

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 — 9 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 sobre palabras clave o temas específicos”.

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, 9 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

Aprendizaje automático, 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

Machine 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 |
| 3. Conceptos teóricos | 11 |
| 3.1. Big Data | 11 |
| 3.2. Proceso ETL | 15 |
| 3.3. Big Data Knowledge Discovery | 18 |
| 3.4. Procesamiento de lenguaje natural | 20 |
| 4. Técnicas y herramientas | 27 |
| 4.1. Técnicas | 27 |
| 4.2. Herramientas | 28 |
| 5. Aspectos relevantes del desarrollo del proyecto | 39 |
| 5.1. Extracción de datos | 39 |

| | |
|--|-----------|
| 5.2. Carga de datos | 49 |
| 5.3. Transformación de los datos | 49 |
| 5.4. Visualización de los datos | 50 |
| 5.5. Orquestación de los procesos | 50 |
| 5.6. Interfaz web de acceso centralizado | 51 |
| 6. Trabajos relacionados | 53 |
| 6.1. Herramientas similares | 53 |
| 6.2. Artículos científicos | 55 |
| 7. Conclusiones y Líneas de trabajo futuras | 57 |
| Apéndices | 58 |
| Apéndice A Plan de Proyecto Software | 61 |
| A.1. Introducción | 61 |
| A.2. Planificación temporal | 61 |
| A.3. Estudio de viabilidad | 73 |
| Apéndice B Especificación de Requisitos | 75 |
| B.1. Introducción | 75 |
| B.2. Objetivos generales | 75 |
| B.3. Catalogo de requisitos | 75 |
| B.4. Especificación de requisitos | 75 |
| Apéndice C Especificación de diseño | 77 |
| C.1. Introducción | 77 |
| C.2. Diseño de datos | 77 |
| C.3. Diseño procedimental | 80 |
| C.4. Diseño arquitectónico | 80 |
| Apéndice D Documentación técnica de programación | 81 |
| D.1. Introducción | 81 |
| D.2. Estructura de directorios | 81 |
| D.3. Manual del programador | 81 |
| D.4. Compilación, instalación y ejecución del proyecto | 87 |
| D.5. Pruebas del sistema | 87 |
| Apéndice E Documentación de usuario | 89 |
| E.1. Introducción | 89 |
| E.2. Requisitos de usuarios | 89 |

Índice general

v

| | |
|-----------------------------------|-----------|
| E.3. Instalación | 89 |
| E.4. Manual del usuario | 89 |
| Bibliografía | 91 |

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 | 12 |
| 3.3. | Cantidad de datos generados cada minuto durante el año 2021 en Internet | 12 |
| 3.4. | Ilustración básica de un proceso <i>ETL</i> | 16 |
| 3.5. | Arquitectura básica del sistema <i>SoMABiT</i> | 19 |
| 3.6. | Evolución de las técnicas <i>NLP</i> | 21 |
| 3.7. | Esquema básico de neuronas recurrentes | 22 |
| 3.8. | Esquema básico de neuronas recurrentes con funciones de olvido incluidas | 22 |
| 3.9. | Arquitectura del modelo <i>Transformer</i> | 23 |
| 3.10. | Mecanismo de atención del modelo <i>Transformer</i> | 24 |
| 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 | 25 |
| 5.12. | Visión general de la arquitectura de <i>Airbyte</i> | 44 |
| 5.13. | Configuración básica del conector original de <i>Airbyte</i> para Twitter | 45 |
| 5.14. | Configuración ampliada del conector mejorado de <i>Airbyte</i> para Twitter | 46 |
| 5.15. | Aplicación web desarrollada como punto de acceso centralizado . | 52 |
| C.1. | Diseño inicial de la pestaña <i>Raw data</i> | 78 |
| C.2. | Primera iteración de la pestaña <i>Raw data</i> | 78 |
| C.3. | Diseño inicial de la pestaña <i>Sentiment overview</i> | 79 |
| C.4. | Primera iteración de la pestaña <i>Sentiment overview</i> | 79 |
| C.5. | Diseño inicial de la pestaña <i>Sentiment analysis</i> | 80 |
| C.6. | Primera iteración de la pestaña <i>Sentiment analysis</i> | 80 |

| | |
|---|----|
| D.1. Configuración del origen de datos en Airbyte: Twitter | 83 |
| D.2. Configuración del destino de datos en Airbyte: CSV Local | 85 |
| D.3. Configuración de la conexión entre origen y destino de los datos en Airbyte | 87 |

Índice de tablas

| | | |
|------|---|----|
| 5.1. | Recursos disponibles a través de la <i>API</i> de Twitter | 40 |
| 5.2. | Muestra de nodos disponibles en <i>Graph API</i> de Facebook | 41 |
| 5.3. | Muestra de nodos disponibles en <i>Graph API</i> de Instagram | 42 |
| 5.4. | Recursos disponibles a través de la <i>API</i> de YouTube | 42 |
| 5.5. | Recursos disponibles a través de la <i>API</i> de Reddit | 43 |

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.
 - ...
- Los datos extraídos de *API* mediante la herramienta Airbyte.
 - ...
- El análisis realizado en *Google Colaboratory*.
https://colab.research.google.com/drive/1d_obU9idFqjsDi7ezeFs1CORxTUagh7V#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*.
 - ...
 - *dataset_season8*.
https://drive.google.com/file/d/1tSA5bGkBGfgVWdltxLEEfd_NNm0qnUYs/view
 - *dataset_daenerys*.
<https://drive.google.com/file/d/1hL3eh3K2lKtMNEkaG2JSgNSXjoNtHyTG/view>
 - *dataset_jon*.
<https://drive.google.com/file/d/1Uji4IajDAYlAj3yhQdQ9B1NcSFg-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.

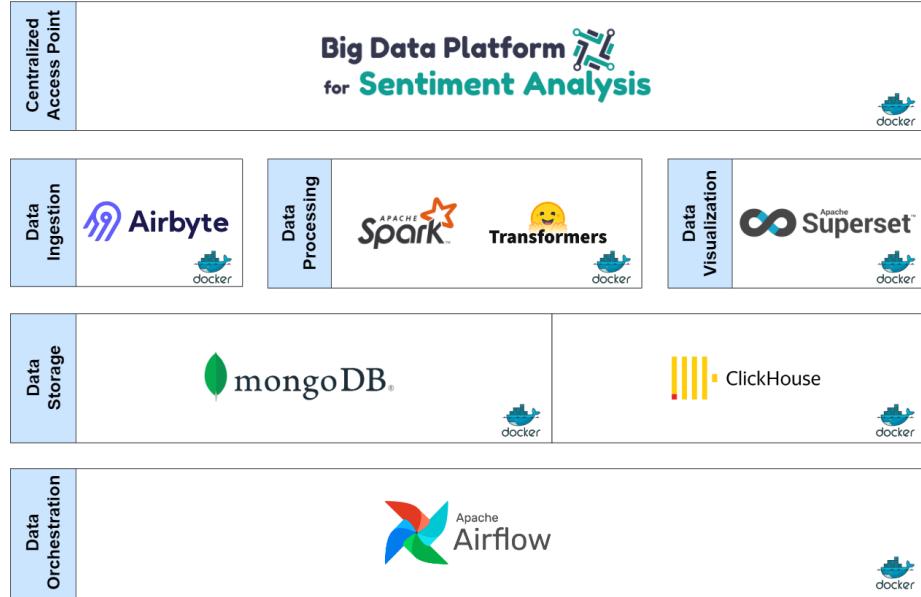


Figura 1.1: Vista básica de la arquitectura del proyecto

El primero es *Airbyte*, una herramienta que permite la extracción de datos desde distintos orígenes, como *APIs*, bases de datos o archivos disponibles en la *web*. Se ha utilizado para la ingestión de datos desde *API* y desde *Google Drive*.

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* 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».

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* 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.

Objetivos del proyecto

Este apartado explica de forma precisa y concisa cuales son los objetivos que se persiguen con la realización del proyecto. Se puede distinguir entre los objetivos marcados por los requisitos del software a construir y los objetivos de carácter técnico que plantea a la hora de llevar a la práctica el proyecto.

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 la [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.

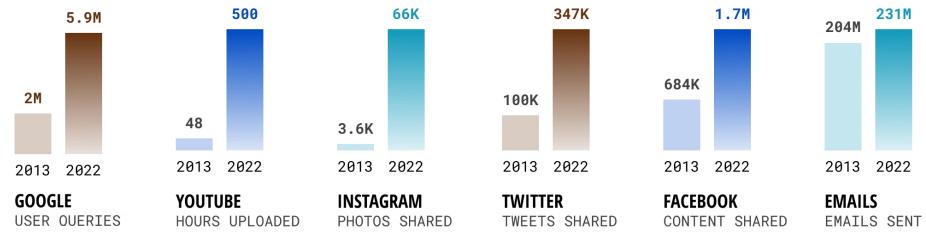


Figura 3.2: Diferencia entre la cantidad de datos generados por minuto entre 2013 y 2022 de algunas plataformas. Fuente: [17]

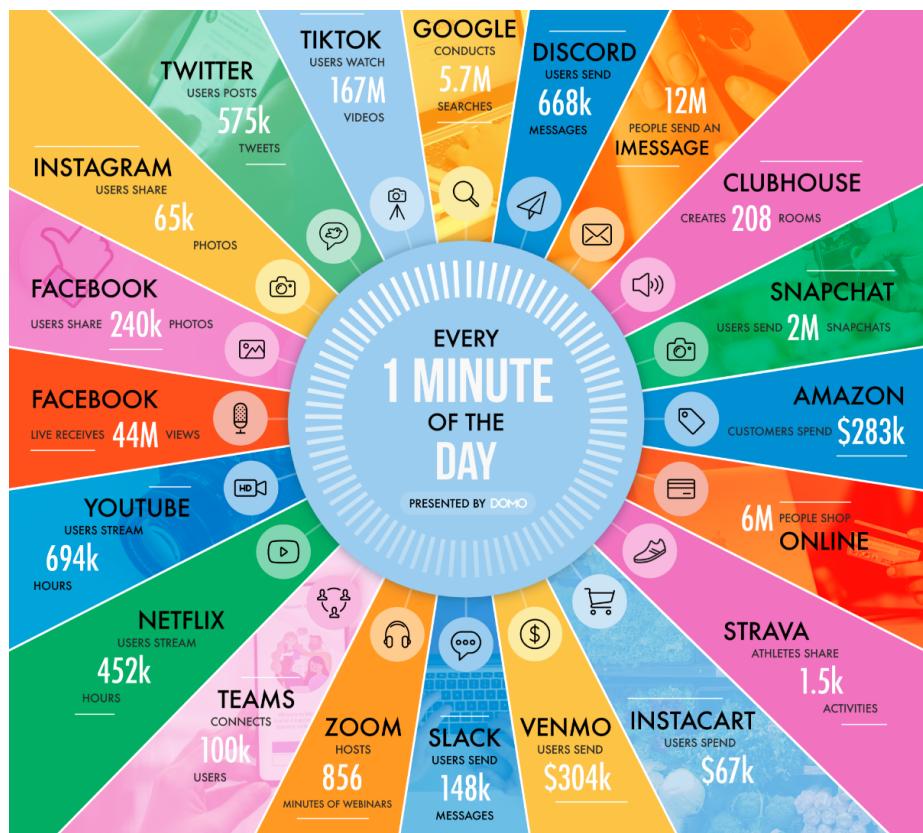


Figura 3.3: Cantidad de datos generados cada minuto durante el año 2021 en Internet. Fuente: [18]

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* [37], 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

A parte 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 [39]. 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.

Primero es necesario disponer de la capacidad de procesar grandes volúmenes de datos en tiempo real. Como se dijo 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, con sus muy seguras 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 [12].

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:

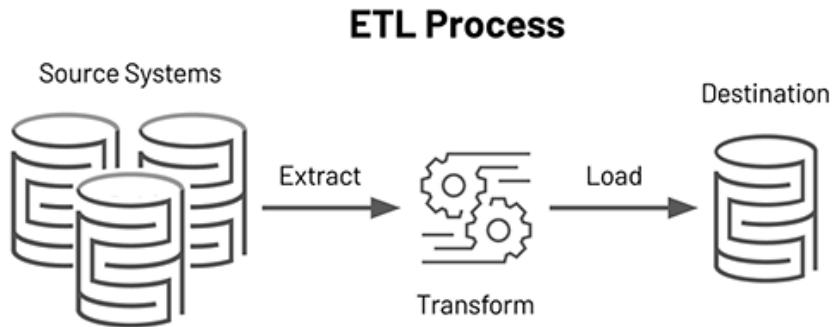


Figura 3.4: Ilustración básica de un proceso *ETL*. Fuente: [12]

- **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*) [23], 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 traspasables 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.

Resulta importante destacar de igual manera 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* [8] (*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.

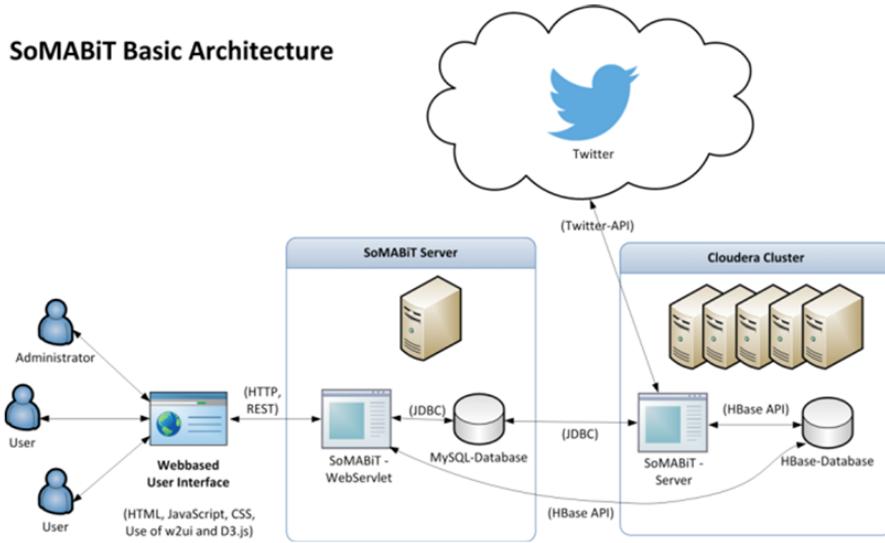


Figura 3.5: Arquitectura básica del sistema *SoMABiT*. Fuente: [8]

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*).

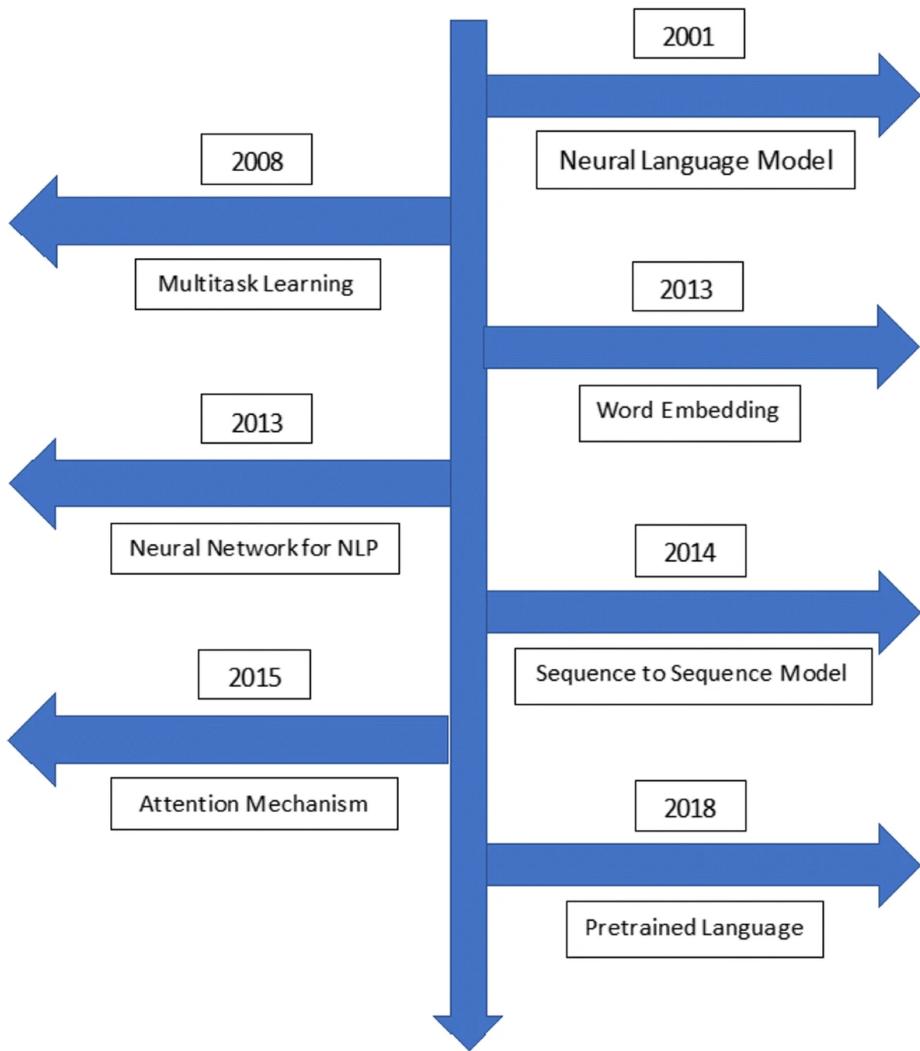


Figura 3.6: Evolución de las técnicas NLP. Fuente: [27]

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.

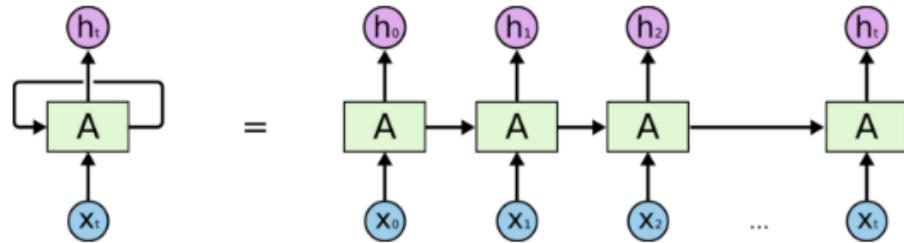


Figura 3.7: Esquema básico de neuronas recurrentes. Fuente: [1]

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

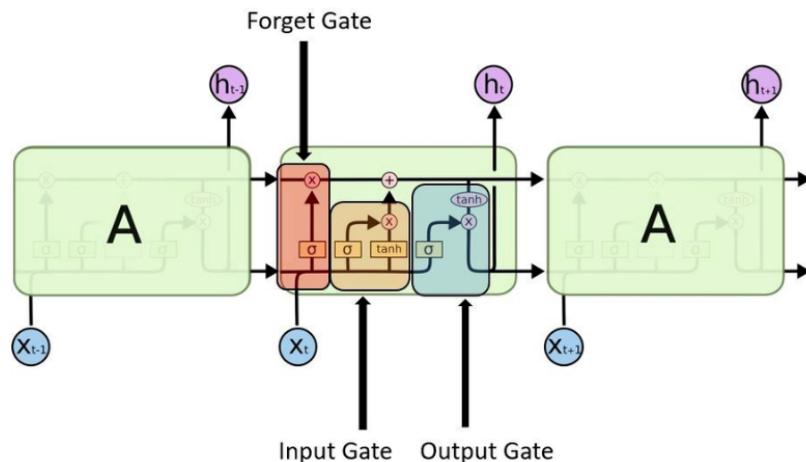


Figura 3.8: Esquema básico de neuronas recurrentes con funciones de olvido incluidas. Fuente: [1]

Estas redes neuronales son una evolución de las *RNN*. A parte 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* [47], 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*.

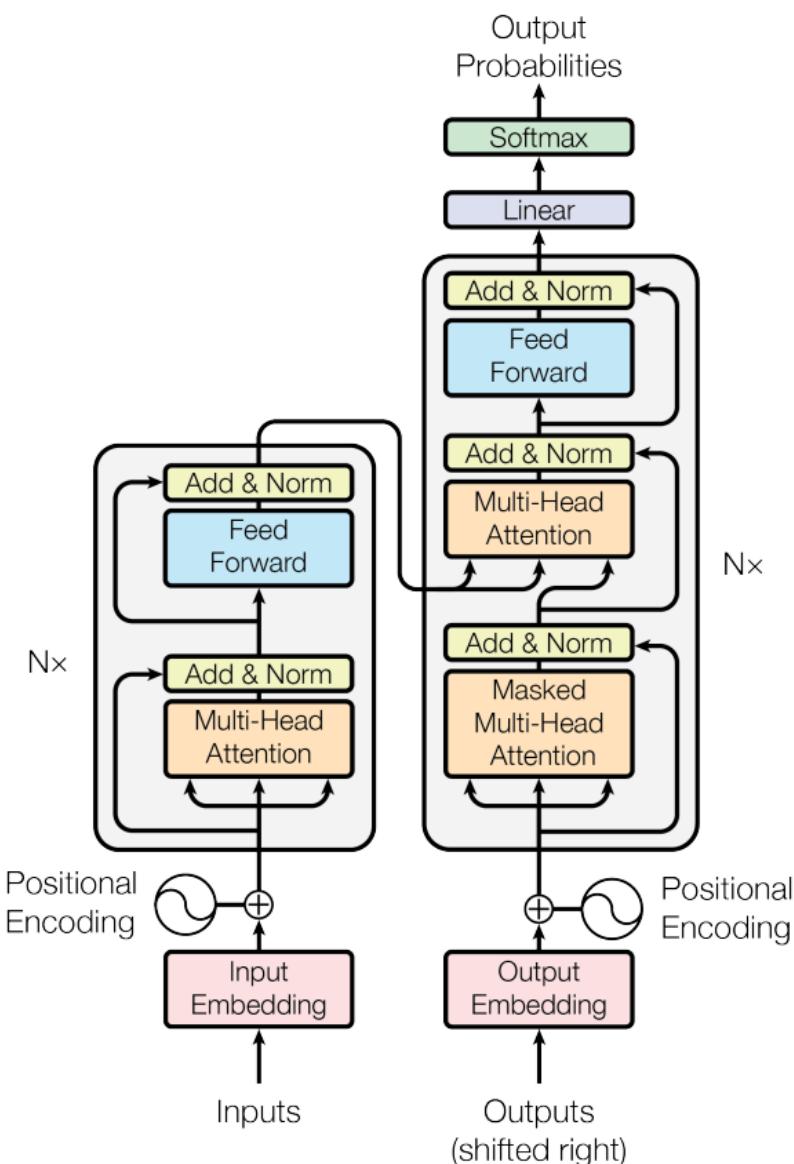


Figura 3.9: Arquitectura del modelo *Transformer*. Fuente: [47]

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 .

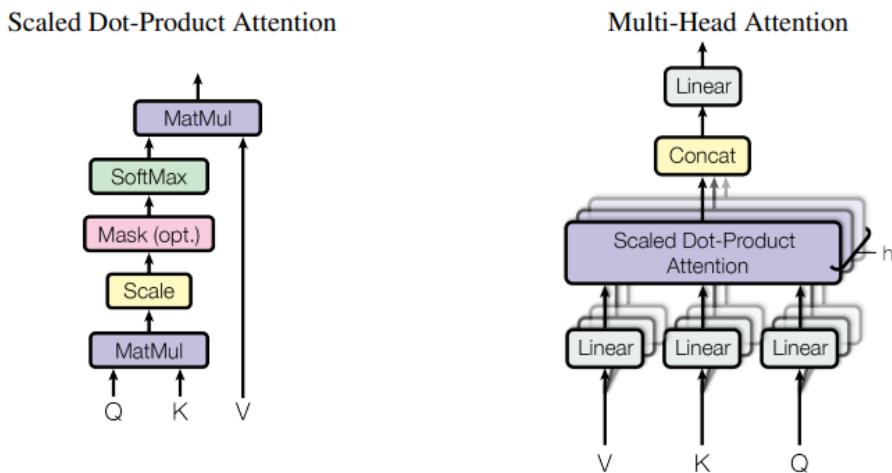


Figura 3.10: Mecanismo de atención del modelo *Transformer*. Fuente: [47]

■ **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.

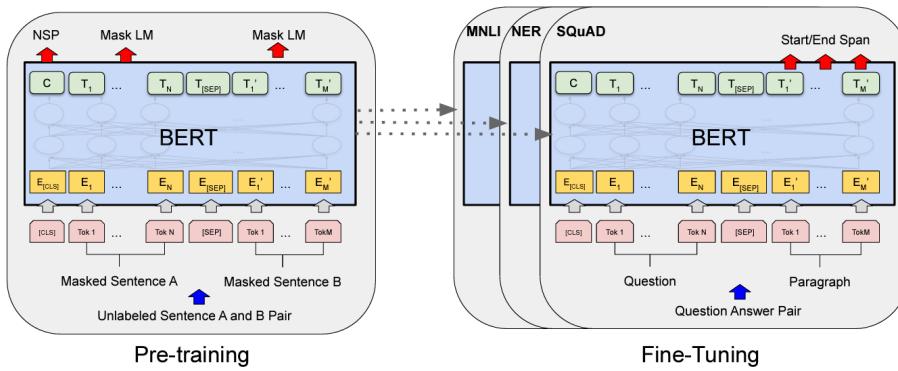


Figura 3.11: Entrenamiento y *fine-tuning* del modelo *BERT*. El pre-entrenamiento del *LLM* permite su posterior adaptación mediante transferencia de aprendizaje a nuevas tareas o conjuntos de datos. Fuente: [16]

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 [19] [28].
- **Traducción automática.** Trata la conversión de un texto en un idioma origen a un idioma destino [50].
- **Filtros de *spam*.** Identificación de mensajes y correos electrónicos no deseados [40].
- **Extracción de información.** Selección automática de información específica y extracción de fragmentos relevantes [29].
- **Elaboración de resúmenes.** Generación de la síntesis de un texto de manera concisa y coherente [38].
- **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 [9].
- **Medicina.** Análisis de registros clínicos, información sanitaria y datos médicos [10].

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 [11].

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 [35], 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*¹, 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*². 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

¹<https://github.com/>

²<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**³

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.

³<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 públicamente⁴. 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 línea⁵ de L^AT_EX. Para utilizarlo no es necesario realizar la instalación de ningún componente, tiene documentación integrada

⁴https://github.com/bbaruque/plantillaTFM_MUINBDES.git

⁵<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*⁶. 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*⁷. 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

⁶<https://joplinapp.org/>

⁷<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*⁸. Se trata de una plataforma de desarrollo, distribución y despliegue que permite separar las aplicaciones de la infraestructura en la que se ejecutan.

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 plataforma⁹ 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 [42], 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 en algunos lenguajes de programación que «envuelven» y «atacan» los *endpoints* de dichas *APIs*. Estos paquetes tienen el objetivo de facilitar al

⁸<https://docs.docker.com/>

⁹<https://www.postman.com/>

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* [44].
- **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 [24].
- **PRAW: The Python Reddit API Wrapper.** Paquete de *Python* que facilita el acceso a la *API* de Reddit [7].

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 plataforma¹⁰ 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.

¹⁰<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 [20].
- **Apache Haudi.** 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 [21].
- **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 [22].
- **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 [36].

Herramienta elegida:

- **MongoDB**¹¹

¹¹<https://www.mongodb.com/>

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.

Apache Spark

Apache Spark¹² es un sistema de procesamiento de datos distribuido y de código abierto. 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.

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*¹³ basado en arquitectura *share-nothing*, compresión de datos y posibilidad de consultas en lenguaje SQL nativo. 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*¹⁴ distribuida, escalable y auto-balanceada. Al contrario que ClickHouse, presenta una arquitectura

¹²<https://spark.apache.org/>

¹³<https://clickhouse.com/>

¹⁴<https://druid.apache.org/>

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.

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 datos¹⁵ 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 moderna¹⁶ 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.

¹⁵<https://www.metabase.com/>

¹⁶<https://superset.apache.org/>

Herramienta elegida:

- **Apache Superset**

La herramienta *Apache Superset* 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 orquestación¹⁷ 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 Python¹⁸ 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*.

Herramienta elegida:

- **Apache Airflow**

¹⁷<https://airflow.apache.org/>

¹⁸<https://github.com/spotify/luigi>

La herramienta *Apache Airflow* 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* [43] 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*¹⁹ 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 [45].

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 v1.1 <i>Premium</i> <i>Enterprise</i> | Operaciones <i>CRUD</i> . |
| Users | v2 v1.1 <i>Premium</i> <i>Enterprise</i> | Gestión y búsqueda de usuarios y relaciones entre los mismos. |
| Spaces | v2 | Búsqueda de espacios y participantes. |
| Direct Messages | v1.1 | Envío y respuesta a mensajes directos. |
| Lists | v2 v1.1 | Gestión de listas de contactos. |
| 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: [46]

¹⁹Lenguaje 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* [30], 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 [33] 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 [34].

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: [31]

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: [\[32\]](#)

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 [\[14\]](#).

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: [\[15\]](#)

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 [48] 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: [25]

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.

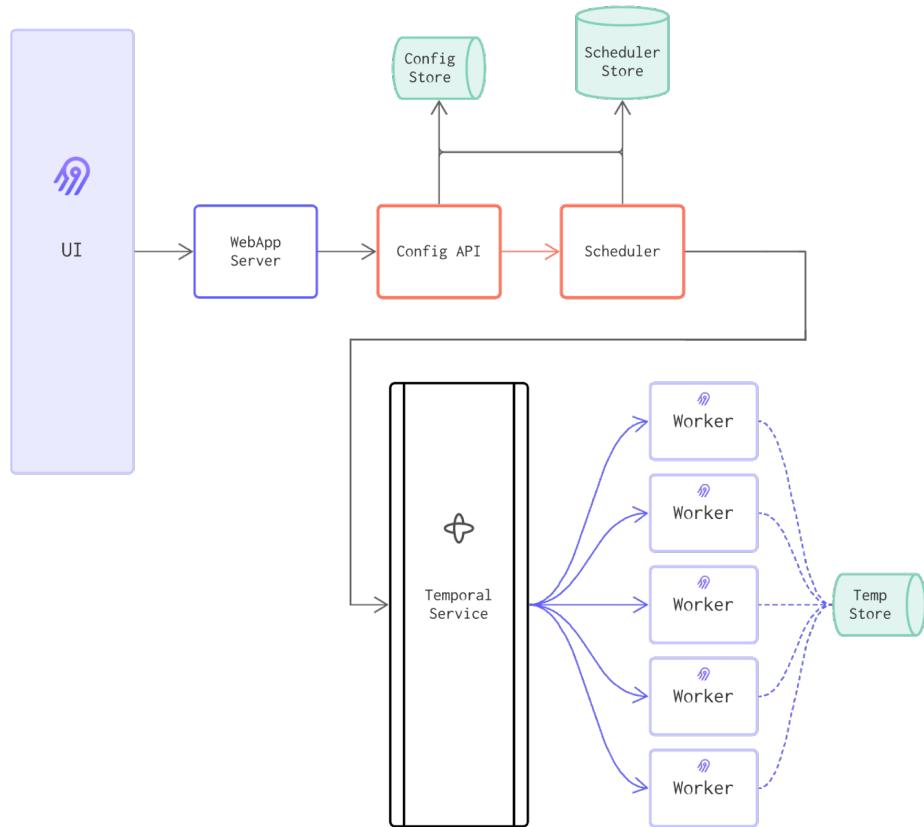


Figura 5.12: Visión general de la arquitectura de *Airbyte*. Fuente: [3]

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/liviuvj/airbyte-source-twitter/tags>.

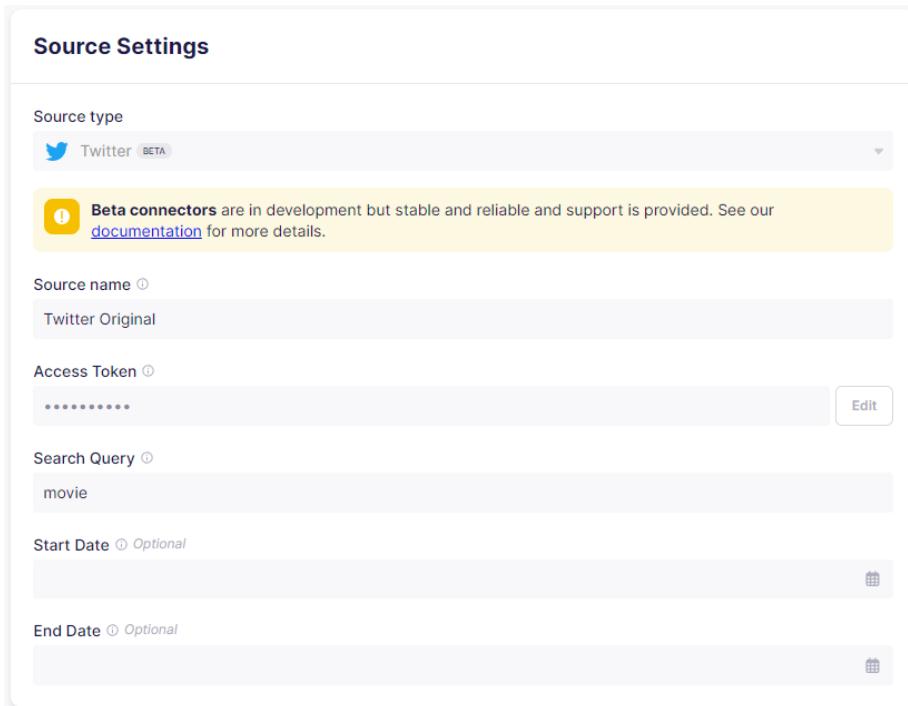


Figura 5.13: Configuración básica del conector original de Airbyte para Twitter

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>.

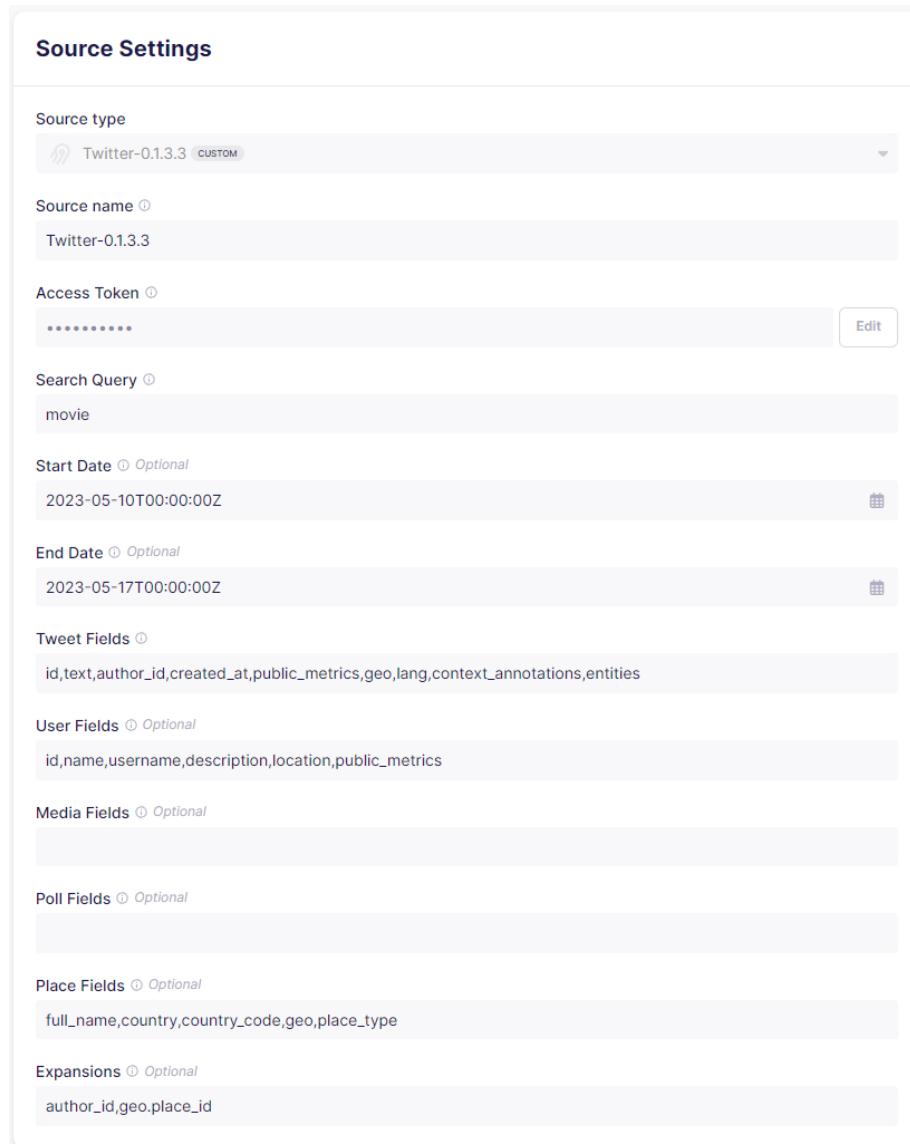


Figura 5.14: Configuración ampliada del conector mejorado de Airbyte para Twitter

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 [6] [41]. 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 [49]. Los usuarios y desarrolladores de la plataforma han organizado numerosas protestas [26] 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.

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)* [13], de la plataforma de *data science* Kaggle²⁰. 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.

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*²¹ 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.

²⁰<https://www.kaggle.com>

²¹https://colab.research.google.com/drive/1d_0bU9idFqjsDi7ezeFs1C0RxTUAgh7V#offline=true&sandboxMode=true

5.2. Carga de datos

Una vez completada la extracción de los datos, resulta necesario persistirlos para su posterior uso. La herramienta seleccionada para realizar esta tarea es *MongoDB* [36].

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*.

5.3. Transformación de los datos

Una vez finalizada la carga inicial de los datos en *MongoDB*, se procede con su procesamiento. Para ello, se ha utilizado *Spark* con el lenguaje *Scala*, ya que presenta una gran velocidad de cómputo ideal al trabajar con grandes cantidades de datos.

Esta tarea contempla la lectura de los datos realizando una limpieza inicial de los mismos, de manera que se recogen únicamente los datos necesarios y se elimina la compleja estructura de datos de la que se han extraído. Posteriormente se seleccionan los campos que serán exportados nuevamente a una colección de *MongoDB* de datos limpios.

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

5.4. Visualización de los datos

En esta sección se detallarán los aspectos relevantes de la herramienta seleccionada para la visualización de los datos.

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* 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.

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*.

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* 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.

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 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.

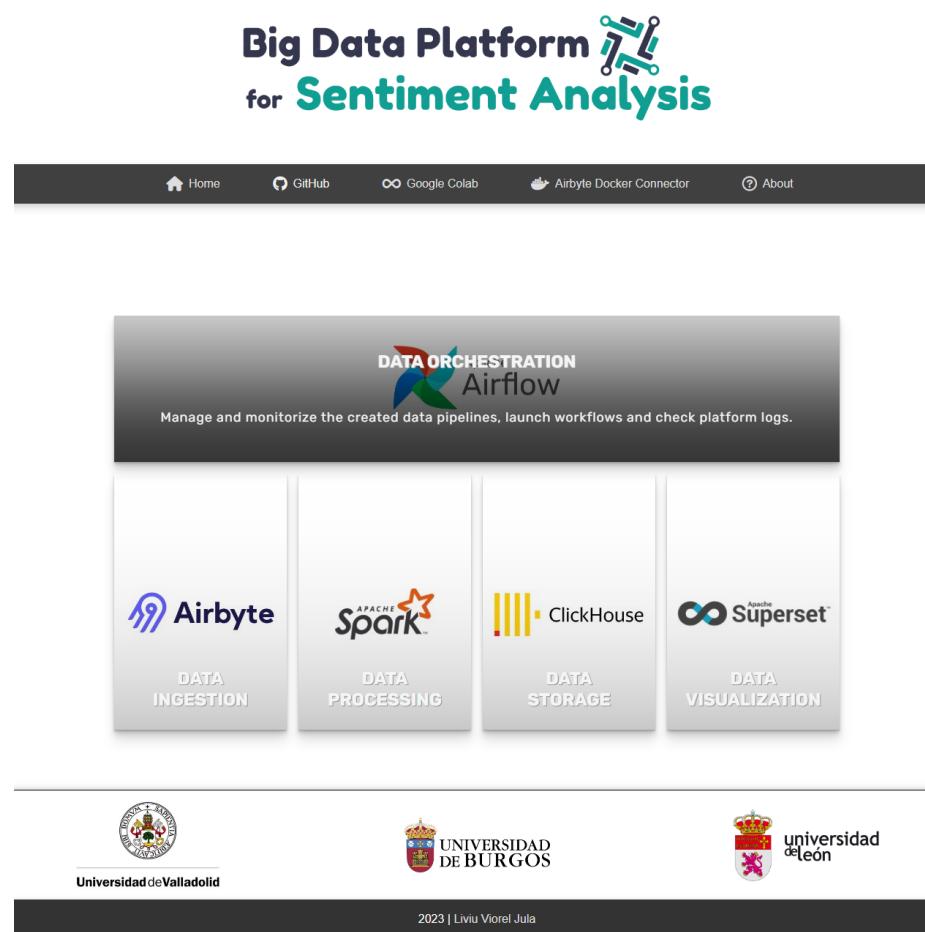


Figura 5.15: Aplicación web desarrollada como punto de acceso centralizado

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 plataforma²² 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...

²²<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 análisis²³ 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.

²³<https://www.repustate.com/>

Social Searcher

Es una herramienta sencilla²⁴ 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 básica²⁵ 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 [16] 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*

²⁴<https://www.social-searcher.com/>

²⁵https://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

Todo proyecto debe incluir las conclusiones que se derivan de su desarrollo. Éstas pueden ser de diferente índole, dependiendo de la tipología del proyecto, pero normalmente van a estar presentes un conjunto de conclusiones relacionadas con los resultados del proyecto y un conjunto de conclusiones técnicas. Además, resulta muy útil realizar un informe crítico indicando cómo se puede mejorar el proyecto, o cómo se puede continuar trabajando en la línea del proyecto realizado.

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

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.

Diseño 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*.

Primera iteración

Los siguientes bocetos capturan la primera iteración del diseño inicial del *dashboard* junto a la interfaz implementada con la herramienta *Apache Superset*.

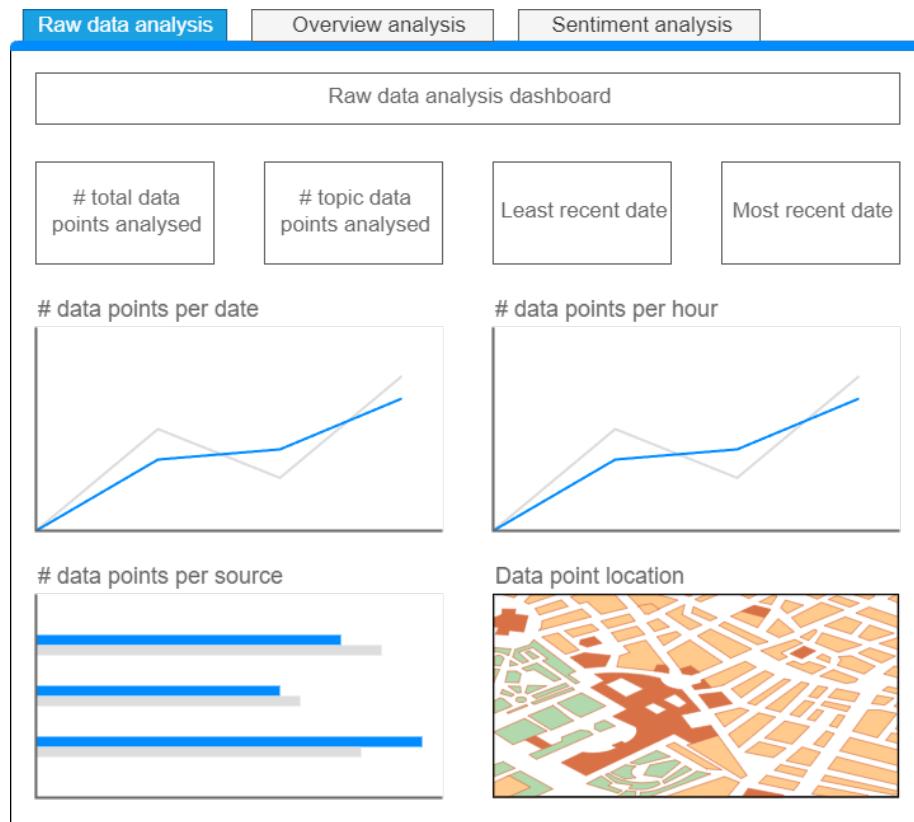


Figura C.1: Diseño inicial de la pestaña *Raw data*

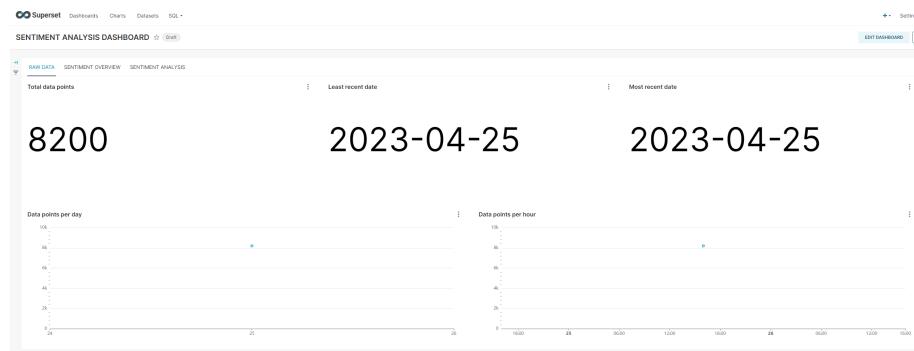
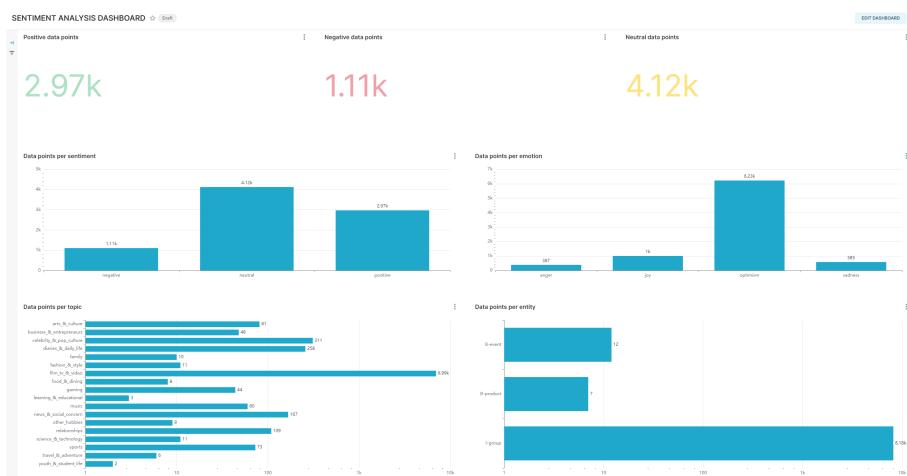


Figura C.2: Primera iteración de la pestaña *Raw data*

Figura C.3: Diseño inicial de la pestaña *Sentiment overview*Figura C.4: Primera iteración de la pestaña *Sentiment overview*

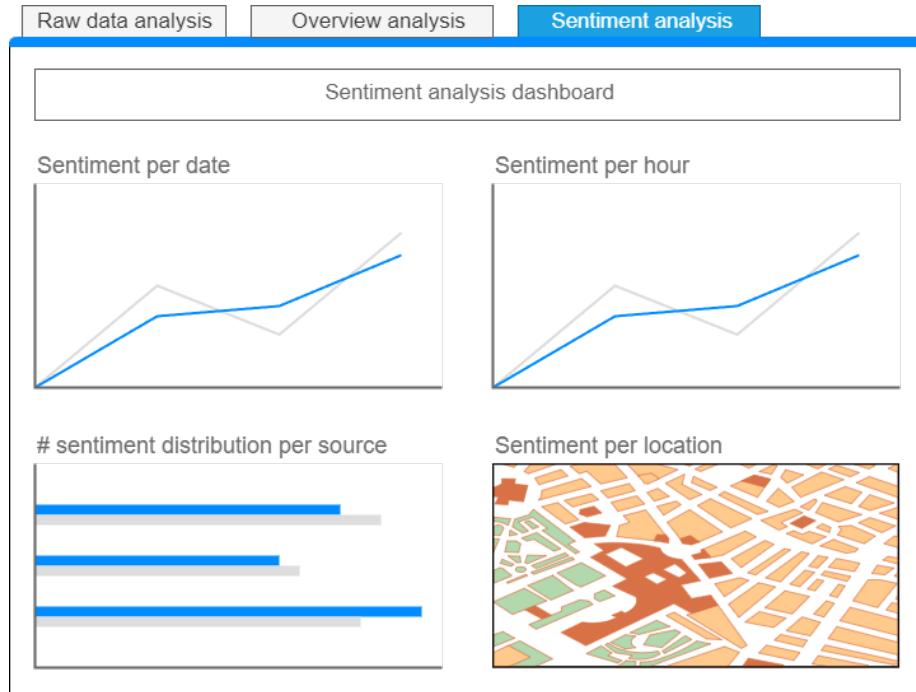


Figura C.5: Diseño inicial de la pestaña *Sentiment analysis*

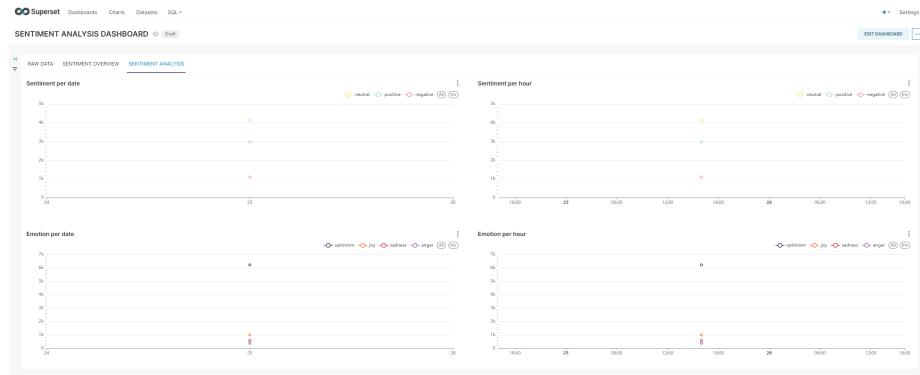


Figura C.6: Primera iteración de la pestaña *Sentiment analysis*

C.3. Diseño procedimental

C.4. Diseño arquitectónico

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

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.

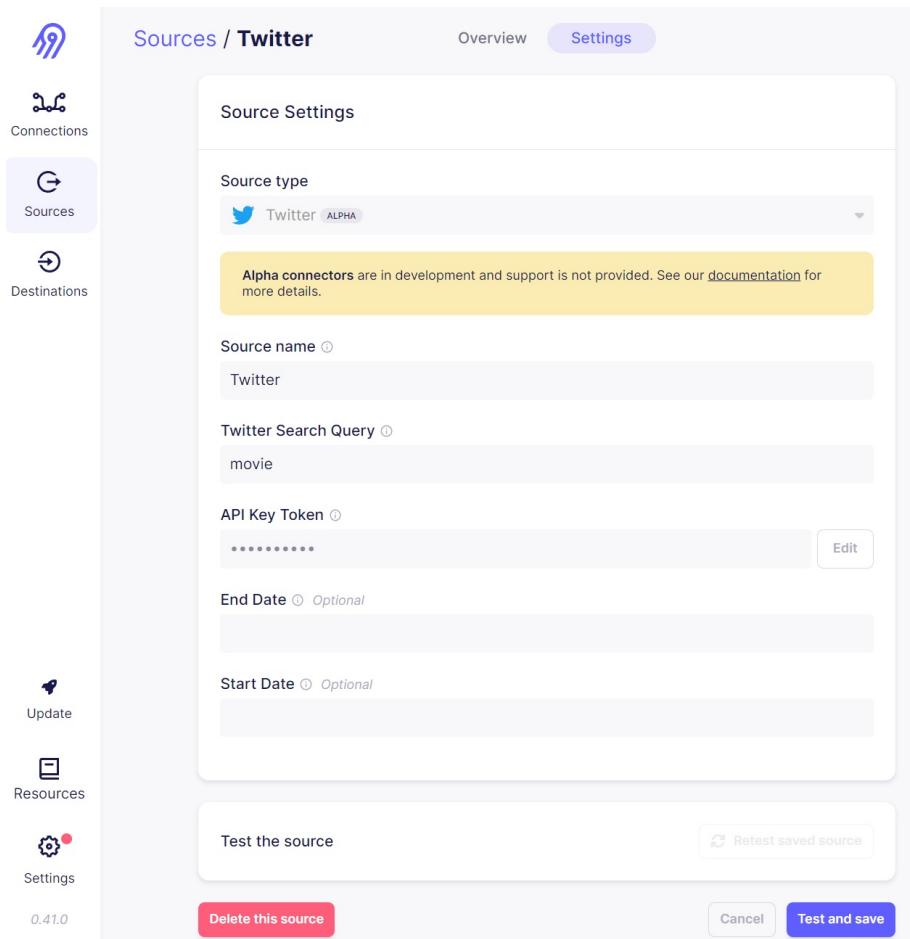


Figura D.1: Configuración del origen de datos en Airbyte: Twitter

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.

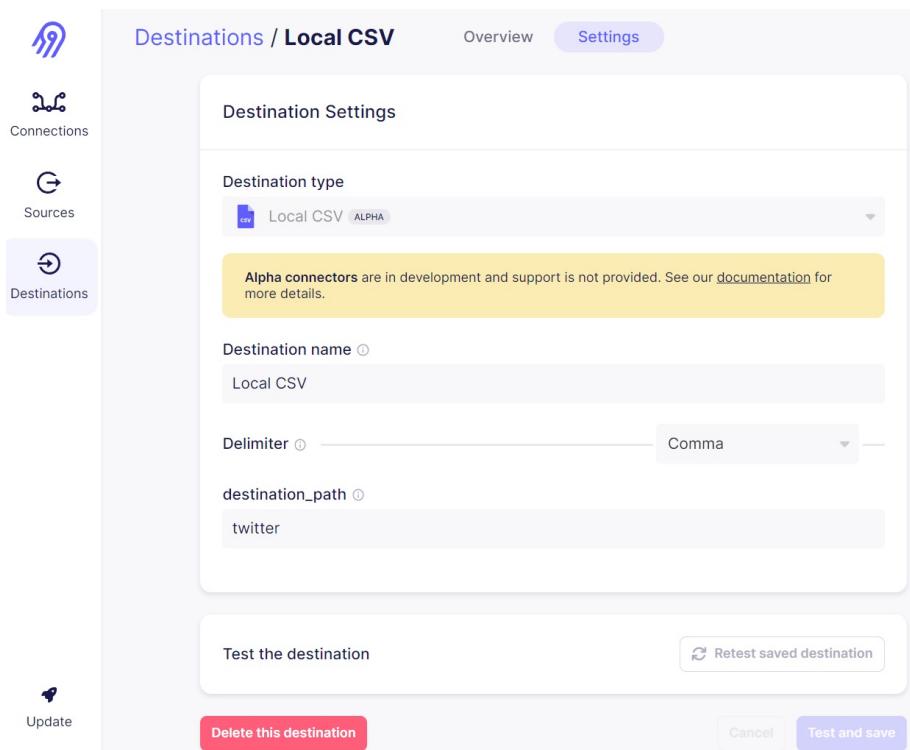


Figura D.2: Configuración del destino de datos en Airbyte: CSV Local

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.

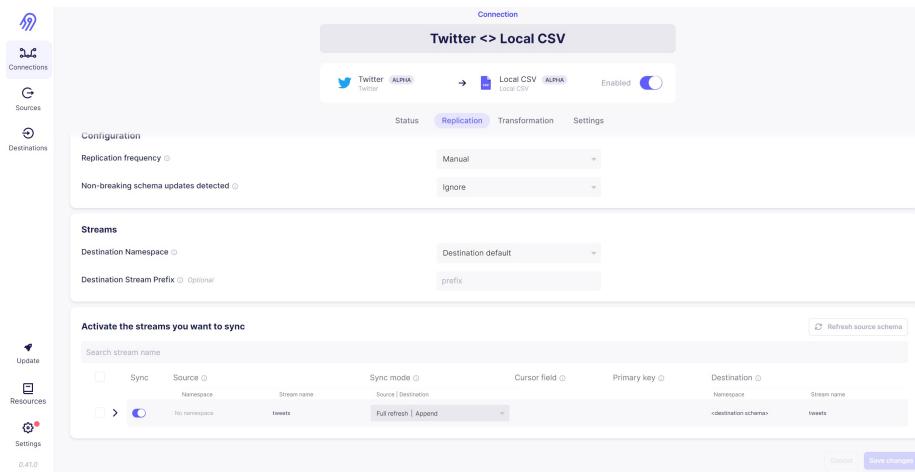


Figura D.3: Configuración de la conexión entre origen y destino de los datos en Airbyte

D.4. Compilación, instalación y ejecución del proyecto

D.5. Pruebas del sistema

Apéndice E

Documentación de usuario

- E.1. Introducción**
- E.2. Requisitos de usuarios**
- E.3. Instalación**
- E.4. Manual del usuario**

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] 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].

- [7] 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].
- [8] Mahdi Bohlouli, Jens Dalter, Mareike Dornhöfer, Johannes Zenkert, and Majid 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].
- [9] 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].
- [10] 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].
- [11] 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].
- [12] Databricks. Extract transform load (etl). [En línea]. Databricks 2023. Disponible en [https://www.databricks.com/glossary/extract-tr ansform-load](https://www.databricks.com/glossary/extract-transform-load), 2023. [Fecha de consulta: 20-06-2023].
- [13] 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].
- [14] 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].
- [15] 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].
- [16] 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].

- [17] 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].
- [18] 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].
- [19] 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].
- [20] 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].
- [21] 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].
- [22] 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].
- [23] IBM. Etl (extract, transform, load). [En línea]. IBM. Disponible en <https://www.ibm.com/topics/etl>, 2023. [Fecha de consulta: 20-06-2023].
- [24] IkarosKun. Python-facebook. [En línea]. GitHub, 2023, 2023. Disponible en <https://github.com/sns-sdks/python-facebook>, 2022. [Fecha de consulta: 28-02-2023].
- [25] Reddit Inc. Api overview. [En línea]. reddit, inc., 2023. Disponible en <https://www.reddit.com/dev/api>, 2023. [Fecha de consulta: 26-02-2023].
- [26] 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].
- [27] 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].
 - [28] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014. [Fecha de consulta: 29-06-2023].
 - [29] 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].
 - [30] 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].
 - [31] 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].
 - [32] 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].
 - [33] 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].
 - [34] 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].
 - [35] 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].
 - [36] 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].

- [37] 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].
- [38] 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].
- [39] 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].
- [40] 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].
- [41] 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].
- [42] 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].
- [43] 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].
- [44] Inc. Twitter. Python client for the twitter api v2 search endpoints. [En línea]. GitHub, 2023. Disponible en <https://github.com/twitterdev/search-tweets-python/tree/v2>, 2023. [Fecha de consulta: 28-02-2023].
- [45] 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].

- [46] 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].
- [47] 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].
- [48] 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].
- [49] 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].
- [50] 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].