL*. También sería útil en gran medida en el caso de querer realizar el entrenamiento de modelos propios sobre cada flujo de datos.

Dependencias *software*

A continuación, se describen las versiones de los componentes *software* empleados para la realización del proyecto:

- Airbyte 0.43.1
- Apache Airflow 2.6.2
- Apache Spark 3.2.3
- MongoDB 6.0
- ClickHouse 22.1.3.7
- Apache Superset 2.1.0
- HuggingFace Transformers 4.30.1
- Python 3.10.6
- Flask 2.3.2

Instalación

Para la instalación del proyecto tan solo es necesario clonar localmente el código del repositorio *Github*: <https://github.com/liviuvj/sentiment-analysis-platform>

Posteriormente, es necesario instalar las dependencias que se encuentran en el archivo de requisitos. Para ello, es buena práctica crear primero un entorno virtual para evitar problemas de versiones posteriormente y realizar la instalación del fichero de requisitos.

```
python3 -m venv venv
source ./venv/bin/activate
pip install -r requirements.txt
```

Compilación

Este proyecto está compuesto en su mayor parte por contenedores *Docker* y código en *scripts* de Python y Scala, por lo que no es necesaria la compilación del mismo. Al ser lenguajes de *scripting* y ficheros de configuración, no resulta necesaria su compilación.

Ejecución

Para la ejecución del proyecto se ha creado un *script bash* que se encarga de realizar la descarga de ficheros adicionales, clonación de repositorios, despliegue de contenedores *Docker* y configuración de los componentes que forman parte del proyecto.

Por lo que tan solo será necesario ejecutar dicho *script bash*:

```
./quickstart.sh
```

No obstante, cada directorio principal correspondiente a las herramientas empleadas para la realización de este proyecto contiene un fichero `.env`. Dicho archivo contiene los parámetros de configuración por defecto, así como usuarios y contraseñas, de los distintos componentes de la plataforma. Por lo que sería recomendable modificar dichos valores en un entorno de producción.

Una vez levantados y configurados todos los componentes del proyecto, se puede acceder las distintas interfaces web disponibles para gestionar o monitorizar las herramientas. Las *URLs* especificadas muestran los puertos asignados por defecto con la configuración realizada.

- **Punto web de acceso centralizado.** Página web ligera que permite el acceso directo a todas las demás interfaces web que presentan los componentes de la plataforma. Accesible en <http://localhost:5000>
- **Apache Airbyte.** Plataforma desde la que se pueden crear, gestionar y monitorizar las conexiones de sincronización de datos entre origen y destino. Accesible en <http://localhost:8000>
- **Apache Spark.** Monitorización de los recursos utilizados por Spark. Accesible en <http://localhost:4040>

- **ClickHouse.** Sencilla interfaz web que permite la ejecución de consultas directas sobre la base de datos en lenguaje *SQL*. Accesible en <http://localhost:8123/play>
- **Apache Superset.** Aplicación en la que se pueden diseñar, implementar, visualizar y compartir cuadros de mando interactivos. Accesible en <http://localhost:8088>
- **Apache Airflow.** Accesible en <http://localhost:8080>

D.5. Pruebas del sistema

A lo largo de este proyecto se han realizado diversas tareas de integración entre los distintos componentes que conforman la plataforma desarrollada. No obstante, no se han implementado unas pruebas unitarias o de integración de manera formal.

Esto se debe a que la mayoría de los componentes utilizados ya presentan sus propias pruebas específicas para la validación, integración y funcionamiento utilizados. Por ello, se ha decidido por la creación de un *script bash* para realizar el despliegue y configuración automáticos de la plataforma. En caso de encontrarse algún fallo en alguna de las tecnologías empleadas, la propia herramienta será capaz de identificar la configuración errónea y notificar al usuario del fallo detectado.

Apéndice E

Documentación de usuario

E.1. Introducción

En las secciones siguientes se incluye la documentación necesaria dar a los usuarios del proyecto las nociones necesarias y la manera de instalar, configurar, ejecutar y utilizar la plataforma desarrollada.

La definición de usuario final para esta plataforma cubre dos roles principales. El primero, un usuario sin conocimientos técnicos que solamente va a visualizar los datos finales a través del *dashboard* diseñado. Y el segundo, un analista de datos que se va a encargar también de la exploración de los datos y de gestionar e implementar los cuadros de mando para la plataforma.

E.2. Requisitos de usuarios

Esta «Plataforma *Big data* para *Sentiment Analysis*» se ha desarrollado en un entorno local y empleando una sola máquina. No obstante, al ser una plataforma modular, escalable y distribuida, en un entorno de producción real se debería realizar el despliegue de cada componente en una máquina o clúster dedicado.

Para los usuarios finales, estos tendrían que acceder simplemente a través de las interfaces web correspondientes a los componentes necesarios. Por lo que los requisitos para estos usuarios serían los siguientes:

- Disponer de una conexión con acceso a la red en la que se haya realizado el despliegue de esta «Plataforma *Big data* para *Sentiment Analysis*».
- Tener asignados los roles o disponer de los permisos y credenciales necesarios para acceder a las interfaces web de las herramientas correspondientes (en este caso, Apache Superset).
- Disponer de los conocimientos necesarios para interpretar los datos visualizados. En caso de ser un analista de datos, necesita también conocer las técnicas de análisis de datos respectivas para llevar a cabo el diseño e implementación de *dashboards*.

E.3. Instalación

Como se ha mencionado anteriormente, al tratarse de una plataforma *Big Data*, esta herramienta idealmente se estaría ejecutando sobre un clúster dedicado para ello y no sobre la máquina del usuario final.

No obstante, gracias al *script bash* que se ha creado para el despliegue y configuración de la plataforma, el usuario final puede instalarse el entorno en la máquina local para realizar las pruebas necesarias en caso de dispone de los requisitos técnicos necesarios.

Para la instalación del proyecto tan solo es necesario clonar localmente el código del repositorio *Github*: <https://github.com/liviuvj/sentiment-analysis-platform>

Posteriormente, es necesario instalar las dependencias que se encuentran en el archivo de requisitos. Para ello, es buena práctica crear primero un entorno virtual para evitar problemas de versiones posteriormente y realizar la instalación del fichero de requisitos.

```
python3 -m venv venv
source ./venv/bin/activate
pip install -r requirements.txt
```

Para la ejecución del proyecto se ha creado un *script bash* que se encarga de realizar la descarga de ficheros adicionales, clonación de repositorios, despliegue de contenedores *Docker* y configuración de los componentes que forman parte del proyecto.

Por lo que tan solo será necesario ejecutar dicho *script bash*:

```
./quickstart.sh
```

No obstante, cada directorio principal correspondiente a las herramientas empleadas para la realización de este proyecto contiene un fichero `.env`. Dicho archivo contiene los parámetros de configuración por defecto, así como usuarios y contraseñas, de los distintos componentes de la plataforma. Por lo que sería recomendable modificar dichos valores en un entorno de producción.

E.4. Manual del usuario

Las siguientes secciones que desarrollan los usos que pueden dar los usuarios finales a la plataforma se han distribuido según el propio tipo de usuario que la va a utilizar.

Usuario final básico

El usuario final básico debería tener el acceso mínimo y necesario para su caso de uso. De esta manera, se tendrá acceso a los siguientes componentes (cuyas *URLs* muestran los puertos asignados por defecto con la configuración realizada):

- **Punto web de acceso centralizado.** Página web ligera que permite el acceso directo a todas las demás interfaces web que presentan los componentes de la plataforma. Accesible en `http://localhost:5000`
- **Apache Superset.** Aplicación en la que se pueden diseñar, implementar, visualizar y compartir cuadros de mando interactivos. Accesible en `http://localhost:8088`

Visualización de cuadros de mando

Por las razones mencionadas anteriormente, este usuario podría únicamente visualizar los *dashboards* implementados en la plataforma.

Para ello, es necesario acceder a la interfaz web de la herramienta Apache Superset y acceder a alguno de los cuadros de mando disponibles según los permisos asignados.

Posteriormente podrá navegar por las distintas vistas del *dashboard* visualizando los gráficos y obtener los *insights* que crea oportunos, además de realizar acciones como el filtrado de los datos con las opciones que disponga para ello en el menú lateral.

Analista de datos

El analista de datos debería tener los conocimientos y capacidades necesarias para realizar más que solamente visualizar los *dashboards* ya implementados. Por ello, este usuario se entiende que tiene acceso a los siguientes componentes (cuyas *URLs* muestran los puertos asignados por defecto con la configuración realizada):

- **Punto web de acceso centralizado.** Página web ligera que permite el acceso directo a todas las demás interfaces web que presentan los componentes de la plataforma. Accesible en <http://localhost:5000>
- **ClickHouse.** Sencilla interfaz web que permite la ejecución de consultas directas sobre la base de datos en lenguaje *SQL*. Accesible en <http://localhost:8123/play>
- **Apache Superset.** Aplicación en la que se pueden diseñar, implementar, visualizar y compartir cuadros de mando interactivos. Accesible en <http://localhost:8088>

Exploración de datos

Este usuario debería tener acceso suficiente para realizar consultas en la base de datos utilizada para almacenar la información a visualizar, el sistema *OLAP* ClickHouse.

Para ello, se facilita su acceso directamente desde la herramienta Apache Superset, en la pestaña dedicada para ello *SQL Lab*. En esta vista se podrán seleccionar uno o varios conjuntos de datos sobre los que realizar consultas en lenguaje *SQL* nativo.

Las consultas realizadas se guardan en un histórico para poder volver a ejecutarlas posteriormente en caso necesario. Además, los resultados de estas consultas se pueden exportar como nuevos *datasets* virtuales que pueden utilizados posteriormente como base de nuevas consultas o gráficos.

The screenshot shows the Apache Superset interface. At the top, there are tabs for 'Dashboards', 'Charts', 'Datasets', and 'SQL'. Below this, the 'SQL' tab is active, showing a query editor and a results viewer.

SQL Editor:

```

1 SELECT
2   tweet_id,
3   -- LEFT(display_url, LOCATE('.', display_url) - 10),
4   -- SUBSTR(display_url, 0, LOCATE('bit', display_url)),
5   substring(display_url, 1, 3),
6
7   HID(display_url, 8, LOCATE('.','display_url', 8)),
8   display_url
9   FROM ur1s
10
11
12
13 -- WITH tmp AS (
14 --   SELECT

```

Results Viewer:

The results viewer displays three query logs:

- Query 1: SELECT ... (Status: Success, Started: 21:14:37, Duration: 00:00:00.11, Rows: 1446)
- Query 2: SELECT ... (Status: Failed, Started: 21:14:34, Duration: 22:45:20.66, Rows: 1446)
- Query 3: SELECT ... (Status: Failed, Started: 21:14:13, Duration: 00:00:00.11, Rows: 0%)

On the left side, there is a sidebar with tables and their columns:

- tweets**: tweet_id, source, search_query, user_id, created_at, language, text, retweet_count, reply_count, like_count, quote_count, sentiment, emotion, topic, entity
- users**: user_id, name, username, description, followers_count, following_count, tweet_count

Figura E.1: Vista del *SQL Lab* para la exploración de datos de Apache Superset

Creación o modificación de visualizaciones

Para la creación o modificación de nuevas visualizaciones es necesario acceder a la pestaña *Charts* de la herramienta Apache Superset. En esta vista se puede seleccionar alguno de los *datasets* disponibles y un tipo de visualización de los más de 40 tipos distintos que tiene la herramienta.

A continuación, se accede a la vista de edición de la visualización. En esta pantalla se pueden seleccionar los campos deseados que se pueden añadir a la visualización, simplemente haciendo *click* y arrastrándolos al elemento correspondiente.

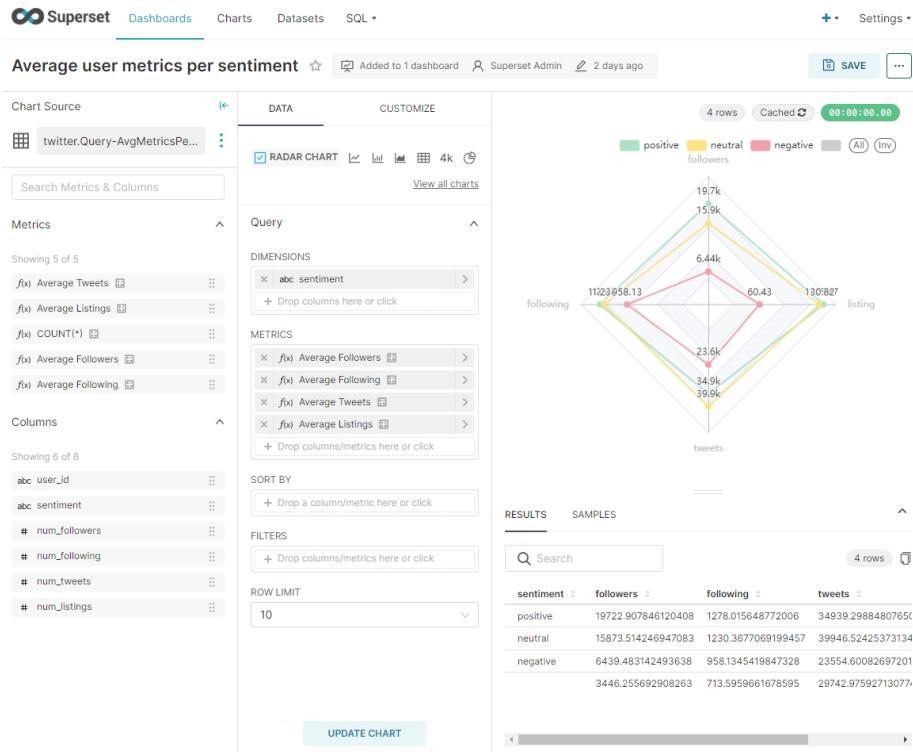


Figura E.2: Vista de edición de visualizaciones de Apache Superset

También se presentan en la misma vista opciones para cambiar el tipo de gráfico y personalizar los elementos visuales del mismo (paleta de colores, leyendas, títulos, etc.). Además, presenta también las opciones de configuración de analíticas avanzadas, como la previsión de valores.

Integración de nuevos cuadros de mando

Para la creación de nuevos cuadros de mando se ha de acceder a la pestaña *Dashboards* de la herramienta Apache Superset. En esta vista se pueden modificar los *dashboards* ya existentes o añadir nuevos.

Esta pantalla de edición cuenta con las siguientes partes principales:

- El menú lateral izquierdo permite la definición y configuración de filtros para el cuadro de mando. Se permite la selección de atributos presentes entre los distintos *datasets* que pueden estar formando el *dashboard* y la elección de a qué gráficos afectan estos filtros.

- La vista central es la plantilla donde se van a colocar y organizar las distintas visualizaciones creadas anteriormente. Se puede ajustar tanto el tamaño como la posición que ocupan en la pantalla.
- El menú lateral derecho contiene las visualizaciones creadas hasta el momento y acceso directo a la creación de nuevas, estas se pueden colocar en la vista central simplemente arrastrándolas. También cuenta con unos elementos gráficos para la estructuración del *dashboard* como separadores, pestañas, organizadores, etc.

Bibliografía

- [1] AICromo. The current state of the art in natural language processing (nlp). [En línea]. AICromo Limited. Disponible en <https://aicromo.com/ai-blogs/the-current-state-of-the-art-in-natural-language-processing-nlp>, 2023. [Fecha de consulta: 28-06-2023].
- [2] Airbyte. Airbyte configuration api. [En línea]. Airbyte, 2023. Disponible en <https://airbyte-public-api-docs.s3.us-east-2.amazonaws.com/rapidoc-api-docs.html>, 2023. [Fecha de consulta: 08-03-2023].
- [3] Airbyte. Architecture overview. [En línea]. Airbyte, Inc., 2023. Disponible en <https://docs.airbyte.com/understanding-airbyte/high-level-view>, 2023. [Fecha de consulta: 23-03-2023].
- [4] Airbyte. Hundreds of connectors out-of-the-box. [En línea]. Airbyte, Inc., 2023. Disponible en <https://airbyte.com/connectors>, 2023. [Fecha de consulta: 28-02-2023].
- [5] Airbyte. Manage airbyte open source. [En línea]. Airbyte, Inc., 2023. Disponible en <https://docs.airbyte.com/category/manage-airbyte-open-source>, 2023. [Fecha de consulta: 05-03-2023].
- [6] Dimosthenis Antypas, Asahi Ushio, Jose Camacho-Collados, Leonardo Neves, Vítor Silva, and Francesco Barbieri. Twitter topic classification. *arXiv preprint arXiv:2209.09824*, 2022. [Fecha de consulta: 07-07-2023].
- [7] AWS. What is a data lake? [En línea]. 2023, Amazon Web Services, Inc. Disponible en <https://aws.amazon.com/big-data/datalakes->

- [and-analytics/what-is-a-data-lake/](https://www.dataversity.net/what-is-a-data-lake/), 2023. [Fecha de consulta: 12-06-2023].
- [8] Karissa Bell. Twitter announces new api pricing, including a limited free tier for bots. [En línea]. engadget. Disponible en <https://www.engadget.com/twitter-announces-new-api-pricing-including-a-limited-free-tier-for-bots-005251253.html>, 2023. [Fecha de consulta: 26-06-2023].
 - [9] Bryce Boe. Praw: The python reddit api wrapper. [En línea]. reddit, inc., 2023. Disponible en <https://github.com/praw-dev/praw>, 2023. [Fecha de consulta: 28-02-2023].
 - [10] Mahdi Bohlouli, Jens Dalter, Mareike Dornhöfer, Johannes Zenkert, and Madjid Fathi. Knowledge discovery from social media using big data provided sentiment analysis (somabit). [En línea]. IBM. Disponible en <https://doi.org/10.48550/arXiv.2001.05996>, 2020. [Fecha de consulta: 20-06-2023].
 - [11] Jose Camacho-Collados, Kiamehr Rezaee, Talayeh Riahi, Asahi Ushio, Daniel Loureiro, Dimosthenis Antypas, Joanne Boisson, Luis Espinosa-Anke, Fangyu Liu, Eugenio Martínez-Cámarra, et al. Tweetnlp: Cutting-edge natural language processing for social media. *arXiv preprint arXiv:2206.14774*, 2022. [Fecha de consulta: 20-06-2023].
 - [12] cardiffnlp. tweet-topic-21-multi. [En línea]. Hugging Face. Disponible en <https://huggingface.co/cardiffnlp/tweet-topic-21-multi>, 2023. [Fecha de consulta: 07-07-2023].
 - [13] cardiffnlp. Twitter-roberta-base for emotion recognition. [En línea]. Hugging Face. Disponible en <https://huggingface.co/cardiffnlp/twitter-roberta-base-emotion>, 2023. [Fecha de consulta: 07-07-2023].
 - [14] cardiffnlp. Twitter-roberta-base for sentiment analysis. [En línea]. Hugging Face. Disponible en <https://huggingface.co/cardiffnlp/twitter-roberta-base-sentiment-latest>, 2023. [Fecha de consulta: 07-07-2023].
 - [15] Guan-Lin Chao and Ian Lane. Bert-dst: Scalable end-to-end dialogue state tracking with bidirectional encoder representations from transformer. *arXiv preprint arXiv:1907.03040*, 2019. [Fecha de consulta: 29-06-2023].

- [16] Qian Chen, Zhu Zhuo, and Wen Wang. Bert for joint intent classification and slot filling. *arXiv preprint arXiv:1902.10909*, 2019. [Fecha de consulta: 29-06-2023].
- [17] K. R. Chowdhary. Natural language processing. [En línea]. Springer Nature. Disponible en https://doi.org/10.1007/978-81-322-3972-7_19, 2020. [Fecha de consulta: 07-02-2023].
- [18] ClickHouse. Distinctive features of clickhouse. [En línea]. 2023 ClickHouse, Inc. Disponible en <https://clickhouse.com/docs/en/about-us/distinctive-features/>, 2023. [Fecha de consulta: 10-04-2023].
- [19] Databricks. Extract transform load (etl). [En línea]. Databricks 2023. Disponible en [https://www.databricks.com/glossary/extract-tranform-load](https://www.databricks.com/glossary/extract-transform-load), 2023. [Fecha de consulta: 20-06-2023].
- [20] Francisco de Abreu e Lima. Game of thrones s8 (twitter). [En línea]. Kaggle. Disponible en <https://www.kaggle.com/datasets/monogene/a/game-of-thrones-twitter>, 2019. [Fecha de consulta: 29-06-2023].
- [21] Google Developers. Quota usage. [En línea]. YouTube Data API, 2023. Disponible en <https://developers.google.com/youtube/v3/getting-started#calculating-quota-usage>, 2023. [Fecha de consulta: 26-02-2023].
- [22] Google Developers. Resource types. [En línea]. YouTube Data API, 2023. Disponible en <https://developers.google.com/youtube/v3/docs#resource-types>, 2023. [Fecha de consulta: 26-02-2023].
- [23] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. [En línea]. arXiv. Disponible en <https://arxiv.org/abs/1810.04805v2>, 2019. [Fecha de consulta: 04-02-2023].
- [24] Domo. Data never sleeps 1.0 vs 10.0. [En línea]. 2023 Domo, Inc. Disponible en <https://www.domo.com/data-never-sleeps>, 2019. [Fecha de consulta: 06-07-2023].
- [25] Domo. Data never sleeps infographic. [En línea]. 2023 Domo, Inc. Disponible en <https://web-assets.domo.com/blog/wp-content/uploads/2021/09/data-never-sleeps-9.0-1200px-1.png>, 2019. [Fecha de consulta: 06-07-2023].

- [26] Hugging Face. The transformer model family. [En línea]. Hugging Face. Disponible en https://huggingface.co/docs/transformers/model_summary, 2023. [Fecha de consulta: 18-04-2023].
- [27] Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. *arXiv preprint arXiv:1708.00524*, 2017. [Fecha de consulta: 29-06-2023].
- [28] Apache Software Foundation. Introduction to apache druid. [En línea]. 2022 Apache Software Foundation. Disponible en <https://druid.apache.org/docs/latest/design/>, 2022. [Fecha de consulta: 10-04-2023].
- [29] Apache Software Foundation. Hdfs architecture. [En línea]. 2008-2023 Apache Software Foundation. Disponible en <https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html>, 2023. [Fecha de consulta: 25-03-2023].
- [30] Apache Software Foundation. What is apache hudi. [En línea]. 2021 The Apache Software Foundation. Disponible en <https://hudi.apache.org/docs/overview>, 2023. [Fecha de consulta: 25-03-2023].
- [31] The Apache Software Foundation. Cassandra overview. [En línea]. 2009-2023 The Apache Software Foundation. Disponible en <https://cassandra.apache.org/doc/latest/cassandra/architecture/overview.html>, 2023. [Fecha de consulta: 26-03-2023].
- [32] The Apache Software Foundation. Fernet. [En línea]. The Apache Software Foundation 2023. Disponible en <https://airflow.apache.org/docs/apache-airflow/stable/security/secrets/fernet.html>, 2023. [Fecha de consulta: 19-05-2023].
- [33] The Apache Software Foundation. Package apache-airflow-providers-airbyte. [En línea]. The Apache Software Foundation 2023. Disponible en <https://airflow.apache.org/docs/apache-airflow-providers-airbyte>, 2023. [Fecha de consulta: 23-06-2023].
- [34] The Apache Software Foundation. Package apache-airflow-providers-docker. [En línea]. The Apache Software Foundation 2023. Disponible en <https://airflow.apache.org/docs/apache-airflow-providers-docker>, 2023. [Fecha de consulta: 23-06-2023].

- [35] The Apache Software Foundation. Spark overview. [En línea]. 2018 The Apache Software Foundation. Disponible en <https://spark.apache.org/docs/latest/index.html>, 2023. [Fecha de consulta: 18-04-2023].
- [36] The Apache Software Foundation. What is airflow? [En línea]. The Apache Software Foundation 2023. Disponible en <https://airflow.apache.org/docs/apache-airflow/stable/index.html>, 2023. [Fecha de consulta: 19-05-2023].
- [37] The Apache Software Foundation. What is apache superset? [En línea]. 2023, The Apache Software Foundation. Disponible en <https://superset.apache.org/docs/intro>, 2023. [Fecha de consulta: 15-04-2023].
- [38] IBM. Etl (extract, transform, load). [En línea]. IBM. Disponible en <https://www.ibm.com/topics/etl>, 2023. [Fecha de consulta: 20-06-2023].
- [39] IkarosKun. Python-facebook. [En línea]. GitHub, 2023, 2023. Disponible en <https://github.com/sns-sdks/python-facebook>, 2022. [Fecha de consulta: 28-02-2023].
- [40] Reddit Inc. Api overview. [En línea]. reddit, inc., 2023. Disponible en <https://www.reddit.com/dev/api>, 2023. [Fecha de consulta: 26-02-2023].
- [41] Antonio Pequeño IV. Reddit stands by controversial api changes as subreddit protest continues. [En línea]. WIRED. Disponible en <https://www.forbes.com/sites/antoniopequenoiv/2023/06/13/reddit-stands-by-controversial-api-changes-as-subreddit-protest-continues/>, 2023. [Fecha de consulta: 26-06-2023].
- [42] Diksha Khurana, Aditya Koli, Kiran Khatter, and Sukhdev Singh. Natural language processing: State of the art, current trends and challenges. *Multimedia tools and applications*, 82(3):3713–3744, 2023. [Fecha de consulta: 28-06-2023].
- [43] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014. [Fecha de consulta: 29-06-2023].
- [44] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*, 2016. [Fecha de consulta: 29-06-2023].

- [45] Daniel Loureiro, Francesco Barbieri, Leonardo Neves, Luis Espinosa Anke, and Jose Camacho-Collados. Timelms: Diachronic language models from twitter. *arXiv preprint arXiv:2202.03829*, 2022. [Fecha de consulta: 07-07-2023].
- [46] Meta. General information - graph api. [En línea]. Meta for Developers, 2023. Disponible en <https://developers.facebook.com/docs/graph-api/overview>, 2023. [Fecha de consulta: 26-02-2023].
- [47] Meta. Graph api reference. [En línea]. Meta for Developers, 2023. Disponible en <https://developers.facebook.com/docs/graph-api/reference>, 2023. [Fecha de consulta: 26-02-2023].
- [48] Meta. Instagram graph api reference. [En línea]. Meta for Developers, 2023. Disponible en <https://developers.facebook.com/docs/instagram-api/reference>, 2023. [Fecha de consulta: 26-02-2023].
- [49] Meta. Permissions reference. [En línea]. Meta for Developers, 2023. Disponible en <https://developers.facebook.com/docs/permissions/reference>, 2023. [Fecha de consulta: 26-02-2023].
- [50] Meta. Rate limits - graph api. [En línea]. Meta for Developers, 2023. Disponible en <https://developers.facebook.com/docs/graph-api/overview/rate-limiting>, 2023. [Fecha de consulta: 26-02-2023].
- [51] Kostadin Mishev, Ana Gjorgjevikj, Irena Vodenska, Lubomir Chitkushev, and Dimitar Trajanov. Evaluation of sentiment analysis in finance: From lexicons to transformers. [En línea]. IEEE Xplore, 2022. Disponible en <https://doi.org/10.1109/ACCESS.2020.3009626>, 2020. [Fecha de consulta: 04-02-2023].
- [52] MongoDB. Compass. the gui for mongodb. [En línea]. 2023 Inc. MongoDB. Disponible en <https://www.mongodb.com/products/compass>, 2023. [Fecha de consulta: 26-03-2023].
- [53] MongoDB. Mongodb architecture guide. [En línea]. 2021 Inc. MongoDB. Disponible en <https://www.mongodb.com/collateral/mongodb-architecture-guide>, 2023. [Fecha de consulta: 26-03-2023].
- [54] Oracle. What is big data? [En línea]. 2023 Oracle. Disponible en <https://www.oracle.com/big-data/what-is-big-data/>, 2023. [Fecha de consulta: 15-06-2023].

- [55] SAS. Data warehouse. [En línea]. 2023 SAS Institute Inc. Disponible en https://www.sas.com/es_es/insights/data-management/data-warehouse.html, 2023. [Fecha de consulta: 12-06-2023].
- [56] Abigail See, Peter J Liu, and Christopher D Manning. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*, 2017. [Fecha de consulta: 29-06-2023].
- [57] Tom Shafer. The 42 v's of big data and data science. [En línea]. 2023 Elder Research, Inc. Disponible en <https://www.elderresearch.com/blog/the-42-vs-of-big-data-and-data-science/>, 2017. [Fecha de consulta: 15-06-2023].
- [58] Tazmina Sharmin, Fabio Di Troia, Katerina Potika, and Mark Stamp. Convolutional neural networks for image spam detection. *Information Security Journal: A Global Perspective*, 29(3):103–117, 2020. [Fecha de consulta: 29-06-2023].
- [59] Chris Stokel-Walker. Twitter's \$42 000-per-month api prices out nearly everyone. [En línea]. WIRED. Disponible en <https://www.wired.com/story/twitter-data-api-prices-out-nearly-everyone/>, 2023. [Fecha de consulta: 26-06-2023].
- [60] tecnativa. Docker socket proxy. [En línea]. 2023 Docker, Inc. Disponible en <https://hub.docker.com/r/tecnativa/docker-socket-proxy>, 2023. [Fecha de consulta: 24-06-2023].
- [61] tner. tner/twitter-roberta-base-dec2021-tweetner7-all. [En línea]. Hugging Face. Disponible en <https://huggingface.co/cardiffnlp/twitter-roberta-base-emotion>, 2023. [Fecha de consulta: 07-07-2023].
- [62] Twitter. Twitter's public workspace. [En línea]. Postman, Inc., 2023. Disponible en <https://www.postman.com/twitter/workspace/twitter-s-public-workspace/collection/9956214-784efcda-ed4c-4491-a4c0-a26470a67400?ctx=documentation>, 2023. [Fecha de consulta: 28-02-2023].
- [63] Inc. Twitter. Getting started with the twitter api. [En línea]. Twitter, 2023. Disponible en <https://developer.twitter.com/en/docs/twitter-api/getting-started/about-twitter-api>, 2023. [Fecha de consulta: 26-02-2023].
- [64] Inc. Twitter. Python client for the twitter api v2 search endpoints. [En línea]. GitHub, 2023. Disponible en <https://github.com/twitt>

- erdev/search-tweets-python/tree/v2, 2023. [Fecha de consulta: 28-02-2023].
- [65] Inc. Twitter. Rate limits. [En línea]. Twitter, 2023. Disponible en <https://developer.twitter.com/en/docs/twitter-api/rate-limits#v2-limits>, 2023. [Fecha de consulta: 26-02-2023].
- [66] Inc. Twitter. Twitter api platform resources. [En línea]. Twitter, 2023. Disponible en <https://developer.twitter.com/en/docs/twitter-api/getting-started/about-twitter-api#item2>, 2023. [Fecha de consulta: 26-02-2023].
- [67] Asahi Ushio and Jose Camacho-Collados. T-ner: an all-round python library for transformer-based named entity recognition. *arXiv preprint arXiv:2209.12616*, 2022. [Fecha de consulta: 07-07-2023].
- [68] Asahi Ushio, Leonardo Neves, Vitor Silva, Francesco Barbieri, and Jose Camacho-Collados. Named entity recognition in twitter: A dataset and analysis on short-term temporal shifts. *arXiv preprint arXiv:2210.03797*, 2022. [Fecha de consulta: 07-07-2023].
- [69] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. [Fecha de consulta: 28-06-2023].
- [70] Josh Wardle. Wiki api. [En línea]. GitHub, 2015. Disponible en <https://github.com/reddit-archive/reddit/wiki/API>, 2023. [Fecha de consulta: 26-02-2023].
- [71] Angela Watercutter. Reddit won't be the same. neither will the internet. [En línea]. WIRED. Disponible en <https://www.wired.com/story/reddit-api-changes-ai-labor/>, 2023. [Fecha de consulta: 26-06-2023].
- [72] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016. [Fecha de consulta: 29-06-2023].