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 — 12 de abril
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, 12 de abril 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 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 de figuras

Índice de tablas

Memoria

Introducción

Descripción del contenido del trabajo y del estrucutra de la memoria y del resto de materiales entregados.

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 aquellos proyectos que necesiten para su comprensión y desarrollo de unos conceptos teóricos de una determinada materia o de un determinado dominio de conocimiento, debe existir un apartado que sintetice dichos conceptos.

Técnicas y herramientas

Esta parte 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 [6].

Sentiment Analysis

El análisis de sentimientos 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 [19], 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 exploración 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 lenguae 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 LaTeX

La elaboración de esta memoria está basada en la plantilla LaTeX provista como ejemplo por los Coordinadores del Máster y disponible públicamente[4]. Para facilitar la edición y gestión de esta plantilla, se ha decidido utilizar una herramienta adecuada para ello.

Herramientas consideradas:

- **MiKTeX + TeX:** Herramientas que realizan la traducción de LaTeX a texto y permiten gestionar y editar este tipo de archivos, respectivamente.
- **Overleaf:** Plataforma en línea que facilita la gestión y edición de documentos con formato LaTeX.

Herramienta elegida:

- **Overleaf**

Overleaf es un editor en línea[5] de LaTeX. Para utilizarlo no es necesario realizar la instalación de ningún componente, tiene documentación integrada

[4] https://github.com/bbaruque/plantillaTFM_MUINBDES.git
[5] https://es.overleaf.com/

para LaTeX 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 LaTeX.

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 herramientas 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.

Postman

Es una plataforma[8] 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 [21], 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 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* [23].
- ***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 [12].

[8] https://www.postman.com/

- ***PRAW: The Python Reddit API Wrapper.*** Paquete de *Python* que facilita el acceso a la *API* de Reddit [5].

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 plataforma[9] 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 [3].

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.

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

[9] https://airbyte.com/

alto rendimiento para procesar grandes conjuntos de datos. Complejo de configurar correctamente para conseguir eficiencia óptima [9].

- ***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 [10].
- ***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 [11].
- ***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 [20].

Herramienta elegida:

- ***MongoDB***[10]

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.

[10] https://www.mongodb.com/

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* [22] 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*[11] 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 [24].

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 *Premium* *Enterprise*	Operaciones *CRUD*.
Users	v2 v1.1 *Premium* *Enterprise*	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: [25]

[11] Lenguaje de consultas para *API*s 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* [14], 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 [17] 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 [18].

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

Tabla 5.2: Muestra de nodos disponibles en *Graph API* de Facebook. Fuente: [15]

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 *hashtag*.
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: [16]

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 [7].

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: [8]

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 [26] 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
Comment	Comentario de las demás estructuras de datos.
Subreddit	Representación de un subforo.
Message	Información sobre mensajes.
Account	Datos de la cuenta de un usuario.

Tabla 5.5: Recursos disponibles a través de la *API* de Reddit. Fuente: [13]

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 5.1 se puede observar una visión general de la arquitectura de esta herramienta.

También presenta una *API* propia [1] desde la que es posible gestionar las configuraciones de dichos recursos sin necesidad de acceder a su interfaz web.

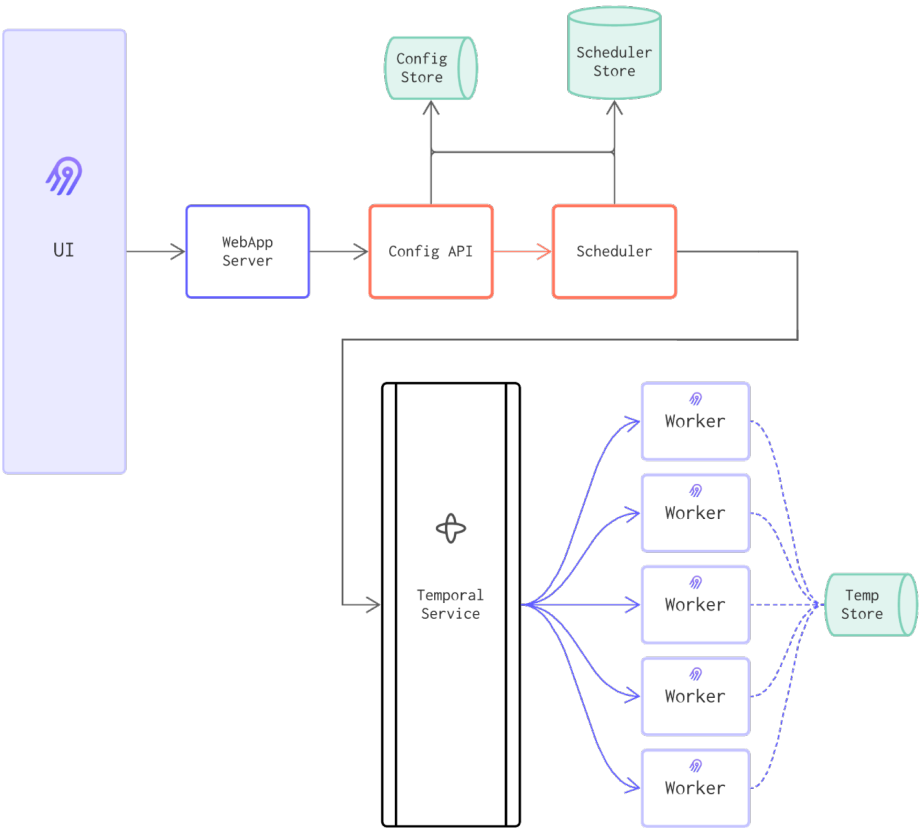

Figura 5.1: Visión general de la arquitectura de *Airbyte*. Fuente: [2]

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* [20].

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*.

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 plataforma[12] 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...

[12] 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 análisis[13] 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.

[13] https://www.repustate.com/

Social Searcher

Es una herramienta sencilla[14] 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 básica[15] 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

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

[14] https://www.social-searcher.com/
[15] https://www.csc2.ncsu.edu/faculty/healey/tweet_viz/tweet_app/
[16] https://arxiv.org/abs/1810.04805

dos componentes, un *encoder* que procesa los datos de entrada y un *decoder* 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 entrenan 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 tal 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 entrenan 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 LaTeX (2h).** Como objetivo para la generación de la memoria del proyecto, se va estudiar una guía sobre LaTeX 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 5.2. 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.

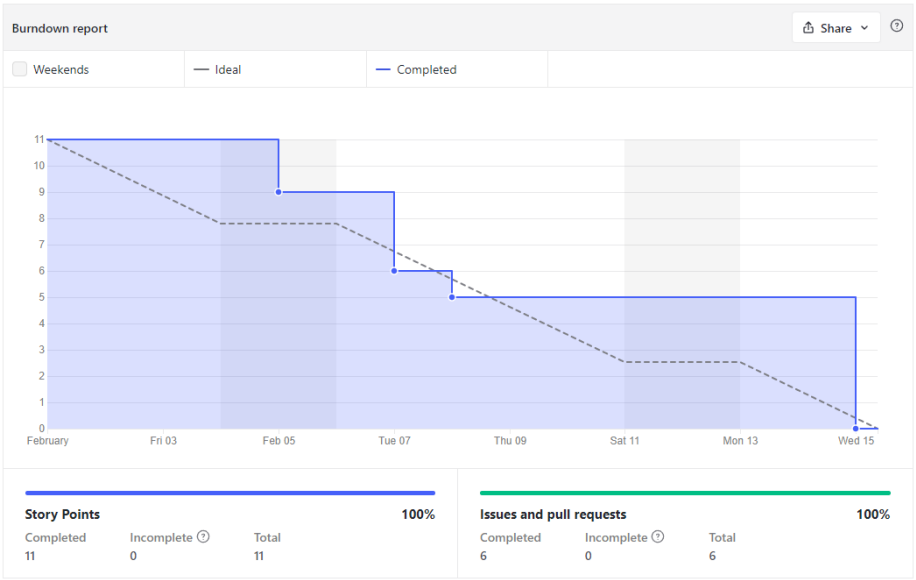

Figura A.1: Gráfico *burn-down* del *Sprint* 0

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 comenzará 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*. Esto se puede ver reflejado en la Figura A.2.

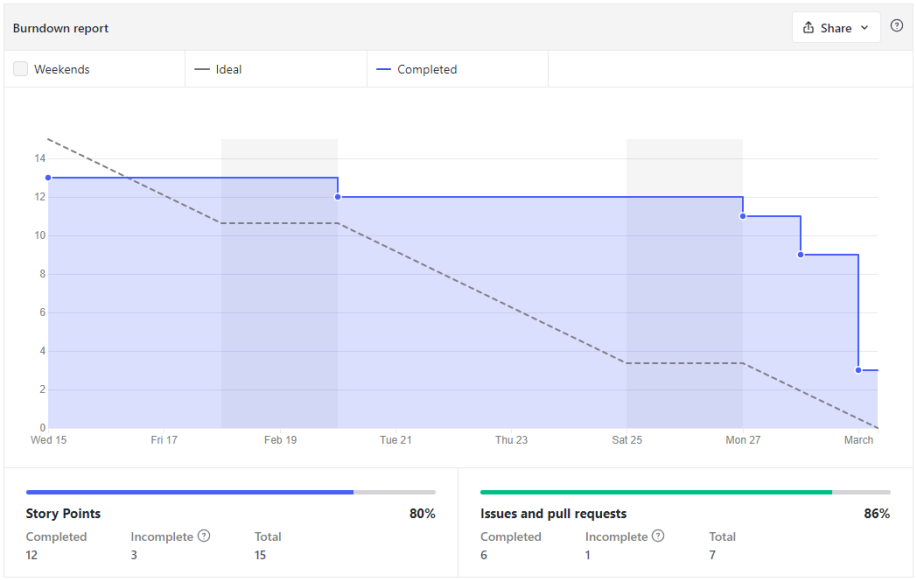

Figura A.2: Gráfico *burn-down* del *Sprint* 1

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 [4].

- **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.

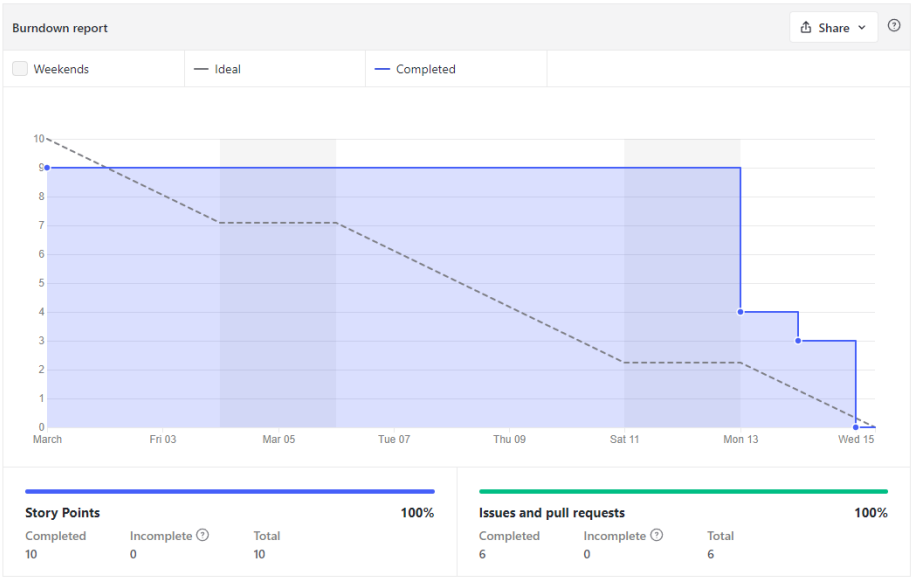

Figura A.3: Gráfico *burn-down* del *Sprint* 2

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 la 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.

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

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 de la fuente 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. **Mandar 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.** Mandando 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.

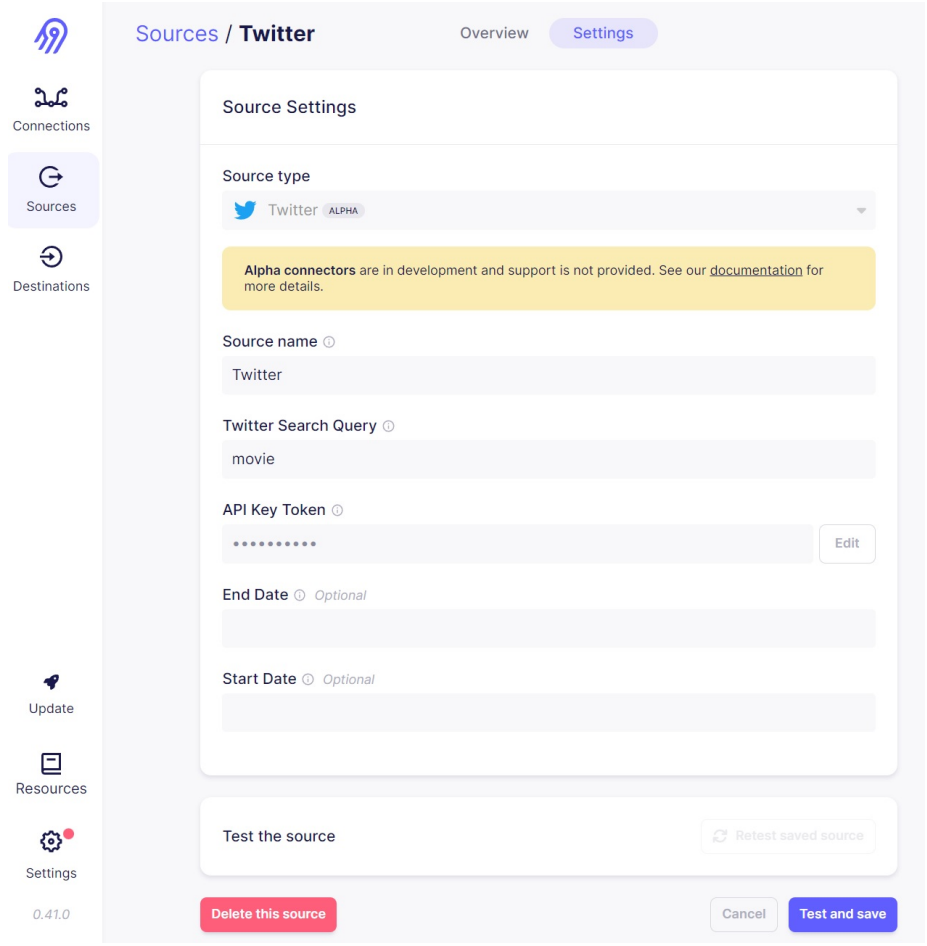

Figura D.1: Configuración de la fuente de datos

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. **Mandar 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.** Mandando 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.

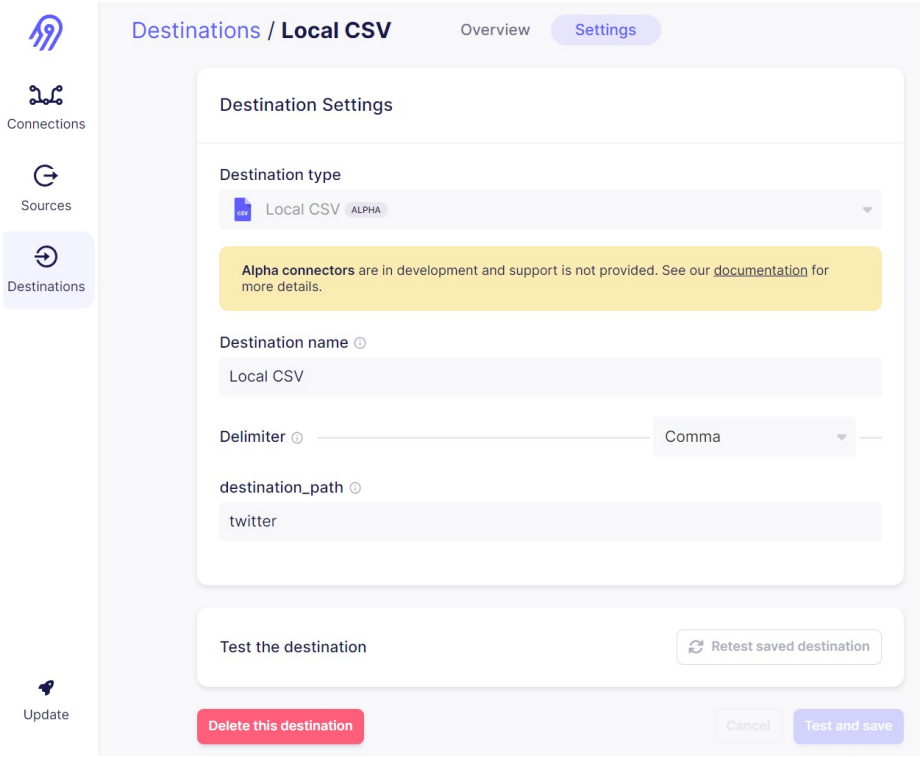

Figura D.2: Configuración de la fuente de datos

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.

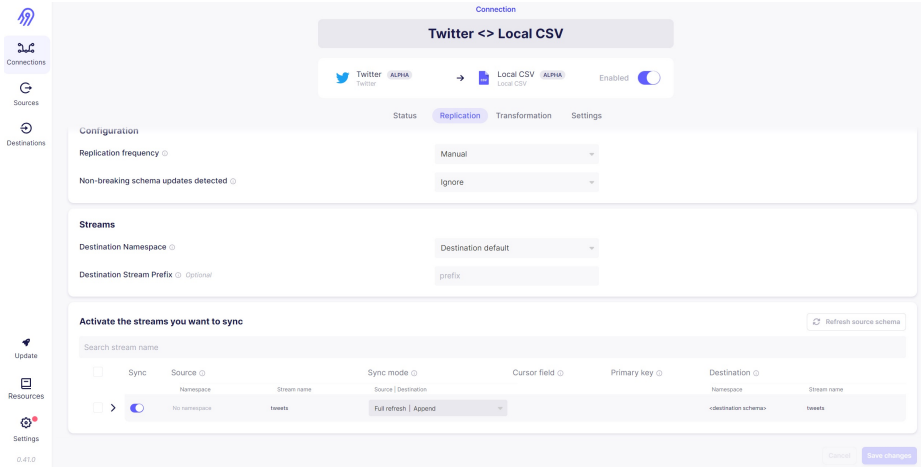

Figura D.3: Configuración de la fuente de datos

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] 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].

[2] 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].

[3] 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].

[4] 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].

[5] 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].

[6] KR1442 Chowdhary and KR Chowdhary. Natural language processing. *Fundamentals of artificial intelligence*, pages 603–649, 2020.

[7] 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].

[8] 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].

[9] 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].

[10] 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].

[11] 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].

[12] IkarosKun. Python-facebook. [En línea]. GitHub, 2023, 2023. Disponible en https://github.com/sns-sdks/python-facebook, 2022. [Fecha de consulta: 28-02-2023].

[13] Reddit Inc. Api overview. [En línea]. reddit, inc., 2023. Disponible en https://www.reddit.com/dev/api, 2023. [Fecha de consulta: 26-02-2023].

[14] 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].

[15] 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].

[16] 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].

[17] 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].

[18] 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].

[19] Kostadin Mishev, Ana Gjorgjevikj, Irena Vodenska, Lubomir Chitkushev, and Dimitar Trajanov. Evaluation of sentiment analysis in finance: From lexicons to transformers. *IEEE Access*, PP:1–1, 07 2020.

[20] 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].

[21] 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].

[22] 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].

[23] 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].

[24] 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].

[25] 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].

[26] 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].