Universidad de Valladolid

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática
Mención en Computación

Plataforma para la extracción y análisis de noticias de la web

Pablo Marcos Parra

Universidad de Valladolid

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática
Mención en Computación

Plataforma para la extracción y análisis de noticias de la web

Autor: **Pablo Marcos Parra**

Tutores: **Valentín Cardeñoso Payo**
Raúl García Serrada (Air Institute)

A todos los que siempre me han apoyado y ayudado, en especial a mis padres.

“Todo tiene que terminar alguna vez. De lo contrario, nada comenzaría nunca.“

Doctor Who

Agradecimientos

A las muchas personas que me han ayudado no solo a lo largo de este proyecto si no a lo largo de toda la carrera.

En primer lugar agradecérselo a todos los profesores de esta etapa, en especial, por supuesto, a mi tutor Valentín que ha estado pendiente desde el primer momento que apareció este trabajo. También a Air Institute y a Raúl por darme la oportunidad de desarrollar este proyecto.

Luego agradecérselo a mis amigos, estoy seguro que sin ellos no estaría donde estoy ahora. Siempre recordaré nuestras noches de juego, nuestras largas sesiones de D&D y las tardes de terraceo al sol.

Por supuesto a toda mi familia: tíos, tías, primos y abuelos que siempre me han estado apoyando.

A mi madre porque siempre, siempre ha confiado y creído en mí y me ha apoyado en los momentos más duros; y a mi difunto padre por haberme hecho el hombre que soy hoy.

Resumen

En la actualidad el flujo de noticias e informaciones es más grande que nunca y es necesario someter a escrutinio público el valor que aportan a la sociedad algunas de estas noticias. Actualmente, la única manera de analizar dichas noticias es recurriendo a una persona o grupos de personas para que extraigan las noticias de la web y vayan estudiando la información que aportan dichas noticias, todo ello de manera manual. Es una tarea importante, pero costosa y repetitiva.

Este trabajo se centra en la automatización de la extracción y análisis de las noticias de la web mediante la creación de una plataforma web que permita acceder y filtrar rápidamente las noticias y realizar un análisis preliminar de su estructura y otros aspectos relevantes, como el análisis de sentimientos.

Para la parte de extracción, se han extraído las noticias de diferentes fuentes web y blogs de noticias en castellano mediante métodos de *web scraping*, se han estructurado mediante un proceso ETL (Extract, Transform y Load) y se han almacenado en una base de datos.

Para la parte de análisis, se han aplicado técnicas de Procesamiento de Lenguaje Natural (PLN). Primero, se realiza un análisis de sentimientos sobre la noticia y posteriormente, un Reconocimiento de Entidades Nombradas (REN) para identificar a las organizaciones, lugares o personas mencionadas.

Finalmente, se crea una plataforma web donde se muestran las noticias extraídas y el resultado del análisis.

Abstract

Nowdays, the flow of news and information is greater than ever and the value that some of this news brings to society needs to be subjected to public scrutiny. Currently, the only way to analyze the news is through the participation of a person or group of people who extract the news from the web and manually annalyze the information provided by the news. It is an important but extensive and repetitive task.

The objective of this work is to automate the extraction and analysis of the news from the web by creating a platform that allows to quickly access and filter the news and perform a preliminary analysis of its structure and other relevant aspects, such as sentiment analysis.

For the extraction part, the news have been extracted from different web sources and news blogs in Spanish using web scraping methods, structured through an ETL process and stored in a database.

For the analysis part, Natural Language Processing (NLP) techniques are applied. First, a sentiment analysis is performed on the news item and then, a Named Entity Recognition (NER) to identify the organizations, places or people mentioned.

Finally, a web platform has been created where the extracted news and the result of the analysis are displayed.

Índice general

Índice de cuadros

Índice de figuras

Parte I

Objeto, Concepto y Método

Capítulo 1

Introducción

1.1 Introducción

Con el nacimiento de Internet a finales del siglo XX, el acceso a la información pasó de ser limitado y controlado por determinados actores y medios a estar disponible de forma casi inmediata a través de un *click*. Las nuevas posibilidades planteadas por las nuevas tecnologías han llevado a nuestra sociedad a una explosión en la cantidad y disponibilidad de noticias y opiniones. Vivimos en la era de la información donde el acceso a las noticias se ha vuelto más amplio y rápido que nunca.

Sin embargo, junto con esa abundancia de información, han aparecido nuevos desafíos significativos en la forma en la que las noticias se producen, distribuyen y consumen. La necesidad de un análisis crítico y riguroso de las noticias se vuelve fundamental para comprender la complejidad de los acontecimientos y su impacto en la sociedad.

En tiempos recientes, han comenzado aparecer con mayor frecuencia las noticias falsas conocidas comúnmente como *"fake news"*. Para combatir la desinformación generada por estas noticias falsas comenzaron a aparecer los llamados *"fact checkers"*, equipos de personas dedicadas exclusivamente a la búsqueda, detección y análisis de noticias falsas. Estos equipos su trabajo de manera completamente manual. Es una tarea repetitiva que puede ser ayudada por las nuevas herramientas tecnológicas que han aparecido en los últimos años.

Este trabajo fin de grado, realizado bajo convenio con la empresa Air Institute [52] como parte del proyecto EthicalNews [51], un proyecto realizado en colaboración con Eurostar y financiado por el Ministerio de Ciencia e Innovación dentro del Programa Estatal de I+D+i Orientada a los Retos de la sociedad. Su objetivo principal del proyecto incluye un apartado de investigación de un nuevo marco de Procesamiento del Lenguaje Natural Ético y eXplicable dirigido a la detección de noticias falsas y a la prevención de

la desinformación.

1.2 Motivación

Dentro del proyecto EthicalNews [51], donde se enmarca este trabajo fin de grado, es necesario un primer paso de creación de un corpus o dataset de texto para luego implementar los diferentes algoritmos para su posterior análisis. Por ello se propuso la creación de esta plataforma para la extracción y procesamiento inicial de los textos tanto del cuerpo como de los titulares de las noticias de diferentes fuentes.

En el caso de las revistas o webs de noticias de procedencia española, dicha extracción es una tarea complicada de automatizar debido a la ausencia de una API propia de cada web (véase la API del periódico New York Times [86]) o una API general que extraiga las noticias de todo el país (veáse NewsAPI [64]).

Es por ello que se plantea la creación de una aplicación para el proyecto, que automatice la extracción y procesamiento de noticias de algunas fuentes de noticias reconocidas, de fuentes de noticias satíricas y de fuentes de noticias altamente sesgadas ideológicamente.

1.3 Objetivos

Los objetivos planteados para este trabajo fin de grado son:

- **Objetivos principales:**
  - Crear una aplicación que permita automatizar la recolección de noticias de diferentes fuentes.
  - Aplicar un análisis preliminar de texto a las noticias extraídas (análisis de sentimientos y análisis de entidades nombradas).
  - Estudiar las posibles tecnologías, librerías y *frameworks* para la consecución de los puntos anteriores, así como para posibles implementaciones futuras dentro del proyecto *Ethical News* [51].
- **Objetivos secundarios:**
  - Crear una plataforma web que permita mostrar los resultados de la extracción.
  - Al entrar en una noticia, mostrar el resultado del análisis realizado.

1.3.1 Tareas a realizar

La consecución de los anteriores objetivos conlleva las siguientes tareas esenciales:

1. Estudio y comparación de las diferentes tecnologías, librerías y *frameworks* para cada uno de los puntos siguientes.
2. Diseño e implementación de *Web Crawlers* y *Web Scrapers* en diversas webs y blogs de noticias en castellano, escogiendo al menos 4 fuentes diferentes. La noticias extraídas serán subidas a una base de datos tras ser procesadas mediante procesos ETL.
3. Diseño e implementación de las bases de datos con las noticias extraídas.
4. Implementación de procesos de Procesamiento de Lenguaje Natural sobre las noticias: análisis de sentimientos y reconocimiento de entidades.
5. Diseño e implementación de una plataforma web para la visualización de resultados.

1.4 Estructura de la memoria

La memoria del presente trabajo fin de grado se estructurará de la siguiente manera:

- **Capítulo 1 - Introducción**: planteamiento del trabajo, contexto, motivación, objetivos y tareas.
- **Capítulo 2 - Metodología**: organización del trabajo, riesgos y costes.
- **Capítulo 3 - Marco conceptual**: aspectos del *Web Crawling* y el *Web Scraping*, así como del Procesamiento del Lenguaje Natural incluyendo las diferentes partes del mismo y las diferentes redes de *Deep Learning* que se usan.
- **Capítulo 4 - Fuentes de datos**: origen y selección de las fuentes de datos.
- **Capítulo 5 - Análisis**: qué es lo que se necesita para el desarrollo del sistema, requisitos y casos de uso.
- **Capítulo 6 - Diseño**: diseño de la solución software propuesta para cumplir con los requisitos y casos de uso propuestos en el análisis.
- **Capítulo 7 - Implementación**: descripción del proceso de implementación de la solución planteada en el diseño.
- **Capítulo 8 - Pruebas**: resultado final de la aplicación web implementada en el capítulo 7, con las pruebas realizadas y ejemplos de uso.

- **Capítulo 9 - Conclusiones y trabajo futuro**: resumen de lo que se ha conseguido en el proyecto, así como posibles líneas de trabajo en el futuro.
- **Apéndice A - Manual de Usuario**: cómo utilizar el sistema.
- **Apéndice B - Manual de Instalación**: cómo instalar y configurar el sistema.

Capítulo 2

Metodología

2.1 Introducción

En este capítulo se mostrará el plan de desarrollo para el software propuesto para este trabajo fin de grado, para ello se ha elegido aplicar la metodología RUP (Rational Unified Process, Proceso Unificado de Rational). La metodología RUP es un sistema para procesos de desarrollo de software desarrollado por la empresa Rational Software (propiedad de IBM), cuyo enfoque es que el desarrollo sea iterativo e incremental. Constituye uno de los estándares más utilizados para análisis, diseño e implementación de software.

2.2 Fases del proyecto

RUP impone una división del proyecto en 4 fases, cada una de ellas subdividida en tareas que deben ser completadas antes de avanzar a la siguiente fase .

2.2.1 Fase de inicio

En la fase de inicio, el objetivo principal es definir correctamente el alcance del sistema. Para ello se define la planificación inicial del proyecto, estableciendo cosas como los requisitos del sistema, casos de uso y gestión de riesgos. Esta planificación inicial puede ser modificada a lo largo del proyecto en caso de ser necesario, pero sirve de base para el resto de fases.

2.2.2 Fase de elaboración

En la fase de elaboración es donde se realiza la parte de análisis detallado del sistema que se va a construir. Esto incluye la realización del análisis de dominio, creación de diagramas y arquitectura del sistema. En esta parte se desarrolla y valida el plan inicial y se realiza la primera implementación del sistema.

2.2.3 Fase de construcción

En la fase de construcción es donde se desarrolla e implementa el sistema de software. Se realiza una serie de iteraciones en las que se desarrollan y prueban cada uno de los componentes del sistema. En esta fase se recomienda usar buenas prácticas como desarrollar iterativamente, tener siempre en cuenta los requisitos, usar componentes individuales, verificar la calidad y establecer un control de cambios y versiones.

2.2.4 Fase de transición

Es la última fase, donde se entrega el sistema. Se pone en funcionamiento y se comprueba que cumpla con todos los requisitos planteados, identificando también posibles área de mejora.

2.3 Artefactos del proyecto

- Plan de desarrollo del software
- Análisis y diseño del problema
- Implementación del sistema
- Test y pruebas del sistema

2.4 Calendarización

Como se ha indicado anteriormente, las fases se han determinado siguiendo la metodología que se ha elegido. El reparto de horas por fases se muestra en la siguiente tabla:

Fase	Nº Iteraciones	Duración estimada
Inicio	1	25
Elaboración	1	75
Construcción	2	150
Transición	1	50
Total	5	300

Cuadro 2.1: Resumen de la temporalización de las fases de planificación

La estimación inicial de la temporalización suma un total de 300 horas, que es la cantidad de horas determinadas para el Trabajo Fin de Grado de Ingeniería Informática, ya que supone una carga lectiva de 12 créditos ECTS (25 horas por crédito).

2.4.1 Fase de inicio

La fase de inicio tiene una duración estimada de 25 horas con las siguientes tareas:

Tarea	Duración
Estudio del sistema	5 horas
Elicitación de requisitos	4 horas
Estudio de herramientas	10 horas
Creación de la planificación (RUP)	5 horas
Gestión de riesgos	1 hora
	25 horas

Cuadro 2.2: Tareas fase de inicio

2.4.2 Fase de elaboración

La fase de elaboración tiene una duración estimada de 75 horas con las siguientes tareas:

Tarea	Duración
Especificación de requisitos	2 horas
Elicitación de casos de uso	3 horas
Diagrama casos de uso	2 horas
Realización de los casos de estudio	5 horas
Diseño de la Base de Datos	3 horas
Investigar librerías Web Scraping en Python	15 horas
Investigar librerías Procesamiento de Lenguaje Natural en Python	20 horas
Investigar desarrollo web con Python	20 horas
Diseño de mockups de la interfaz de la web	5 horas
	75 horas

Cuadro 2.3: Tareas fase de elaboración

2.4.3 Fase de construcción

La fase de construcción tiene una duración estimada de 150 hora, divididas en dos iteraciones con las siguientes tareas:

Tarea	Duración
Implementación de la Base de Datos	2 horas
Implementación de los Web Scrapers	20 horas
Implementación de algoritmos Procesamiento de Lenguaje Natural	20 horas
Implementación de la plataforma web	35 horas
Pruebas de funcionalidad	15 horas
Documentación de los programas	8 horas
	100 horas

Cuadro 2.4: Tareas fase de construcción (iteración 1)

Tarea	Duración
Corrección de problemas	10 horas
Despliegue con Docker	20 horas
Nuevas pruebas de funcionalidad	15 horas
Documentación de los programas	5 horas
	50 horas

Cuadro 2.5: Tareas fase de construcción (iteración 2)

2.4.4 Fase de transición

La fase de transición tiene una duración estimada de 50 horas con las siguientes tareas:

Tarea	Duración
Revisión de la documentación	4 horas
Creación del manual de usuario	3 horas
Creación del manual de instalación	3 horas
Revisión de la memoria	40 horas
	50 horas

Cuadro 2.6: Tareas fase de transición

2.5 Entregables

- **Fase de inicio**
  - Requisitos
  - Gestión de riesgos
  - Alcance y objetivos
- **Fase de elaboración**
  - Definición de los casos de uso
  - Diagramas de dominio
  - Diagramas de secuencia
  - Diagramas de caso de uso
  - Arquitectura del sistema
  - Diseño inicial, mockups
- **Fase de construcción**
  - Versión inicial del sistema
  - Resultados de pruebas
- **Fase de transición**
  - Versión final del sistema
  - Manual de usuario
  - Manual de instalación
  - Memoria del TFG

2.6 Análisis de riesgos

En este apartado se van a definir los posibles riesgos que pueden aparecer a lo largo del desarrollo del proyecto. Se plantearán los riesgos así como una probabilidad estimada de que ocurran, su impacto, la estrategia a seguir en caso de que aparezcan y planes de contingencia que prevengan dichos riesgos.

R0	Cambios en las fuentes de datos
Descripción	La estructura de las webs a extraer cambia y el nombre de algunos elementos a extraer también
Consecuencias	Algunas partes o todas las partes del Web Scraper dejan de funcionar
Probabilidad	Media
Impacto	Alto
Mitigación	Modularizar al máximo el Web Scraper para poder identificar y solucionar rápidamente los fallos

Cuadro 2.7: Riesgo 0

R1	Falta de disponibilidad
Descripción	El alumno se encuentra ausente por enfermedad u otros problemas
Consecuencias	El proyecto deja de avanzar
Probabilidad	Media
Impacto	Bloqueante
Mitigación	No aplica

Cuadro 2.8: Riesgo 1

R2	Fallo del servidor con la base de datos
Descripción	No se puede establecer conexión con el servidor que contiene las bases de datos con noticias debido a problemas en el equipo local o en el servidor
Consecuencias	El proyecto deja de avanzar
Probabilidad	Baja
Impacto	Bloqueante
Mitigación	Intentar reestablecer la conexión lo antes posible

Cuadro 2.9: Riesgo 2

R3	Pérdida de entregables
Descripción	Se pierden datos o información del proyecto
Consecuencias	Tener que volver a realizar el entregable perdido
Probabilidad	Muy baja
Impacto	Bloqueante
Mitigación	Utilizar control de versiones en la nube como GitHub

Cuadro 2.10: Riesgo 3

R4	Caída de la red durante extracción de datos
Descripción	Se pierde la conexión a Internet mientras se realiza la extracción de las noticias de las webs
Consecuencias	Se detiene la extracción de noticias
Probabilidad	Media
Impacto	Medio
Mitigación	Implementar un sistema de recuperación que permita continuar la extracción desde donde falló

Cuadro 2.11: Riesgo 4

R5	Mala planificación
Descripción	Mala previsión en el tiempo que pueda requerir una tarea que se descubre durante su realización
Consecuencias	Retraso respecto a la planificación inicial
Probabilidad	Media
Impacto	Alto
Mitigación	Replanificar las siguientes tareas del proyecto teniendo en cuenta la temporalización actual

Cuadro 2.12: Riesgo 5

R6	Desconocimiento del funcionamiento de las herramientas usadas
Descripción	A lo largo del proyecto se usarán diferentes tecnologías y herramientas, algunas de las cuales es posible que el alumno no conozca de forma parcial o total.
Consecuencias	Retraso respecto a la planificación inicial, reducción en la calidad del proyecto
Probabilidad	Alta
Impacto	Medio
Mitigación	Usar herramientas con las que el alumno esté familiarizado

Cuadro 2.13: Riesgo 6

2.7 Seguimiento del proyecto

Etapa	**Duración estimada**	**Duración real**
Inicio	25 horas	25 horas
Elaboración	75 horas	100 horas
Construcción (iteración 1)	100 horas	150 horas
Construcción (iteración 2)	50 horas	100 horas
Transición	50 horas	50 horas

Cuadro 2.14: Seguimiento: horas estimadas vs horas reales

Etapa	**Fecha inicial**	**Fecha final**
Inicio	10 de octubre 2022	10 de noviembre 2022
Elaboración	11 de noviembre 2022	31 de enero 2023
Construcción (iteración 1)	1 de febrero 2023	31 de marzo 2023
Construcción (iteración 2)	1 de abril 2023	31 de mayo 2023
Transición	1 junio 2023	16 de junio 2023

Cuadro 2.15: Seguimiento: fechas de realización de las fases

El plan inicial era presentar este Trabajo Fin de Grado en febrero de 2023, pero debido a la aparición continuada de algunos riesgos mencionados anteriormente se ha retrasado hasta la convocatoria ordinaria en junio. Estos riesgos que han aparecido de forma regular son:

- R0 (Cambios en las fuentes de datos)
- R1 (Falta de disponibilidad)
- R5 (Mala planificación)
- R6 (Desconocimiento del funcionamiento de las herramientas usadas): este riesgo se ha visto especialmente en la parte de desarrollo web y en la parte de despliegue del servidor con `Docker`

Parte II

Marco Conceptual y Contexto

Capítulo 3

Marco Conceptual

3.1 *Web Crawling* y *Web Scraping*

El *Web Crawling* y el *Web Scraping* son dos métodos usados para la extracción y recolección de páginas webs y su información. Son dos técnicas, que al igual que en este proyecto, suelen ser usadas de manera conjunta lo que hace que a veces las personas intercambien su significado y las confundan [60]. A continuación se explica en que consiste cada una de las técnicas.

3.1.1 *Web Crawling*

Se conoce como *Web Crawling* al proceso en el cual, mediante el uso de *bots* (también conocidos como *spiders*), se navega sistemáticamente por las páginas web indexando su contenido.[82]

Los *Web Crawlers* suelen ser mal vistos por parte de los desarrolladores de webs, ya que al navegar por la web actúan como un usuario real más, pero realizan muchas más peticiones que el usuario humano realizándolas de manera rápida y consecutiva, lo que incrementa la carga de los servidores. Es por ello que suelen implementarse soluciones *anti-crawling*, tales como límite de peticiones por IP o *user-agent*. [66]

Algunos ejemplos muy conocidos de *Web Crawlers* son los motores de búsqueda, como Bing o Google, que usan el *Web Crawling* para extraer toda la información de una página web e indexarla en su motor de búsqueda [72]. De esta forma, cuando se introduce un término en dichos motores de búsqueda, los resultados que se muestran contienen la información pedida o gran parte de ella.

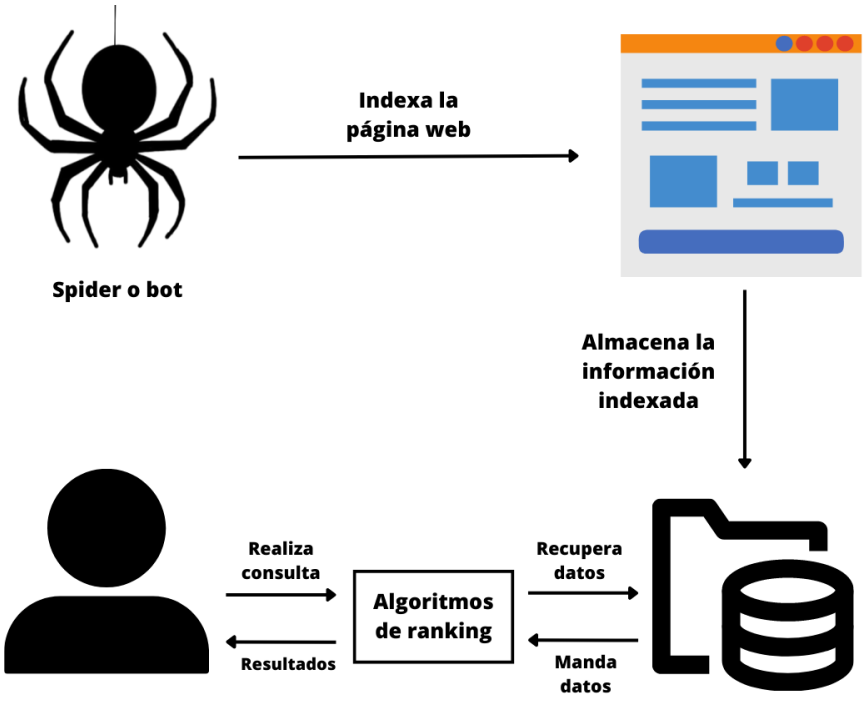


Figura 3.1: Funcionamiento básico de un Web Crawler

3.1.2 *Web Scraping*

El término *Web Scraping* se refiere al proceso de extracción de una información concreta de una página web.[82] Dicha extracción se suele producir sobre la página extraída e indexada anteriormente con el Web Crawler, pero tras ser *Parse* para arrojar resultados con el contenido que se desea.[72]

Existen varias técnicas y herramientas para realizar *Web Scraping*, que van desde la extracción de forma manual hasta la extracción completamente automatizada. Algunas de estas técnicas son:

- **Human copy-and-paste (manual)**: es la forma más sencilla de *Web Scraping*. Consiste en copiar y pegar manualmente el contenido de una página web y pasarla un archivo de texto o una hoja de cálculo. A pesar de ser manual, es una técnica muy útil a la hora extraer información de webs que previenen la extracción automática.
- **Text pattern matching (automática)**: es una aproximación sencilla a la extracción de información de las páginas webs. Se basa en el uso del comando `grep` de UNIX o de expresiones regulares para extraer información que coincida con los parámetros que se desean [94].
- **HTML parsing (automática)**: consiste en obtener la estructura de datos de la web y traducirla a un formato relacional llamado *wrapper* que sirve para establecer plantillas de extracción que pueden ser identificadas en la estructura de las páginas

HTML. Esta es la técnica usada por la librería de `Python BeautifulSoup` que se explicará en el siguiente capítulo.

- **DOM parsing (automática)**: esta técnica se basa en que el programa del Web Scraper funcione como un navegador completo (como `Firefox` o `Google Chrome`), en el cual el *Web Scraper* tome el control de las acciones del lado del cliente y permita obtener contenido dinámico generado por *scripts* de la página web [94]. Esta es la técnica usada por la librería `Selenium`, que también se explicará en el siguiente capítulo.

- **Análisis de páginas web mediante *Machine Learning* (automática)**: es una de las técnicas más avanzadas y novedosas. Se basa en usar Visión de Computadora (*Computer Vision*) junto con otras técnicas de *Machine Learning* que permita identificar y extraer la información de las páginas webs de forma visual de manera similar a como lo haría el ojo humano [84].

3.2 Procesamiento Lenguaje Natural

El Procesamiento del Lenguaje Natural (PLN), comunmente conocido como *Natural Language Processing* (NLP), se refiere a la rama de la Inteligencia Artificial (IA), que se encarga de otorgar a los ordenadores la capacidad de entender texto y lenguaje oral de manera similar a como lo hacen los humanos.[11]

El NLP combina el estudio de la lingüística, su entendimiento, desarrollo y reglas, con modelos informáticos y estadísticos de *Machine Learning* y *Deep Learning* de forma que permite que los ordenadores procesen el lenguaje humano, de forma que la interacción no pierda el sentido y sentimiento que se quiere expresar.

La mayor dificultad a la hora de tratar y modelizar el lenguaje humano es que está lleno de ambigüedades y dobles sentidos, lo que hace que sea muy complicado determinar el sentido real de los textos. Algunos ejemplos de estas irregularidades en el lenguaje escrito son palabras que se escriben igual pero con distinto significado (palabras homónimas), sarcasmos, frases hechas, modismos o metáforas. En cuanto al lenguaje oral, aparecen otros problemas a mayores, tales como inflexiones en la voz, cambios de velocidad o acentos. [49]

Algunos de los problemas que se abordan en este campo incluyen, a modo de ejemplo:

- **Reconocimiento de voz:** conocido también como *speech-to-text* (STT), en el cual se extrae y reconstruye texto a partir de la voz de una persona en un audio. Se comparan los fonemas creados por el interlocutor con los modelos ya entrenados que contienen los fonemas de frases, oraciones y palabras conocidas [49].

- **Traducción automática:** consiste en pasar el texto de un idioma a otro. El mayor problema con la traducción automática, es la existencia de modismos y frases hechas que no tienen una traducción figurada que permita trasladar el sentido de un idioma a otro [49].
- **Resumen de texto:** es una tarea en la cual se introduce a un programa una gran cantidad de texto, se procesa y se genera un resumen o síntesis de su contenido. Los programas de este tipo más básicos se basan en la frecuencia en la que aparecen frases o palabras, mientras que los mejores programas de resumen de texto tienen en cuenta también el contexto del contenido [47].
- **Generación de texto:** es la tarea de crear texto de forma que el texto creado tenga sentido y se parezca al lenguaje humano real. Los mejores generadores de texto intentan que el texto creado sea lo más indistinguible posible de los textos humanos [44]. Algunos de los generadores de texto más populares se basan en modelos computacionales que ya han sido preparados y entrenados para la tarea, como GPT-3 (OpenAI [67]) o BERT [3] (Google).
- ***Chatbots* (agentes conversacionales):** son programas que simulan y procesan conversación humana, de forma oral o escrita, sirviendo de interfaz de comunicación entre los ordenadores y las personas de una forma más humana.

  Existen varios tipos de *Chatbots*, entre los que destacan:

  - ***Chatbots* declarativos:** son *Chatbots* básicos, creados para una sola función. Suelen resolver correctamente interacciones para las que han sido programados, aunque más allá de su programación no son capaces de responder [68]. Un ejemplo de *Chatbots* declarativos son los *Chatbots* de FAQ (preguntas frecuentes) que suelen aparecer en webs de comercios electrónicos para ofrecer ayuda al usuario.
  - ***Chatbots* conversacionales:** son *Chatbots* más avanzados, capaces de ofrecer respuestas más sofisticadas, interactivas y personalizadas. Estos *Chatbots* suelen tener en cuenta el contexto del texto que se plantea y aprenden de las interacciones nuevas. Un ejemplo son los asistentes virtuales como Alexa (Amazon) o Siri (Apple) [68]. En diciembre de 2022, se publicó ChatGPT (OpenAI [67]), uno de los *Chatbots* más avanzados hasta la fecha que es capaz de responder de forma coherente peticiones muy complejas.

Figura 3.2: *Chatbot* declarativo de asistencia de la compañía aérea Norwegian Airlines (22/02/2023)

Figura 3.3: Ejemplo de conversación con ChatGPT (21/02/2023)

3.2.1 Pre-procesamiento de texto

A la hora de trabajar con métodos de NLP, será necesario realizar una serie de transformaciones a la entrada original para poder obtener mejores resultados. Los pasos que normalmente se realizan son:

- **Normalización:** como se ha mencionado anteriormente, el lenguaje humano tiende a ser muy variable y lleno de ambigüedades, lo que dificulta el análisis de texto, por tanto se aplica en primer lugar un proceso de normalización en el cuál se realizan transformaciones como eliminación de acentos, signos de puntuación y *Stopwords*. También se pasan todos los carateres a minúsculas [11].
- **Tokenización:** es el proceso mediante el cual se divide un texto en sus distintos componentes, eliminando los espacios en blanco y saltos de línea dando lugar una cadena de *tokens*, que son cadenas de caracteres que tienen significado por sí mismo dentro del texto [11].
- ***stemming***: es el proceso mediante el cual se quitan y reemplazan sufijos de la raíz de una palabra [11]. Es una forma rápida, pero quizás menos efectiva de obtener el morfema de las palabras. En el caso de nuestro ejemplo se obtendría:
- **Lematización:** es otro proceso de obtención de raíces de las palabras. Es más complejo y completo que el proceso de *stemming*, ya que tiene en cuenta la morfología de las palbras para devolver la forma básica de la palabra [7], conocida como lema. Un lema es una serie de caracteres que conforman una unidad semántica y puede constituir una entrada de diccionario.

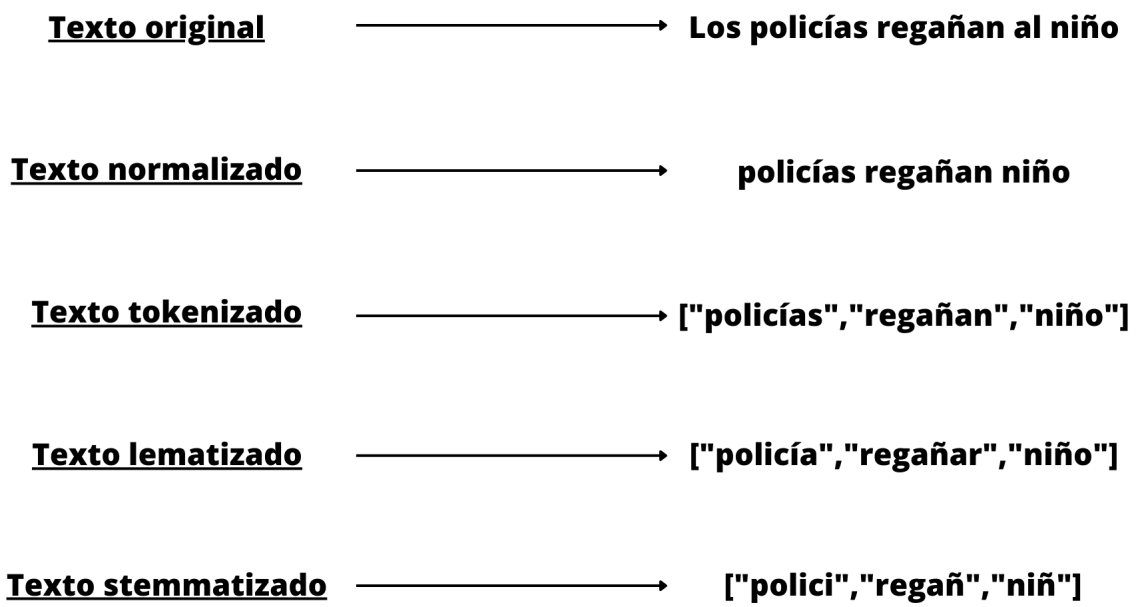


Figura 3.4: Ejemplo de preprocesado de texto

Cabe anotar que aplicar tanto *stemming* como lematización, son más complicados y costosos desde un punto de vista computacional en el idioma castellano debido a la complejidad del propio idioma, donde no se puede sacar de forma directa el lema en muchas ocasiones.

3.2.2 Reconocimiento de entidades

El Reconocimiento de Entidades (en inglés *Named Entity Recognition*, NER) es un proceso que se refiere a la tarea combinada de encontrar fragmentos de texto que constituyen un nombre propio y clasificar las entidades según su tipo (personas, lugares, organizaciones, etc)[56]. Algunos de estos tipos son:

Tipo	**Etiqueta**	**Categorías**
Personas	PER	Individuos, personajes ficticios, pequeños grupos
Organización	ORG	Compañías, agencias, partidos políticos
Localizaciones	LOC	Lugares físicos, montañas, lagos, ríos, mares, océanos
Entidades geopolíticas	GPE	Países, provincias, regiones
Instalaciones	FAC	Puentes, edificios, aeropuertos
Vehículos	VEH	Aviones, trenes, automóviles

Cuadro 3.1: Tipos genéricos de entidades

Tipo	**Etiqueta**	**Ejemplo**
Personas	PER	El problema es de *Daniel*.
Organización	ORG	La *OMS* lanzó una alerta.
Localizaciones	LOC	El *Monte Everest* es la montaña más alta
Entidades geopolíticas	GPE	En *Francia* van a subir los impuestos.
Instalaciones	FAC	El avión aterrizó al atardecer en el aeropuerto de *El Prat*.
Vehículos	VEH	El *Ford Focus* es un coche fiable.

Cuadro 3.2: Ejemplos de entidades

Existen varios problemas referentes a la identificación de nombres propios. El primer problemas es la propia identificación de las entidades. En castellano, inglés y otros idiomas similares que usan el alfabeto latino es sencillo identificarlas debido a que suelen empezar por una letra en mayúscula [53]. Sin embargo en otros idiomas como el árabe, chino o japonés, que son idiomas basados en símbolos, esta regla no se aplica. La regla también debe tener en cuenta el contexto, ya que una palabra al comienzo de una oración en castellano o inglés suele comenzar por una letra mayúscula.

Otro de los problemas a la hora de etiquetar una entidad es la ambigüedad de las palabras. En los sistemas NER pueden aparecer dos tipos de ambigüedad [56]:

- Ambigüedad en dos entidades distintas con el mismo nombre y que son del mismo tipo. Por ejemplo, *George Bush* puede hacer referencia a *George H. W. Bush* (41º presidente de EEUU) o a su hijo *George W. Bush* (43º presidente de EEUU). En ambos casos se hace una referencia a una persona (PER) pero son distintas entidades.
- Ambigüedad de dos entidades distintas con el mismo nombre y distinto tipo. Por ejemplo, *Washington* puede hacer referencia a un lugar (*Washington D.C., capital de EEUU*) o a una persona (*George Washington* primer presidente de EEUU).

Aunque existen soluciones más complejas, la solución más sencilla normalmente es tener en cuenta el contexto en el que aparece la entidad. Por ejemplo, en la frase ”Habrá atascos en todo el área de Washington”, la aparición de la expresión ”área de”nos permite identificar *Washington* como un lugar.

3.2.3 Análisis de sentimientos

El análisis de sentimientos es el campo de estudio en NLP en el cual se busca detectar opiniones favorables o desfavorables acerca de temas específicos en los textos. El sentimiento, en este campo, comprende la identificación de:

- **Expresiones de sentimientos.** Aquellas palabras (adjetivos, nombres, adverbios o verbos) que se usan para expresar sentimientos. Por ejemplo: amar, odiar gustar, horrible, precioso, bueno, magistralmente.
- **Polaridad y fuerza de las expresiones.** Cada una de las expresiones de sentimientos tienen una fuerza y una polaridad diferente. Se considera que hay tres tipos de polaridades: positiva, negativa y neutral; siendo positiva una opinión a favor del tema, negativa una opinión en contra del tema y neutral como la ”no opinión”[57]. Por ejemplo, ’amar’ es considerada una palabra más fuerte y positiva que ”gustar”.
- **Relación con el tema.** Por ejemplo, en la frase ”X venció a Y”, se puede ver la expresión ”venció”que tiene sentimiento positivo para X pero un sentimiento negativo para Y. [63]

Hay que tener en cuenta que estos puntos están relacionados entre sí.

Existen varios niveles de análisis respecto a la granularidad del texto:

- **Nivel de documento:** consiste en clasificar la opinión de todo un documento de texto. Esto no es aplicable a textos que tratan la opinión acerca de entidades diferentes. El nivel de documento es usado, por ejemplo, en campañas de marketing

para medir el impacto que tiene un producto analizando las reseñas y valoraciones que recibe. [57]

- **Nivel de frase:** consiste en determinar para cada frase del documento si la opinión es positiva, negativa o neutral.
- **Nivel de aspecto (*feature level*)**: los dos niveles anteriores no son capaces de discernir el objeto de la opinión, en este nivel lo que se busca es entender las relaciones con el tema. Por ejemplo, en la frase ”Fernando Alonso es un gran corredor a pesar de la mala fiabilidad de su vehículo”, se dice que el sentimiento es positivo hacia Fernando Alonso pero negativo hacia su vehículo, con lo cual clasificar la frase como positiva o negativa es muy complicado y sería necesario analizar el aspecto.

3.3 *Deep Learning* aplicado al NLP

Durante la última década, los algoritmos de NLP han sufrido una gran evolución. Inicialmente se usaban las llamadas *Bag of Words (BoW)*, en las cuales se creaban modelos de *Machine Learning* donde se capturaba la presencia y frecuencia de las palabras de un documento pero sin tener en cuenta la estructura gramatical o el orden en el que aparecían las palabras, pero con la aparición y expansión del Deep Learning aparecieron los *embeddings*, representaciones matriciales estáticas de texto en las cuales cada palabra del vocabulario se encuentra codificada como un vector [50].

Estos *embeddings* comenzaron a ser utilizados como entrada para las diferentes arquitecturas de redes neuronales artificiales que fueron apareciendo a lo largo de los años, siendo algunas de las arquitecturas más importantes:

3.3.1 Redes Neuronales Recurrentes (RNN)

Una Red Neuronal Recurrente (RNN, *Recurrent Neural Network*) es un tipo de red neuronal artificial que usa datos secuenciales. A diferencia de las Redes Neuronales tradicionales la información de las entradas anteriores y actual para determinar la salida actual, es decir, que la salida de una RNN es dependiente de los elementos anteriores de la secuencia [48]. Al trabajar con secuencias, este tipo de redes son buenas con el NLP ya que el lenguaje humano son letras y palabras puestas en orden para comunicar algo. Este orden de las palabras en una oración es muy importante para conservar su significado, por ejemplo no significa lo mismo ”le quité el papel al chicle”que ”le quité el chicle al papel”.

Existen distintos tipos de RNN, que se clasifican según la entrada y salida que se quiera obtener [83]:

- One-to-many: es una arquitectura que permite la entrada de un dato y la salida de una secuencia de datos. Esta red puede ser usada por ejemplo en el *image captioning* (descripción de imágenes en castellano) ya que se introduce la imagen y la red devuelve una secuencia de palabras como respuesta.

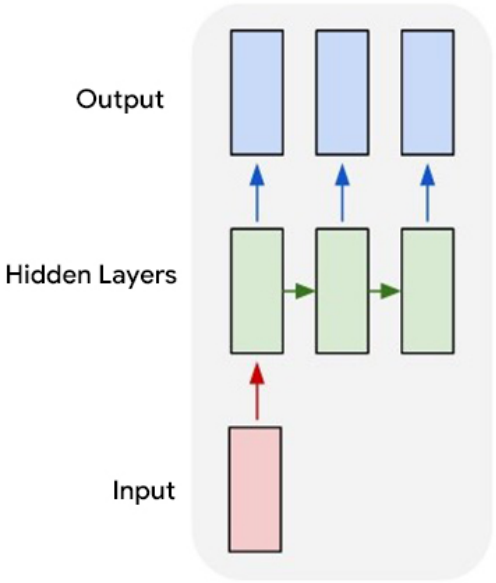


Figura 3.5: Estructura RNN one-to-many

- Many-to-one: esta arquitectura permite la entrada de varios datos para dar como salida un único dato. Un ejemplo de este tipo de redes podría ser un algoritmo de text to image (texto a imagen).

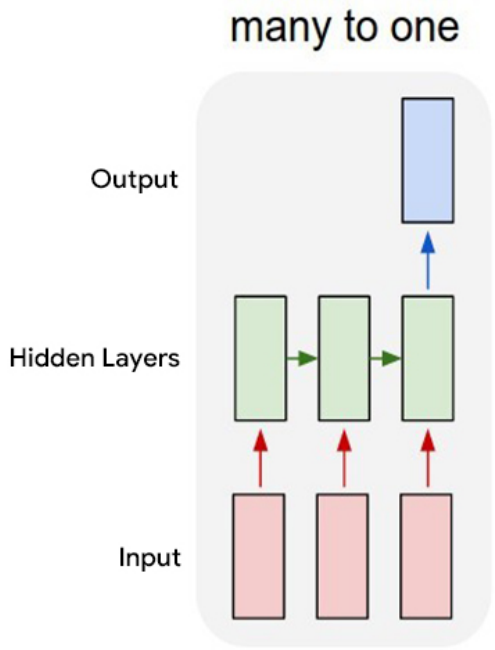


Figura 3.6: Estructura RNN many-to-one

- Many-to-many: la arquitectura recibe como entrada y da como salida múltiples datos. Un ejemplo de esto, son los *Chatbots* primitivos.

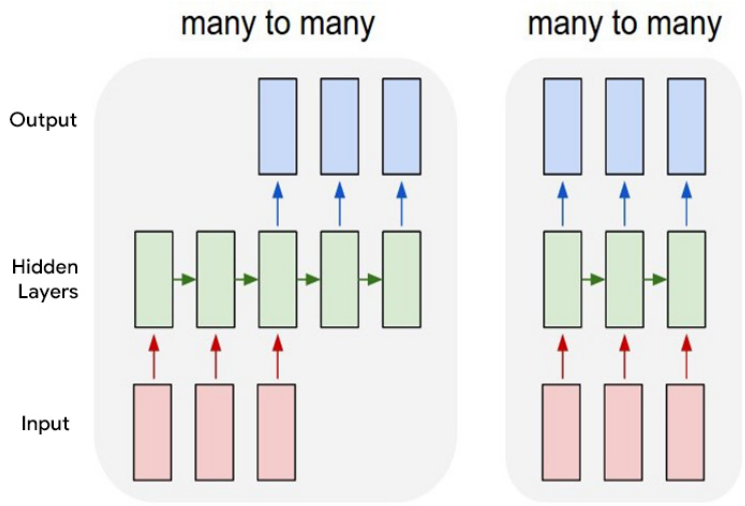


Figura 3.7: Estructura RNN many-to-many

3.3.2 Memoria Larga de Corto plazo (LSTM)

Las redes de memoria larga de corto plazo o en inglés *Long-Short Term Memory* (LSTM) son una variación de la arquitectura de las Redes Neuronales Recurrentes. Las RNN se encuentran limitadas a la hora de manejar dependencias de datos en un largo plazo en

las secuencias o como se le conoce el problema de la desvanecimiento del gradiente [78]. Esta solución fue planteada en el año 1997 por Sepp Hochreiter y Jürgen Schmidhuber en su paper ”Long Short-term Memory”[46].

A la arquitectura RNN se le añadió entonces una unidad de memoria llamada çelda LSTM”diseñada para recordar y olvidar información en función de la entrada recibida. Cada una de estas celdas se compone de tres compuertas que son tres pequeñas neuronas con función de activación Softmax:

- Puerta de entrada (*input gate*): qué información se va a agregar a la celda de memoria.
- Puerta de olvido (*forget gate*): qué información no se va a preservar (se va a olvidar) y no va a pasar a la celda de memoria.
- Puerta de salida (*output gate*): se encarga de preservar la celda de memoria y del estado oculto actual y que se envía al siguiente estado oculto.

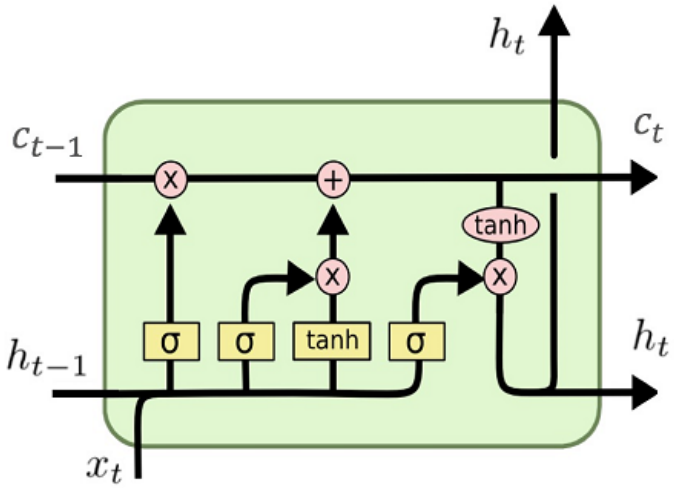


Figura 3.8: Arquitectura LSTM

3.3.3 Arquitectura de *transformers*

Presentados en el año 2017 por empleados de Google en el artículo titulado *Attention is All You Need* (Vaswani et al., 2017) [91], los *transformers* son una arquitectura de red neuronal muy utilizada actualmente para NLP y son el reemplazo y sucesor de las dos redes anteriormente mencionadas. Algunas de las ventajas y características que introdujo esta arquitectura son:

- Los mecanismos de atención: permiten dar contexto a las palabras que aparecen en una secuencia, independientemente del orden de procesado de la secuencia. La

función de atención permite valorar numéricamente las conexiones entre palabras. Por ejemplo, en la frase .El niño estaba cansado", el uso de estos mecanismos de atención permite a la red neuronal saber que el adjetivo çansado"tiene un valor de atención alto con "niño"pero no tan alto con .el".

- Mejora en la capacidad de paralelización en el entrenamiento: al no procesar secuencialmente las frases, se hace mucho más fácil paralelizar el proceso de entrenamiento, lo que permite que la red sea entrenada con conjuntos de datos más grandes.

La arquitectura *transformer* se basa en la estructura *encoder-decoder* (codificador-decodificador en castellano. En esta estructura el codificador traduce la entrada formada por la secuencia de símbolos y los transforma en una secuencia de valores continuos. Con esos valores continuos, el decodificador genera una secuencia de salida de símbolos. Esto se repite iterativamente, donde el modelo consume los símbolos anteriores como entrada adicional a medida que se va generando el texto. Esta estructura queda descrita en la figura 3.9.

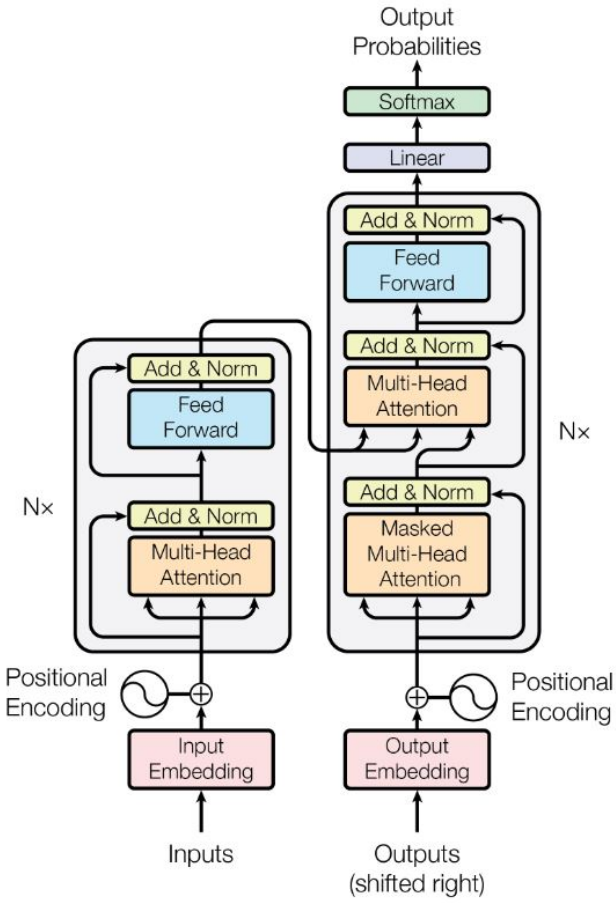


Figura 3.9: Arquitectura *transformer* [91]

Mecanismos de atención

La clave del buen funcionamiento de la arquitectura son los mecanismos de atención. Estos mecanismos de atención ayudan a crear y dar un nivel de fuerza de forma vectorial a las relaciones de una palabra con el resto.

En la práctica existen tres maneras diferentes de aplicar estos mecanismos de atención a una red neuronal *transformer* [96]:

- Atención codificador-decodificador: es un mecanismo de atención que permite al decodificador atender a la secuencia de entrada cuando genera la secuencia de salida.
- Auto-atención del codificador: permite al codificador atender a todas las partes de la salida del codificador anterior.
- Auto-atención del decodificador: permite al decodificador atender a todas las partes de la secuencia que hay en el decodificador.

Para calcular la atención se usan los vectores de atención que pueden definirse como el mapeo de una consulta y un set de pares clave-valor para una salida (de aquí en adelante *Query-Key-Value*) o *soft dictionary*. Se muestra como salida la suma ponderada que corresponde a las claves que son más similares a la consulta permitiendo a la red a centrarse en un subconjunto de su vector de entrada. Estos vectores se pueden definir formalmente como:

- Matriz Q (*Query*): donde se guardan los tokens o, a efectos prácticos, su correspondiente *embedding*.
- Matriz K (*Key*): donde se guardan los tokens o *embeddings* como claves de un diccionario.
- Matriz V (*Value*): donde se guardan los embeddings de salida.

A mayores se dispone de d_k que es el tamaño de las claves de atención y se elige mediante hiper-parámetros elegidos a la hora de diseñar la arquitectura. Con esto y aplicando una función Softmax aparece la fórmula:

$$Attention(Q, K, V) = softmax(\frac{QK^t}{\sqrt{d_k}})V \tag{3.1}$$

Parte III

Desarrollo del Sistema

Capítulo 4

Fuentes de datos

Para las fuentes de datos de noticias de la web, se pedían inicialmente al menos cuatro fuentes diferentes lo más heterogéneas posibles, buscando que las fuentes presentasen perfiles ideológicos variados y diferentes niveles de fiabilidad basada en el historial y trayectoria de los medios analizados. Finalmente se escogieron 6 fuentes para aumentar la cantidad de datos disponibles.

4.1 Agencia EFE

La Agencia EFE [32] es una agencia de noticias internacional con sede en Madrid, España, y que cuenta con una dilatada trayectoria periodística desde su fundación en 1939. Es la primera agencia de noticias en español y la cuarta del mundo.

EFE ofrece información sobre muchos temas incluyendo política, economía, cultura y deportes. La cobertura del medio es muy amplia, llegando a cubrir información de más de 120 países. Al año se estima que la agencia distribuye casi tres millones de noticias, más de 8200 noticias diarias de media). [73]

Además, la Agencia EFE cuenta con una de las mayores bases de datos históricas de información periodística en castellano, con más de 13 millones y medio de noticias entre el 1 de enero de 1988 y el 24 de marzo de 2023, que podría ser utilizada también para ampliar el *corpus* de noticias que se va a extraer [31].

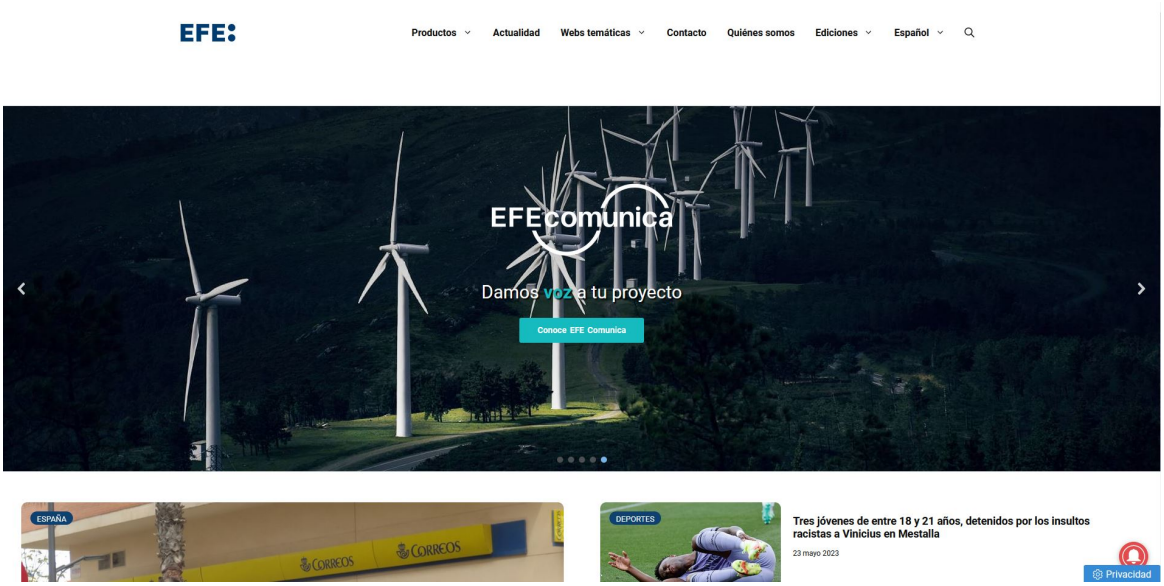


Figura 4.1: Interfaz web Agencia EFE (23/05/2023)

4.2 Europa Press

Europa Press [77] es una agencia de noticias española con sede en Madrid, España. Fue fundada en 1953 como una empresa que elaboraba y distribuía material editorial como libros y folletos hasta que en 1966, con la puesta en marcha de la Ley de Prensa de 1966 [37] que permitía la creación de nuevas agencias de noticias, se puso en marcha el servicio de noticias. Compite directamente con la Agencia EFE (4.1) y Colpisa (4.3), siendo la segunda agencia de información con mayor facturación en el año 2020[85].

Europa Press dispone de corresponsalías en todas las comunidades autónomas de España. Actualmente, publica de media unos tres mil artículos diarios [5].

Figura 4.2: Interfaz web Europa Press (23/05/2023)

4.3 Colpisa

Colpisa [16] es una agencia de información de españa, fundada en 1972 como una iniciativa de diversos periódicos a nivel regional para centralizar la información y facilitar su acceso. Desde enero de 2007, la agencia comenzó a formar parte del grupo Vocento, pasando a ser la redacción central de dicho grupo. En 2010 se transformó en la plataforma de contenidos multimedia que es actualmente. [15]

El grupo Vocento, es el grupo que dirige el periódico ABC (de línea editorial conservadora [12]) y algunos de los periódicos de tirada regional más grandes como El Correo, El Norte de Castilla y El comercio [92]. Esto último hace que a nivel de información regional, junto con Europa Press (4.2), la agencia Colpisa sea una de las más destacadas.

Figura 4.3: Interfaz web Colpisa (23/05/2023)

4.4 VisionTimes

VisionTimes o como es conocido en chino Kanzhongguo es un noticiero fundado en 2001 con la intención de proveer de información a la comunidad china fuera de China.[88] La línea editorial de este periódico se asocia con la religión conocida como Falun Gong, conocida por su gran enemistad con el gobierno chino, por lo que algunas noticias relacionadas con China contienen un fuerte sentimiento anti-gobierno.[69]

Este periódico ha sido vinculado con la difusión de noticias tanto falsas como a favor de ideologías de extrema derecha, incluyendo campañas de desinformación y uso de Deepfake.[4][2]

Dispone de versión online en varios idiomas, incluyendo el español [87], que es el idioma del cual se han extraído noticias de este noticiero.

Figura 4.4: Interfaz web VisionTimes (23/05/2023)

4.5 Outono

Outono (Contando Estrelas) [33] es un blog personal creado en 2004 y publicado desde España. Su autor, Elentir, define la línea editorial del blog como católica, democrática y liberal-conservadora, además de independiente[34]. Al ser un blog personal, las noticias que se publican en esta web son subjetivas y se encuentran altamente sesgadas por la ideología del autor.

Figura 4.5: Interfaz web Outono (23/05/2023)

4.6 ElMundoToday

ElMundoToday [36] es un periódico satírico publicado online. Fue creado en 2009 por Kike García y Xavi Puig [35]. La página utiliza el estilo de la prensa tradicional online, con la salvedad de que el titular y el contenido de la noticia es ficticio, humorístico y satírico.

En un primer vistazo las noticias podrían parecer verídicas, lo que ha llevado en ocasiones a confusiones por parte de usuarios de Internet y periodistas que han dado por ciertas sus noticias. El caso más sonado fue el que surgió a partir del titular de ElMundoToday que decía "*La infanta Elena pide que la imputen como a su hermana*", donde el canal de colombiando de noticias NTN24 dio por buena la información y llegaron incluso a conectar en directo con el corresponsal en Madrid para comentar la noticia.[62]

Figura 4.6: Interfaz web ElMundoToday (23/05/2023)

Capítulo 5

Análisis

5.1 Descripción del sistema

El proyecto consiste en la creación de una aplicación de extracción y análisis de noticias de diferentes fuentes. Se puede dividir el proyecto en cuatro partes:

- **Extracción de noticias**: mediante el uso de *Web Scrapers* y *Web Crawlers* (3.1) se extraen las noticias de manera automática de las fuentes de datos que se seleccionen (descritos en 4). La información extraída son las partes relevantes de una noticia en la web, según están detalladas en 5.1.1.
- **Bases de Datos**: una vez extraídos los datos de las diferentes webs de noticias, se almacena la infrormación extraída en una base de datos. Dicha base de datos puede incluir información adicional de la noticia como enlace original, tema, fuente y etiquetas de la noticia.
- **Análisis mediante técnicas de Procesamiento del Lenguaje Natural**: sobre las noticias extraídas, se propone realizar un análisis inicial del cuerpo de las noticias, a fin de identificar el sentimiento predominante mediante análisis de sentimientos (3.2.3) y la identificación de las entidades implicadas en la noticia (3.2.2).
- **Plataforma web**: finalmente para mostrar los resultados obtenidos, se ha propuesto la creación de una plataforma web sencilla que permita filtrar las noticias y obtener su contenido y el resultado del análisis de dicho contenido.

5.1.1 Noticias

En este Trabajo Fin de Grado, se va a trabajar sobre las noticias que se extraigan de la web. Dichas noticias van a estar compuestas por:

- Fuente.
- Titular.
- Contenido de la noticia.
- Fecha de publicación.
- Tema o tópico principal al que pertenece.
- Autor, si lo hubiera, ya que en el caso de agencias de noticias no suelen aparecer los autores.
- Etiquetas, si las hubiera, para señalar los tópicos o temas secundarios.

5.2 Requisitos

A partir de los objetivos planteados en el capítulo 1 y las partes planteadas en 5.1, se han obtenido los siguientes requisitos funcionales y no funcionales para cada una de las partes planteadas. Estos requisitos son las condiciones básicas que debe cumplir cada una de las partes del sistema, por lo que es posible que no se incluya toda la funcionalidad o funciones adicionales que posteriormente hayan sido implementadas.

5.2.1 *Web Scraping* y *Web Crawling*

Requisitos funcionales

- **RF101**: El sistema debe entrar y navegar por las páginas webs y blogs de noticias.
- **RF102**: El sistema debe guardar la información recogida en una base de datos.
- **RF103-1**: El sistema debe extraer y guardar el titular de la noticia.
- **RF103-2**: El sistema debe extraer y guardar el cuerpo de la noticia.
- **RF103-3**: El sistema debe extraer y guardar la fecha de publicación de la noticia.
- **RF103-4**: El sistema debe extraer y guardar la fecha de recogida de la noticia.
- **RF103-5**: El sistema debe extraer y guardar el enlace a la noticia original.

- **RF103-6**: El sistema debe extraer y guardar el tópico o tema principal de la noticia.
- **RF103-7**: El sistema debe extraer y guardar la fuente de cada noticia.

Requisitos no funcionales

- **RNF101**: El sistema debe implementarse en `Python` versión 3.
- **RNF102**: El proceso de *Web Scraping* de noticias debe realizarse períodicamente, en un lapso de tiempo de entre 1 día y 1 semana.
- **RNF103**: Se deben extraer noticias de al menos 4 fuentes distintas, con diferentes sesgos.
- **RNF104**: El idioma de las fuentes deben ser el castellano.
- **RNF105**: El sistema debe ser capaz de adaptarse con facilidad a los cambios en la estructura de los sitios webs de noticias.
- **RNF106**: El sistema debe ser capaz de manejar errores y excepciones que puedan aparecer en el proceso de *Scraping Web* y guardado de la noticia, tales como errores de conexión o errores de sintaxis HTML.
- **RNF107**: El sistema almacenará los datos extraídos en una base de datos `PostgreSQL`.

5.2.2 Plataforma web

Requisitos funcionales

- **RF201**: El sistema debe mostrar la lista de noticias.
- **RF202-1**: Las usuarios deben poder filtrar las noticias por fuente de origen.
- **RF202-2**: Los usuarios deben poder filtrar las noticias por palabras clave.
- **RF202-3**: Los usuarios deben poder ordenar las noticias según fecha de extracción, fecha de publicación y orden alfabético.
- **RF203**: El sistema debe mostrar el contenido de la noticia, incluyendo titular, cuerpo, autor, fecha de publicación, enlace y tema de la noticia.
- **RF204**: El sistema debe realizar un análisis de sentimientos sobre el cuerpo de la noticia y mostrar el resultado.
- **RF205**: El sistema debe realizar un análisis de entidades sobre el cuerpo de la noticia y mostrar el resultado.

Requisitos no funcionales

- **RNF201**: El sistema debe proveer de un diseño sencillo de utilizar y aprender.
- **RNF202**: El sistema debe ser eficiente.
- **RNF203**: El tiempo de respuesta del sistema debe ser inferior a 3 segundos.
- **RNF204**: El sistema debe ser accesible desde los navegadores `Firefox` y `Chrome`.
- **RNF205**: El sistema debe emplear `Python` versión 3 para la parte de análisis de sentimientos y de entidades.
- **RNF206**: El sistema debe emplear para el *backend* algún *framework* web basado en `Python` como `Flask` o `Django`.
- **RNF207**: El sistema debe emplear para el frontend algún *framework* web como `Angular`, `Bootstrap` o `Vue.js`.
- **RNF208**: El sistema debe usar la base de datos `PostgreSQL`, definida y poblada en el apartado anterior.

5.3 Casos de uso

Para el uso de la plataforma web se han planteado los siguientes casos de uso para el usuario:

- **Visualizar noticia**: el usuario puede seleccionar una noticia para ver su contenido completo.
- **Visualizar análisis de noticia**: al visualizar la noticia, también se lanza el módulo de análisis lo que muestra las entidades y los sentimientos del contenido.
- **Filtrar noticias**: el usuario puede filtrar las noticias que se muestran por palabra clave y fuente y cambiar el orden en el que se muestra.

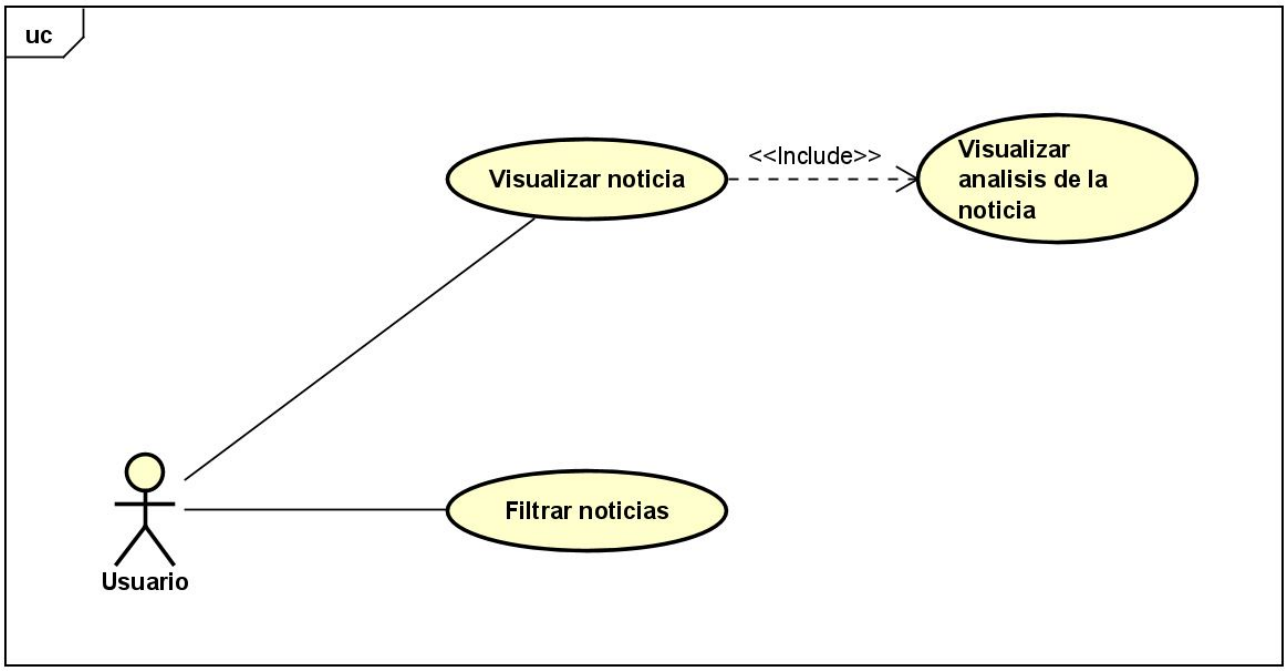


Figura 5.1: Diagrama de casos de uso

Capítulo 6

Diseño

6.1 Diseño de *Web Scrapers*

Para la creación de los *scrapers*, se ha decidido usar el patrón de diseño *Template* que es una arquitectura en la cual se define el esqueleto de un algoritmo en un método y permite que las subclases redefinan ciertas etapas del algoritmo sin cambiar su estructura general. La razón principal para los *scrapers* que se han planteado, todos siguen los mismos pasos a la hora de extraer el contenido de las noticias, lo cual podría llevar a duplicar código y esto se evita con el uso de este patrón.

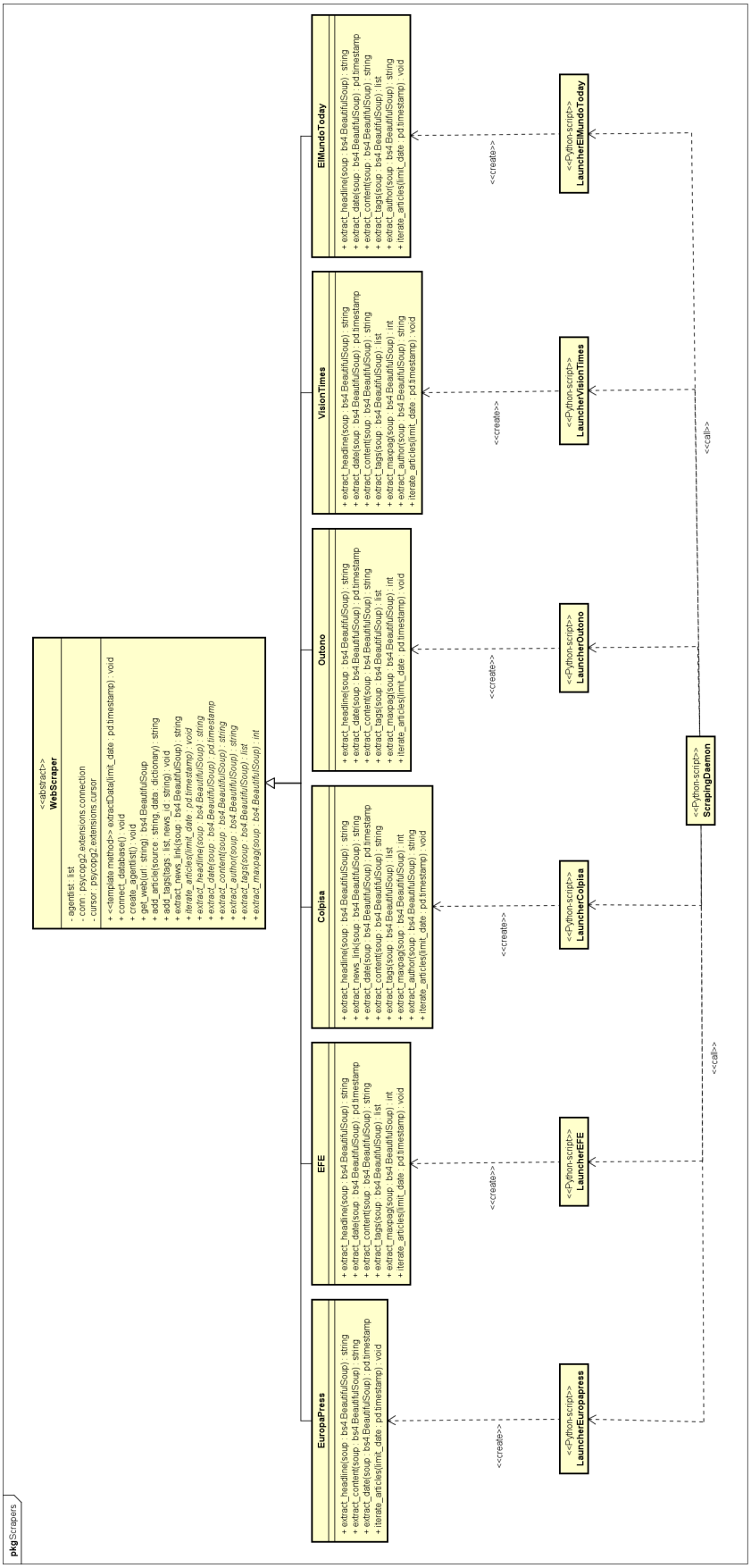


Figura 6.1: Diseño de los Web Scrapers (Patrón Template)

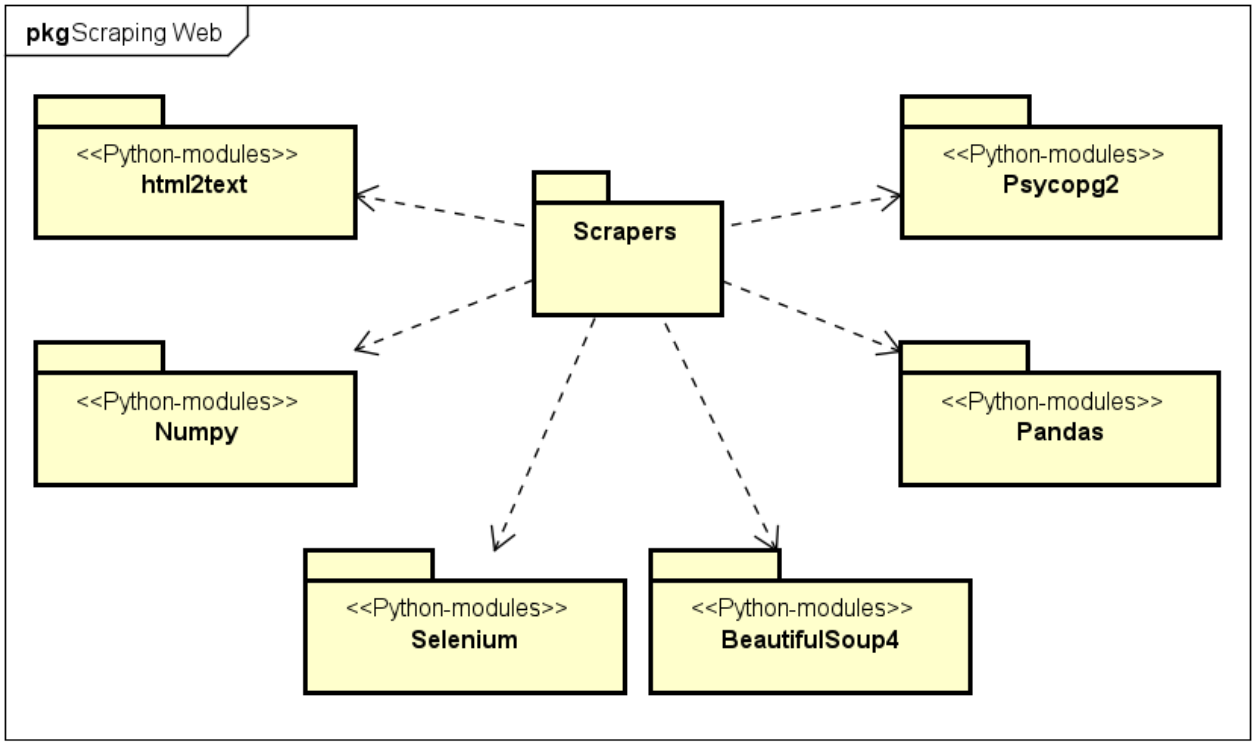


Figura 6.2: Dependencias paquetes de Python

6.2 Diseño de bases de datos

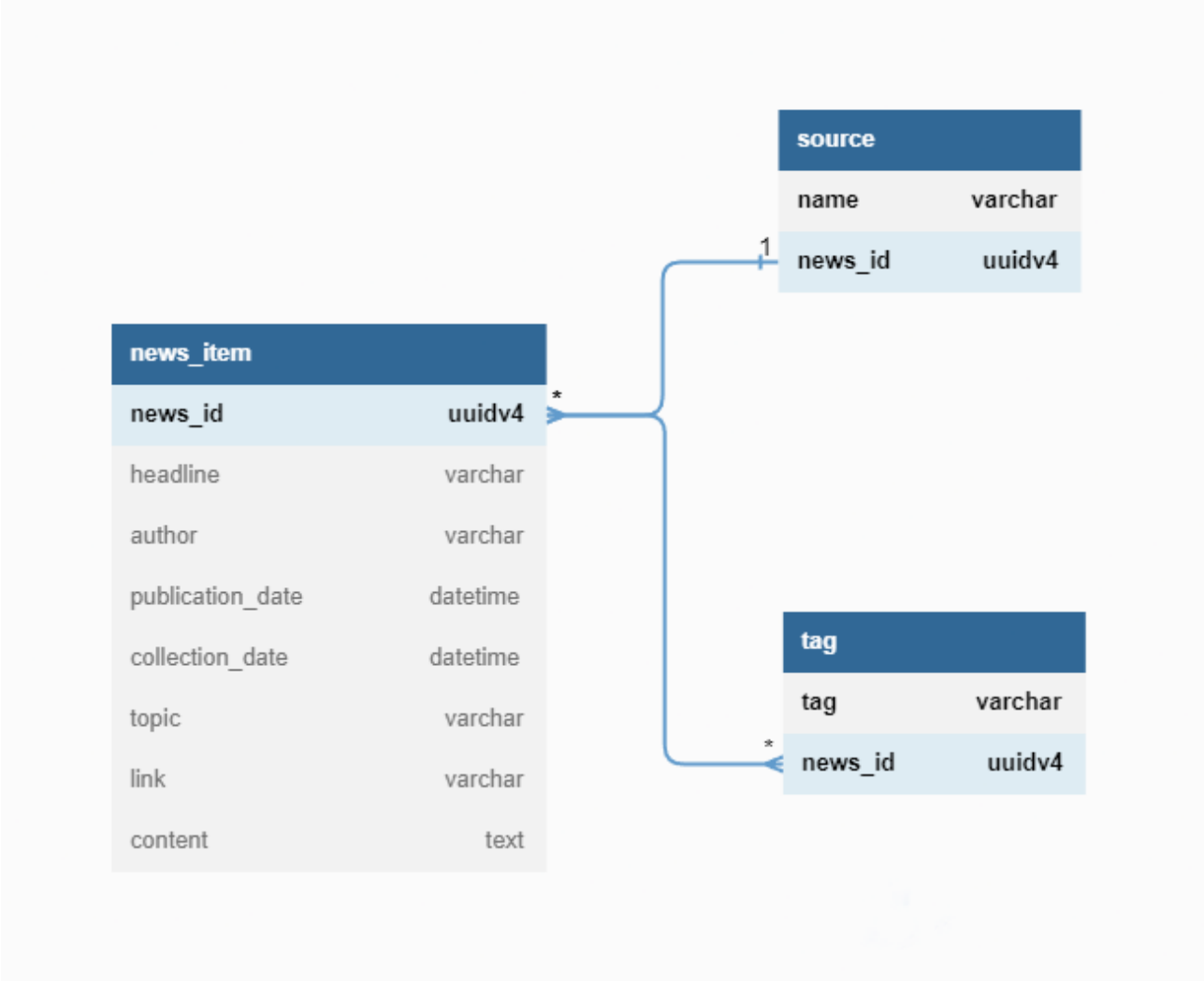


Figura 6.3: Esquema de la base de datos

6.3 Diseño de la plataforma web

A la hora de diseñar la interfaz de la plataforma web lo que se buscaba era que cumpliera con los siguientes principios y condiciones, a mayores de los requisitos planteados en el apartado 5.2.2:

- **Facilidad de uso**: la interfaz debía ser intuitiva y fácil de usar, incluso para usuarios inexpertos.
- **Consistente**: el diseño debía ser consistente.

- **Estética**: la interfaz debía ser agradable a visualmente y de aspecto profesional por lo que debía evitar colores chillones o combinaciones extrañas.
- **Claridad**: el usuario debía tener en todo momento claro la información que estaba recibiendo y cómo podía interactuar con la web.

6.3.1 Mockups versión 1

Landing page

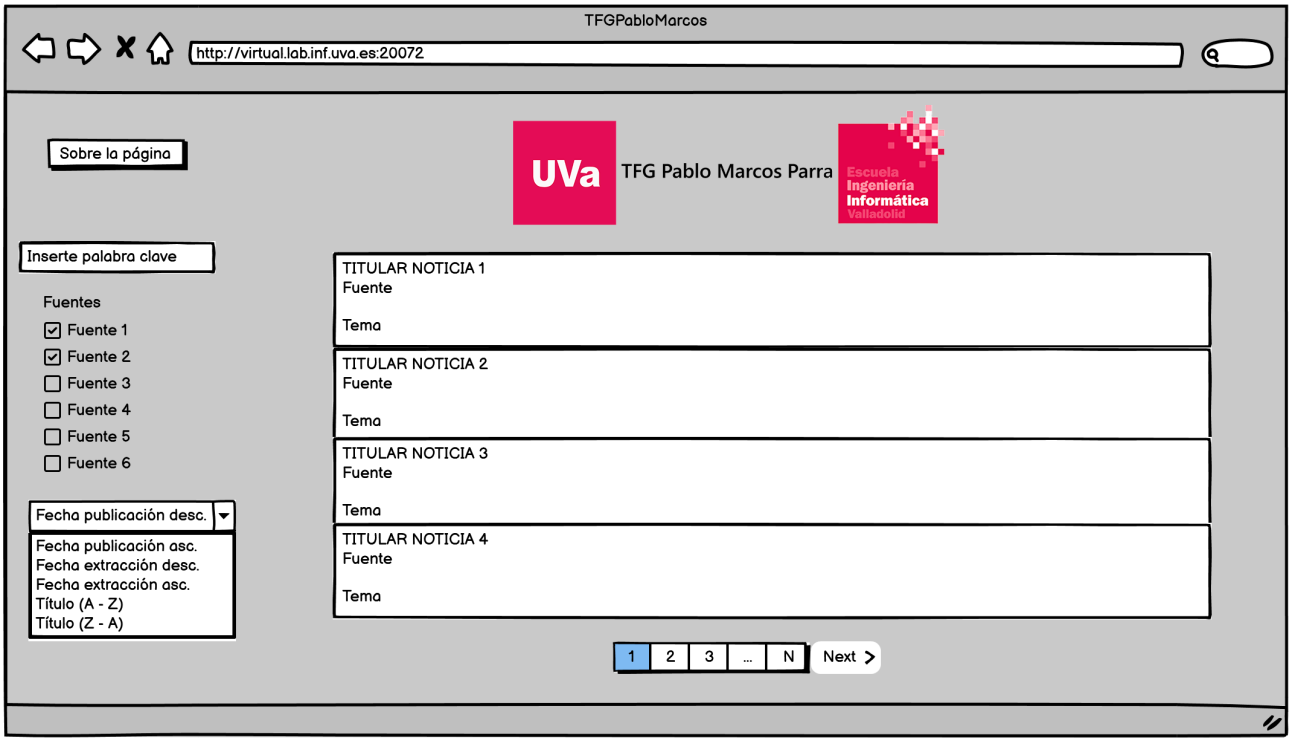


Figura 6.4: Versión 1. *Landing page*

Se diseñaron los *mockups* de una primera versión tentativa de lo que iba a ser la plataforma web. Las partes principales de la interfaz ya aparecen en esta primera versión. Tanto los filtros, como la paginación, como el botón ”Sobre la página”.

Visualizar noticia

En esta primera versión, se planteó el poner un botón para que fuera el usuario el que lanzase el análisis de la noticia. Al hacerlo, se mostraría en esa ventana en el lateral dos salidas, la primera un gráfico con la distribución de las frases de la noticia según su sentimiento y la segunda una enumeración de las entidades y sus tipos.

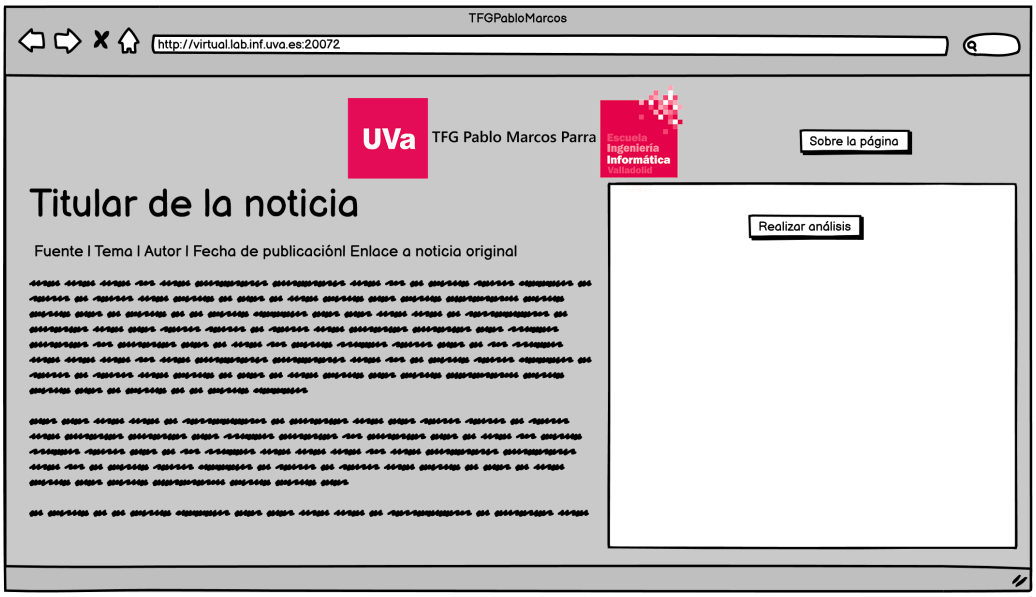


Figura 6.5: Versión 1. Visualizar noticias

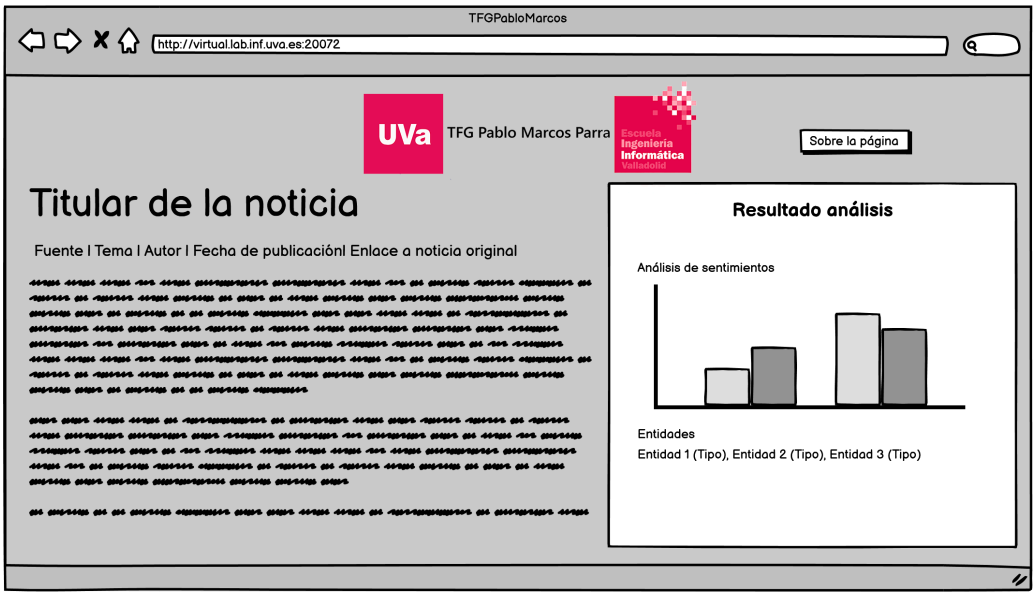


Figura 6.6: Versión 1. Resultado final análisis

6.3.2 Mockups versión 2

En esta segunda versión se buscaba mejorar los aspectos que en la versión anterior parecían más débiles, como por ejemplo la visualización de los resultados obtenidos.

Landing page

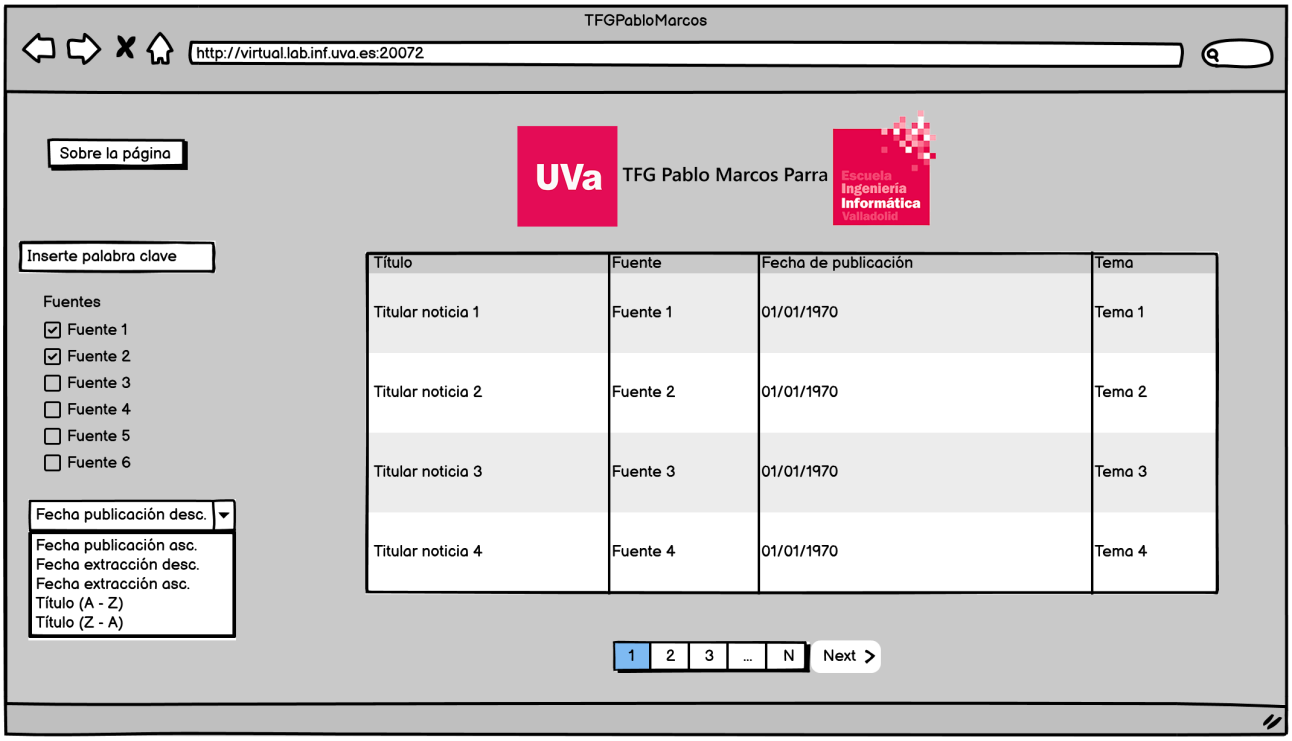


Figura 6.7: Versión 2. *Landing page*

En esta versión la única diferencia respecto a la versión anterior es la manera en la que se muestran las noticias. En este caso, se optó por usar una tabla formato *dataframe* en el cual se mostraban claramente todos los parámetros que podían resultar interesantes para el usuario.

Visualizar noticia

En la segunda versión se tomó otra aproximación a la manera de visualizar los resultados del análisis. En la versión 1 gran parte del espacio no era aprovechado, por lo que se decidió que en el centro se mostraría lo principal que es la noticia y en los laterales los resultados.

En el cuerpo de la noticia es donde se aplicó esta vez el análisis de sentimientos ya que en vez de mostrar el resultado en una gráfica abstracta que quizás no permitía interpretar claramente el resultado. En este caso se decidió que ya que se analizaba frase a frase el sentimiento, se visualizaría mejor si se señalaba este sentimiento en el propio texto. La leyenda de colores junto con el sentimiento predominante del texto se mostraban en el lado izquierdo de la pantalla. En el lado derecho se mostraba una tabla con las entidades y su tipo. A mayores se añadió un botón ”Volver”que permitiera al usuario volver a la página anterior fácilmente.

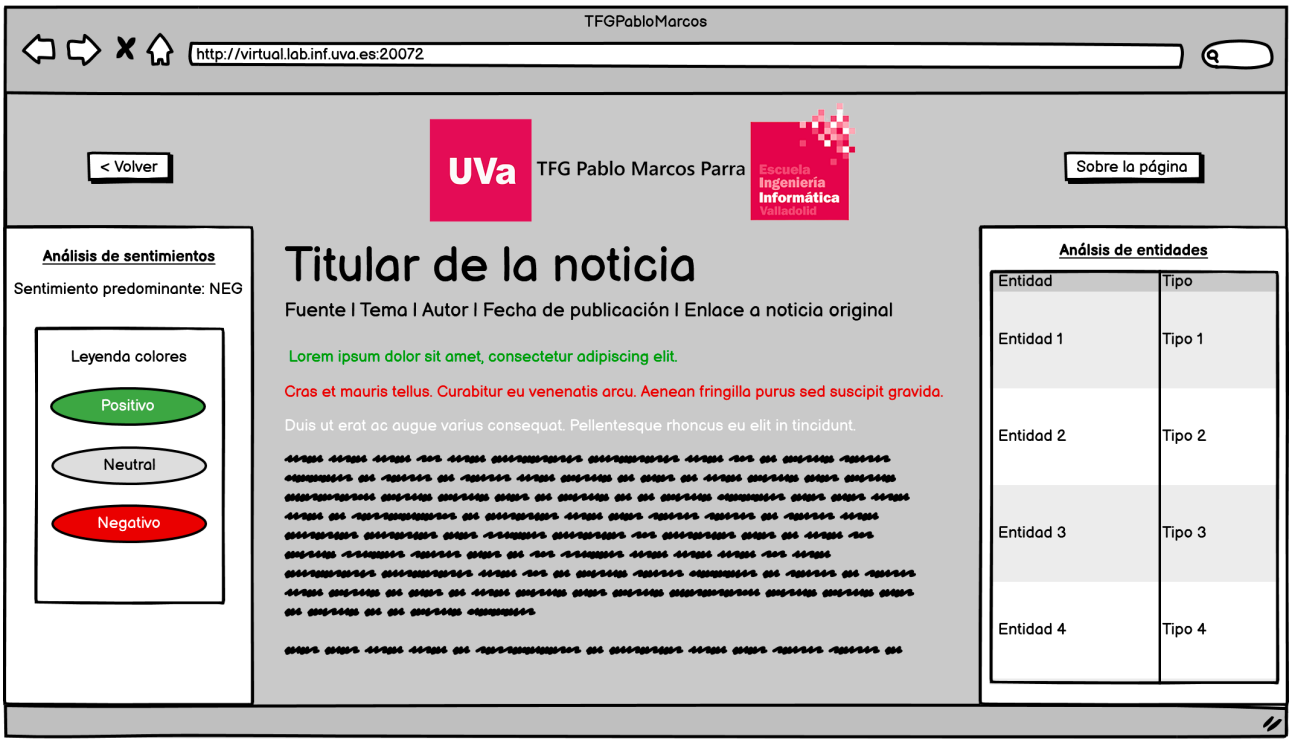


Figura 6.8: Versión 2. Visualizar noticias y resultados del análisis

6.3.3 Resultado final

A continuación se mostrarán las figuras con la implementación de la interfaz que finalmente se ha implementado.

Landing Page

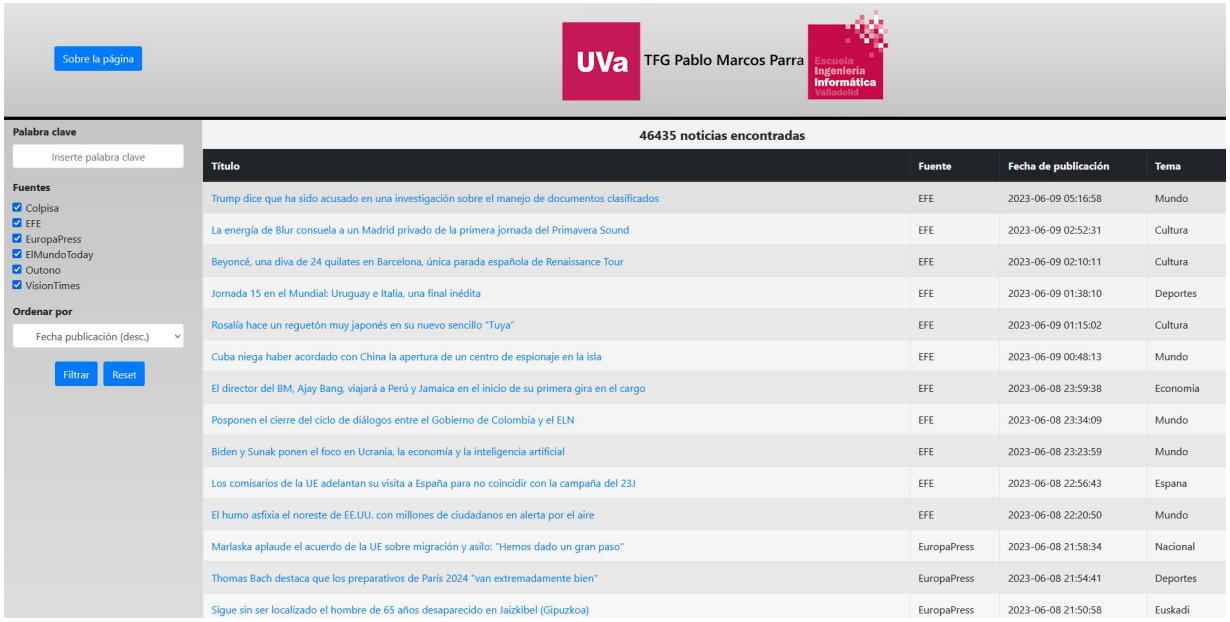


Figura 6.9: Interfaz final. *Landing Page*

Página de noticia

Figura 6.10: Interfaz final. Visualizar noticia y resultados de análisis

Página ”Sobre la página”

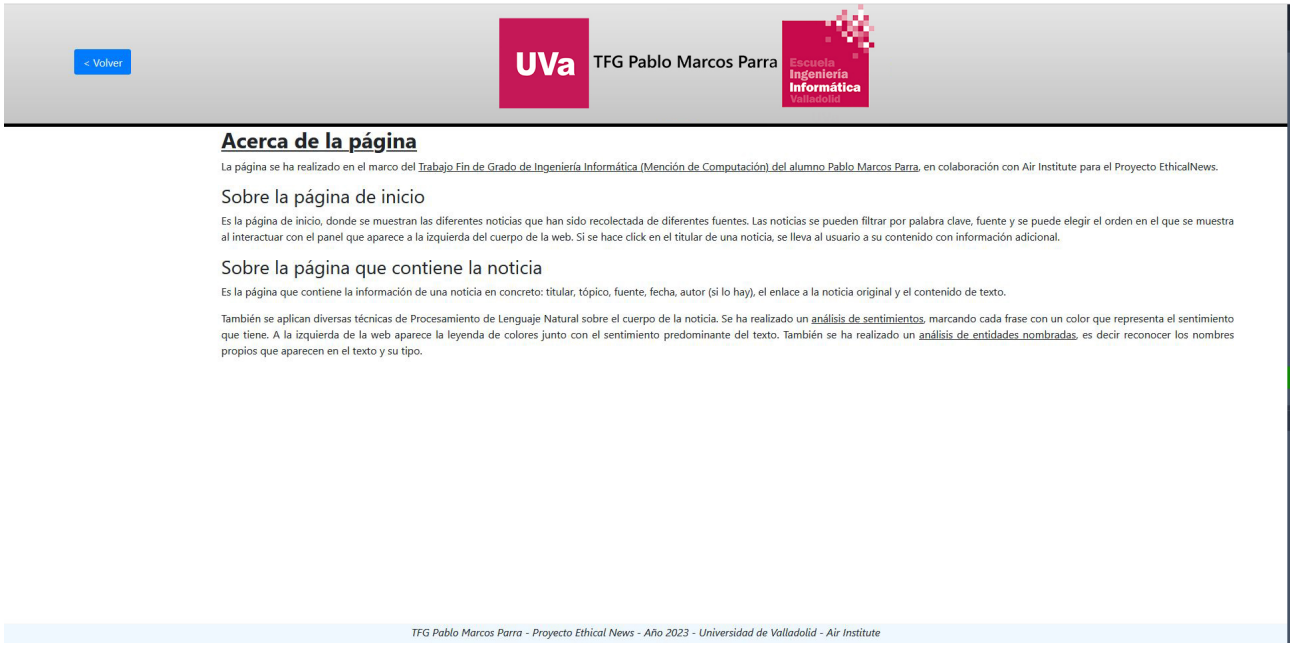


Figura 6.11: Interfaz final. Página ”Sobre la página”

6.4 Arquitectura despliegue

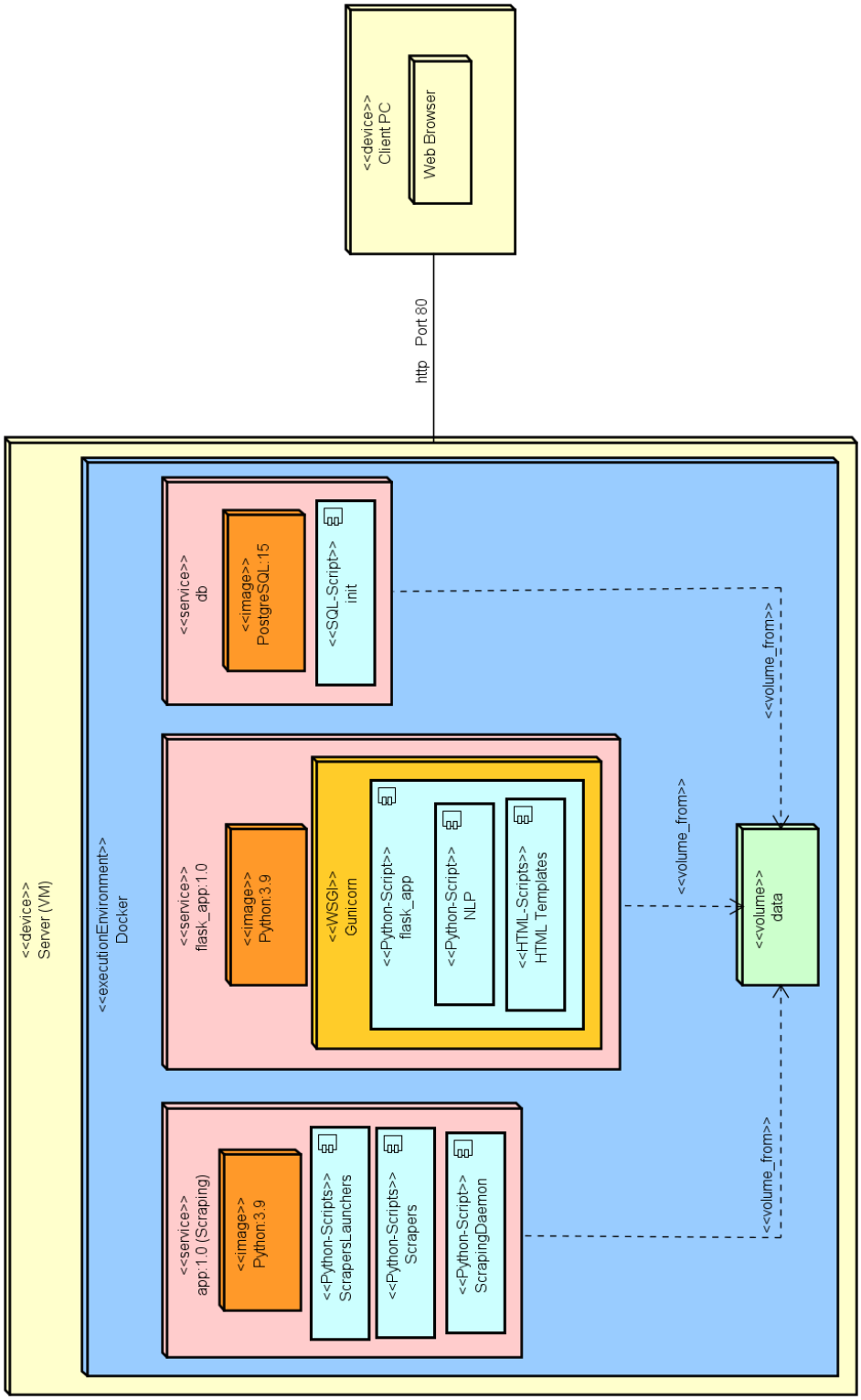


Figura 6.12: Diagrama de despliegue

Capítulo 7

Implementación

7.1 Herramientas de Desarrollo

En esta sección se estudiarán las herramientas de desarrollo software de que en han usado a lo largo del proyecto.

7.1.1 Entornos de desarrollo utilizados

Pycharm

PyCharm es un entorno de desarrollo integrado (IDE) utilizado para programar en el lenguaje `Python`. Desarrollado por la compañía JetBrains desde el año 2010. Para este proyecto se ha usado la versión Community (gratuita y de código abierto).[54]

Algunas de las razones por las que se ha usado Pycharm incluyen la facilidad de uso y ejecución de programas `Python`, el control de versiones y su integración con otros sistemas y que incluye soporte para *framework*s populares de `Python`, como el que se ha usado en este proyecto (`Flask`).

Visual Studio Code

Visual Studio Code es un editor de código fuente (IDE) desarrollado por Microsoft desde 2015. Es una herramienta de código abierto y gratuita que cuenta con una amplia variedad de funciones para el desarrollo de software.[59]

En el caso del proyecto, Visual Studio Code ha sido usado principalmente para la

edición de los HTMLs de la plataforma web debido a que cuenta con una gran cantidad de extensiones y complementos que permiten un uso más ligero y rápido.

Jupyter Notebook

Jupyter Notebook es una aplicación web de código abierto creada en 2014 por Jupyter [55] que permite crear cuadernos (notebooks) `Python` interactivos que contienen código fuente, visualizaciones y texto enriquecido (Markdown).

Los *notebooks* de Jupyter están compuestos por celdas que pueden contener código, Markdown y otros elementos multimedia como por ejemplo imágenes. El código en las celdas se puede ejecutar directamente en el navegador web.

Se ha usado este entorno en concreto al realizar pruebas de funcionamento de los *Web Scrapers*, ya que una de las ventajas de Jupyter Notebook es que permite una rápida visualización de resultados según se van ejecutando las celdas de código sin necesidad de ejecutar el código completo, lo que ha hecho que la depuración de código sea mucho más ligera y rápida.

7.1.2 Herramientas de gestión de versiones

Git

Git es un sistema de control de versiones de código abierto, diseñado para la gestión y manejo de proyectos software de todos los tamaños. Fue creado por Linus Torvalds en 2005 para el desarrollo del kernel de Linux.

Un sistema de control de versiones permite a varios desarrolladores trabajar en el mismo proyecto de software al mismo tiempo, sin problemas de conflictos de versiones al mantener un historial detallado de los cambios realizados en el código fuente lo que permite encontrar errores, revertir cambios y acceder a las diferentes versiones existentes del código.[41]

Git funciona utilizando una estructura de árbol, donde cada nodo representa una versión del código, y hay diferentes ramas donde se van almacenando esos nodos. Estas ramas se pueden fusionar con la rama principal o ”master”. La ventaja de Git es que es un sistema distribuido lo que significa que cada desarrollador dispone de su propia copia local del repositorio, permitiéndole trabajar sin conexión y facilitando la colaboración con sus compañeros.

GitHub

GitHub es una plataforma de desarrollo colaborativo de software basada en la nube que se utiliza para alojar y compartir proyectos de software. Fue lanzado en 2007, pero desde 2018 es gestionado por Microsoft. [42]

GitHub utiliza el sistema de control de versiones Git (ver 7.1.2).

En el caso del proyecto, se ha usado Git para el control de versiones y GitHub para alojar el contenido del proyecto.

7.1.3 Herramientas de diseño

Astah* Professional

Astah* Professional es una herramienta de modelado de software usada para crear diagramas UML (Lenguaje de Modelado Unificado) y otros tipos de diagramas que han sido utilizados para la parte de diseño de software. Es un programa desarrollado por Change Vision desde 2005.[6]

Astah* Professional se ha usado para visualizar y diseñar la arquitectura de software, la lógica de procesos y la estructura de datos de el sistema de software planteado, incluyendo diagramas de clases, diagramas de casos de uso, diagramas de secuencia, diagramas de actividades y diagramas de estado.

La licencia utilizada ha sido la facilitada por la facultad de Ingeniería Informática de Valladolid.

Balsamiq Wireframes

Balsamiq Wireframes es una herramienta de diseño y creación de interfaces que permite crear *mockups* de manera simple y fácil, gracias a su interfaz de usuario que permite arrastrar y soltar los componentes de tu boceto en la pantalla. Fue creado por Balsamiq en el año 2008 [8].

Cuenta con una gran variedad de plantillas y símbolos predefinido que permite realizar un prototipado rápido. Una de las características interesantes de Balsamiq Wireframes es la apariencia (*low-fidelity), baja fidelidad* que da una sensación de dibujo a mano y permite que el creador se centre en estructura y contenido más que en aspecto [9].

7.1.4 Base de datos

PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos relacional de código abierto, que usa y extiende el lenguaje SQL combinado con otras características que permiten almacenar datos de manera segura y fácilmente escalable. PostgreSQL fue creado en el año 1986 en la Universidad de California Berkeley.[76]

Este sistema fue elegido por parte de la empresa para facilitar su integración con el resto de los artefactos del proyecto.

7.2 Análisis de librerías y *frameworks*

Una parte muy importante en el desarrollo de este proyecto ha sido el estudio y análisis de las differentes alternativas que se han planteado para la realización de las diferentes partes del proyecto. En esta sección, para cada una de las librerías y herramientas, se estudiarán sus características y finalmente se compararán y justificarán las razones por las que han sido empleadas o no en el proyecto.

7.2.1 Opciones para implementación de *Web Scraping* con `Python`

Librería de `Python Requests`

La librería `Request` es la librería más estándar y utilizada para la realización de solicitudes HTTP en `Python`.[23]. La librería ejerce como API para facilitar que el usuario mande solicitudes *HTTP/1.1* de manera muy sencilla, sin necesidad de grandes conocimientos del funcionamiento de dicho protocolo.

La librería se compone de dos métodos principales *GET* y *POST*, aunque existen otros métodos como *PUT*,*DELETE*, *HEAD*, *PATCH* y *OPTIONS*. En el caso de un *Web Scraper* el método que se va a usar es el método *GET*.

GET es la función más utilizada. El método *GET* manda una solicitud HTTP *GET* mediante la cual se va a intentar conseguir o recuperar datos a partir de una fuente específica HTTP. Un ejemplo sencillo de esta función sería:

```
import request
r=request.get("https://api.github.com/events")
```

Esta llamada devuelve un objeto *Response* llamado r que contiene la información

decodificada automáticamente del servidor. Es un diccionario JSON que contiene información como la cabecera HTTP, el texto o la codificación (UTF-8, ISO-8859-1...) de la web. También incluye el código de la respuesta, siendo algunos de los más comunes el *código 200 - Solicitud exitosa* o *el código 404 - Web no encontrada*.

La última versión estable de la librería a fecha 22 de febrero de 2023 es la versión 2.28.2.

Librería de `Python BeautifulSoup`

La librería BeautifulSoup es una librería de `Python` usada para extraer datos a partir de ficheros HTML o XML [20] creada por Leonard Richardson. Funciona con cualquier otro parser para proveer una manera idiomática de navegar, buscar y modificar el árbol con todos los elementos del documento.

Existen varias versiones de la librería, siendo la última la librería conocida como *beautifulsoup4*, que es compatible con una amplia variedad de analizadores HTML y XML, incluyendo algunas de las más populares como *html.parser*, *lxml* y *html5lib*. La última versión estable de la librería a 22 de febrero de 2023 es la versión 4.11.1.

Librería de `Python Selenium`

La librería `Selenium` es una librería usada en la automatización de navegadores web, comúnmente utilizada en la automatización de pruebas de software y de tareas repetitivas en la web [75]. Es por ello que es usado para la creación de *Web Crawlers* (3.1.1) y *Web Scrapers* (3.1.2).

`Selenium` dispone de una interfaz que permite interactuar con una amplia variedad de navegadores web y simular acciones del usuario, tales como hacer clic en enlaces y botones, llenar formularios y navegar a diferentes páginas. Además tiene compatibilidades con scripts Javascript.

La última versión estable de la librería a 22 de febrero de 2023 es la versión 4.8.2.

Librería de `Python Pyppeteer`

`Pyppeteer` es una librería de `Python` para la automatización de navegadores web. Se basa en la librería Puppeteer de Javascript [61], la cual tiene un funcionamiento muy similar a `Selenium` pero limitado a navegadores Chrome/Chromium.

La última versión estable de la librería a 22 de febrero de 2023 es la versión 1.0.2.

Comparativa y selección

Para la tarea de *Web Scraping*, finalmente se ha decidido usar la librería **`BeautifulSoup`** debido a su capacidad de realizar la tarea de *Web Scraping* y *Web Crawling* de manera completa y eficiente, a su simplicidad de uso e implementación y a la amplia de disponibilidad de documentación. Debido a que `BeautifulSoup` se apoya en la librería `Request`, dicha librería ha sido usada también en la recuperación de contenido de las páginas webs.

En el caso de alguna fuente de datos, como ElMundoToday (4.6), se ha usado a mayores de `BeautifulSoup` la librería **`Selenium`** debido a la existencia de algunos botones de navegación que no podían ser utilizados por `BeautifulSoup`. La librería `Pyppeteer` ha sido finalmente descartada debido a que, a pesar de que dispone de la capacidad de pulsar dichos botones de navegación, es una librería mucho más incompleta y compleja de usar en un entorno `Python` que `Selenium`.

7.2.2 Opciones para implementación de NLP

Librería de `Python NLTK`

La librería `NLTK` (*Natural Language ToolKit*) es una de las librerías para NLP más usadas y populares entre la comunidad. Fue desarrollada en el año 2001 por Steven Bird y Edward Loper en la Universidad de Pennsylvania [21]. Desde entonces ha estado en contínuo desarrollo. En el año 2009, los autores de la librería la tradujeron de `Python` 2 a `Python` 3 y publicaron el libro *"Natural language processing with `Python`: analyzing text with the natural language toolkit"* [11], donde se incluían ejemplos de uso de esta librería.

`NLTK` proporciona herramientas fáciles de utilizar para analizar y manipular los datos que se desprenden del lenguaje natural humano. Se usa tanto a nivel académico en investigaciones, como en el desarrollo de aplicaciones que usen NLP. Algunas de las principales funciones que incluye son: tokenización, etiquetado gramatical (también conocido como POS *Tagging*) para etiquetar el tipo de cada palabra, análisis sintáctico, *stemming* y lematización, reconocimiento de entidades nombradas (NER) y análisis de sentimiento, entre otras funciones.

El análisis de sentimientos se realiza con `VADER` (*Valence Aware Dictionary and sEntiment ReasoneR*) que es una sub-librería de `NLTK` y que está diseñada especialmente para trabajar con texto de redes sociales y texto informal, incluyendo emoticonos, acrónimos y jerga informal. VADER se basa en un enfoque léxico y heurístico, lo que implica que utiliza un léxico o conjunto predefinido de palabras asociadas con unas determinadas puntuaciones de sentimiento y ciertas reglas heurísticas para calcular el sentimiento general de un texto [22]. Su ventaja frente a otras librerías de análisis de sentimientos es la facilidad de implementación, ya que no necesita grandes conjuntos de entrenamiento.

La última versión estable de la librería `NLTK` a 22 de febrero de 2023 es la versión 3.4.5, lanzada el 20 de agosto de 2019.

Librería de `Python` Pysentimiento

La librería `Pysentimiento` es una librería de `Python` de código abierto desarrollada para realizar análisis de sentimientos en textos en español, inglés, italiano y portugués (a fecha mayo de 2023). Fue lanzada en el año 2021 y presentada en el paper *"pysentimiento: A `Python` Toolkit for Sentiment Analysis and SocialNLP tasks"(Juan Manuel Pérez et al.)* [74]. Entre sus funciones se encuentra la detección de discursos de odio, de ironía, de emociones y de sentimientos, así como el etiquetado de entidades y de estructuras gramaticales. La librería apareció como respuesta a otras librerías para análisis de sentimientos y APIs que solo se podían aplicar en al lenguaje inglés, para facilitar el uso de estas herramientas en castellano.

Su funcionamiento se basa en la arquitectura *transformer* y en modelos de lenguaje en castellano e inglés que usan dicha arquitectura. Para entrenar su funcionamiento en inglés usaron BERT base [19], RoBERTa base [58], BERTweet [65] y otros modelos multilenguaje, como DistilBERT [81] and mBERT [19]. Para el lenguaje en castellano, se usaron menos modelos para el entrenamiento ya que hay pocos modelos disponibles véase BETO [13] (la versión de BERT en castellano) y las librerías multilenguaje anteriormente mencionadas.

La última versión estable de la librería Pysentimiento a 22 de febrero de 2023 es la versión 0.5.2.

Librería de `Python` spaCY

La librería de `Python` `spaCY` es una librería gratuita de código abierto usada para NLP. Está diseñada para construir aplicaciones que trabajen y procesen extensas cantidades de texto [38], haciendo que sea muy sencillo extraer información de los textos, hacer funcionar sistemas que incluyan Natural Language Understanding (un subcampo del NLP) o preprocesar el texto para su uso en otras librerías de Deep Learning como `Pytorch` o `Tensorflow`. También se encuentra disponible en `Cython`, una versión de `Python` que soporta llamadas a funciones y declaraciones del lenguaje `C`, así como a sus variables y atributos de clase [10].

`SpaCY` viene con una gran cantidad de Pipelines preentrenadas y actualmente soporta la tokenización y entrenamiento de modelos en más de 70 lenguajes (incluyendo español e inglés). La libería usa tecnología puntera basada en modelos de redes neuronales [39] para las diferentes tareas que se mencionarán más adelante. También incluye el uso de *transformers* preentrenados como BERT [3].

Algunas de las tareas que puede realizar esta librería son:

- Tokenización: segmentar el texto en palabras, signos de puntuación u otros símbolos.
- *Part-of-Speech* (POS) *Tagging*: asignar a cada palabra su tipo de palabra (verbo, adjetivo, adverbio, pronombre, sustantivo, preposiciones, conjunciones y artículos).
- Extracción de dependencias: analiza la relación sintáctica de los tokens.
- Lematización.
- *Named Entity Recognition* (NER) y *Entity Linking* (EL): reconoce entidades o nombres propios del mundo real. Posteriormente se puede aplicar Entity Linking para tratar con los problemas de ambigüedad de ciertas entidades.
- Similaridad: compara diferentes textos para posteriormente ver lo similares que son.

- Clasificación de texto: asignar categorías o etiquetas a un documento o parte de él.
- Entrenamiento de nuevos modelos estadísticos para NLP.

La última versión estable de la librería SpaCY a 22 de febrero de 2023 es la versión 3.5.0.

Comparativa y selección

Existe otra alternativa a las diferentes librerías vistas que se planteó en los estados más iniciales del proyecto que consistía en crear y entrenar nuestros propios modelos de lenguaje para las dos tareas planteadas para este apartado (análisis de sentimientos y reconocimiento de entidades) pero debido a su alto coste tanto computacional como temporal se desechó rápidamente.

Finalmente las librerías seleccionadas y usadas fueron:

- **SpaCY**: se ha elegido SpaCY por su facilidad de uso, su eficiencia y su rapidez de funcionamiento en el lenguaje castellano. Además la librería presenta unos modelos preentrenados muy útiles y adecuados para este proyecto, ya que son modelos que han sido preparados y entrenados con datos de noticias reales. Se ha usado en el preprocesamiento de texto y en el reconocimiento de entidades (NER).
- **Pysentimiento**: para el análisis de sentimientos se ha seleccionado Pysentimiento, ya que tras comparar su funcionamiento con algunas de las noticias que se han extraído en castellano, ha sido el que mejor resultado ha dado.

7.2.3 Herramientas de desarrollo web

Front-end

Para el front-end se ha partido de la base de **HTML5 y CSS3** y posteriormente se planteó el uso de algunos *frameworks* para el desarrollo front-end tales como:

- **`Bootstrap`**: es un framework open-source para desarrollo de front-end creado por un desarrollador y un diseñador de Twitter a mediados de 2011 [90]. Es una de las herramientas más populares y usadas por comunidad ya que permite crear rápida y eficientemente webs responsivas y visualmente atractivas para el usuario.

  Es muy sencillo de usar, ya que simplifica en gran medida el desarrollo web al proporcionar una serie de componentes(rejillas, botones, barras de navegación, formularios...), estilos y otras funcionalidades, como la interactividad con JavaScript.

  La última versión a fecha 22 de febrero de 2023 es `Bootstrap5` (v5.2.3).

- **Angular**: es un framework open-source para desarrollo de front-end creado por Google que está basado en JavaScript, aunque dispone también de soporte de TypeScript [43]. Fue lanzado en septiembre del año 2016. Requiere una base de HTML, CSS y Javascript y sus diferentes estándares.

  Es un framework orientado a la optimización y rapidez de aplicaciones web de una sola página (*single-page applications*), debido a que primero la interfaz es mostrada y cargada al arranque de la aplicación y luego solo se cargan solo los datos. Angular se basa en la arquitectura Modelo-Vista-Controlador (MVC) en el cual se separa la aplicación en tres partes distinguibles en la implementación: el modelo se encarga de la manipulación de datos, el controlador se encarga de la lógica de la aplicación y la vista de la interfaz [93]. Al igual que otros *frameworks* similares, dispone de componentes y templates propios.

  La última versión estable a fecha 22 de febrero de 2023 es Angular v15.2.0.

Selección: finalmente debido a su facilidad de uso y rapidez de aprendizaje se ha optado por usar **Bootstrap** además de que la base inicial de conocimiento de HTML, CSS y JavaScript del alumno eran demasiado básicos para poder usar Angular.

Back-end

Para el desarrollo del back-end, de la lógica y de la interacción con la base de datos del sistema se planteó desde un inicio el uso de *frameworks* basados en el lenguaje `Python`, ya que las otras partes de la plataforma (Scrapers y NLP) se habían implementado ya en este lenguaje. Las opciones planteadas son:

- **Flask**: es un framework web ligero y flexible que permite el desarrolo sencillo de aplicaciones web en `Python` [17]. Es usado en algunas aplicaciones populares como Pinterest [14] o LinkedIn [80]. Fue creado en el año 2010 por Armin Ronacher a modo de broma de *April Fool's* pero adquirió rápidamente la fama suficiente como para ser desarrollado en un producto completo [79].

  `Flask` se basa en el concepto de *"microframework"*, lo que implica que proporciona la base para construir las aplicaciones web pero sin imponer demasiadas restricciones o dependencias. Esto tiene algunos pros y contras importantes a tener en cuenta. Un pro importante es que es una librería muy ligera, con pocas dependencias que actualizar y tener en cuenta para cuestiones de seguridad. Una contra también importante es que al ser tan ligero, a veces el desarrollador tiene que realizar más trabajo o añadir a mano dependencias o *plugins*.

  Los componentes principales de `Flask` son:

  - `Werkzeug`, una herramienta de utilidad para el lenguaje `Python` para *Web Server Gateway Interface* (WSGI) [71] que sirve para describir como el servidor web

se comunica con la aplicación web, tal y como se describe en el estándar de `Python` PEP 3333 [30].

- `Jinja2`, una herramienta rápida y extensible para la creación de plantillas HTML con algunos marcadores especiales que permiten escribir código similar a la sintaxis de `Python`. `Jinja2` a su vez depende de `Markupsafe`, que se encarga de controlar la entrada de la aplicación cuando se renderizan las plantillas para evitar ataques a la aplicación [70].

La última versión estable a 22 de febrero de 2023 es la 2.2.3.

- **`Django`**: es un framework web de alto nivel, *open-source* basado en el lenguaje `Python` y que sigue la arquitectura Modelo-Template-Vista, similar a la explicada en 7.2.3, que en este caso consta de Template que es la interfaz de la aplicación, Vista en este caso es la lógica de la aplicación. Fue creado en otoño del año 2003 por los programadores web del periódico *Lawrence Journal-World*, Adrian Holovaty y Simon Willison. Fue publicado para uso del público general en julio de 2005 [95].

  `Django` dispone de una serie de características y herramientas que permiten un desarrollo de web fácil. Algunas de las características más importantes son según la documentación del framework [40]:

  - *Object-Relational Mapping* (ORM): el ORM permite a `Django` interactuar con bases de datos relacionales utilizando objetos de `Python` en lugar de tener que construir y usar consultas SQL, lo que facilita la gestión de la base de datos y la portabilidad a diferentes motores.
  - Seguridad: de base, ofrece diferentes herramientas para aportar seguridad y protección a las aplicaciones que se construyan con el framework. Incluye defensa ante métodos de seguridad comunes tales como inyecciones de SQL, ataques de scripting entre sitios (XSS) y ataque de falsificación de solicitudes entre sitios (CSRF).
  - Enrutamiento: el enrutador de `Django` es flexible y mapea las URLs de las vistas-controlador correspondientes. También permite optimizar la ruta para mejores resultados en motores de búsqueda

La última versión estable a 22 de febrero de 2023 es la 4.1.7.

Selección: debido a la facilidad de uso y a la simplicidad y ligereza de la instalación, además de la facilidad de incluir librerías que se han usado anteriormente y con las que el alumno estaba familiarizado, se decidió usar **`Flask`** para la implementación del back-end del sistema.

7.3 Implementación

7.3.1 *Web Scrapers*

Una vez seleccionadas las herramientas para el *Web Scraping* (`BeautifulSoup` y `Selenium`) se procedió a implementar los *scrapers* para cada una de las fuentes siguiendo el patrón *Template* descrito en 6.1. Para ello primero se implementó la clase abstracta *WebScraper* donde se define el método *template* donde se define el esquema o esqueleto que seguirán las subclases concretas que en este caso son los *scrapers* de cada una de las fuentes definidas en el capítulo 4.

Se definen los siguientes métodos abstractos que deben ser implementados por las subclases concretas:

- `iterate_articles`: define la manera que tiene el *scraper* de recorrer los artículos de la página web o blog.
- `extract_headline`: define la manera que tiene el *scraper* de extraer el titular de una noticia.
- `extract_date`: define la manera que tiene el *scraper* de extraer el la fecha de publicación de un artículo.
- `extract_content`: define la manera que tiene el *scraper* de extraer el texto con el contenido de la noticia.
- `extract_author`: define la manera que tiene el *scraper* de extraer el autor.
- `extract_tags`: define la manera que tiene el *scraper* de extraer las etiquetas si las hubiera.
- `extract_maxpag`: define la manera que tiene el *scraper* de extraer el límite de la paginación de una web.

Para guardar los datos extraídos en la base de datos, en ocasiones ha habido que transformar la información para que fuera compatible con la implementación de dicha base de datos. La clase *WebScraper* se encarga de inicializar la conexión con la base de datos con la librería `Psycopg2` y posteriormente implementa diferentes métodos para guardar en la base de datos los datos extraídos.

Hay páginas webs que bloquean a los usuarios que realizan muchas solicitudes consecutivas, por lo que para evitar que las webs de noticias bloqueen el acceso a los *scrapers* se han tomado las siguientes medidas:

- Tiempo de espera entre peticiones

- Cambio de User Agent (Agente de Usuario) entre peticiones.

Para lanzar los *scrapers* se decidió crear unos ficheros *Launcher* en los cuales se instancian cada uno de los 6 *scrapers* creados. Dentro del fichero se puede seleccionar la fecha límite de las noticias que se van a extraer desde el momento de lanzamiento. Finalmente, se ha creado un *daemon* que permite lanzar de forma diaria estos *Launchers*, que son lanzados de forma concurrente usando la librería `Subprocess`.

7.3.2 Web App con `Flask`

Para la creación de la web con `Flask`, se ha desarrollado por una parte la parte del *frontend* o interfaz y por otra la parte del *backend* o lógica. Para la interfaz se ha usado `HTML`, `CSS` y `Bootstrap`. `Flask` se encargaba de gestionar la parte lógica y mandar los datos a la interfaz para mostrarlos. `Flask` se conecta a la base de datos de noticias mediante `Psycopg2` y extrae tanto el listado de noticias como el contenido de las noticias individuales para mostrárselas al usurario.

El uso de `Flask` para la página de las noticias con paginación y filtro planteó varios problemas a la hora de mantener el estado de los filtros del usuario al cambiar de página, por lo que fue necesario implementar a mayores cookies de sesión que permiten hacer un seguimiento de los filtros seleccionados.

7.3.3 Procesamiento del Lenguaje Natural

Para esta parte se han usado las librerías `SpaCy` y `Pysentimiento`, pero a mayores se ha realizado un preprocesamiento inicial del texto en el cual se eliminaban entre otras cosas los enlaces que aparecían en el texto de algunas noticias, junto con símbolos especiales como paréntesis o corchetes. Ambas partes son llamadas por `Flask` cuando el usuario selecciona una noticia.

Análisis de sentimientos

Con `Pysentimiento` se ha inicializado un analizador de sentimientos que permite ver frase por frase de la noticia el sentimiento (negativo, neutral o positivo). Una vez analizadas todas las oraciones se sacaba el sentimiento predominante a partir del conteo de frases de cada sentimiento que había. El analizador devolvía el sentimiento predominante de la noticia, los sentimientos individuales de cada oración y cada oración preprocesada. Estos dos últimos datos son los que permiten que el *frontend* pinte sobre el texto de las noticias.

Reconocimiento de entidades

Con SpaCy se ha realizado la parte del reconocimiento de entidades, para ellos se han seleccionado todas aquellas palabras identificadas por el modelo de SpaCy que tenían una de las clases de nombre propio que fuera diferente de otros. Finalmente, se devuelve una lista que contiene las entidades y su clase.

7.3.4 Despliegue web

Para poder hacer accesible la página web (definida en 1.3 y 5.2.2) desde cualquier ordenador a través de Internet, se decidió desplegar el servicio web en una máquina virtual de la escuela de Ingeniería Informática de Valladolid.

La máquina que se solicitó inicialmente fue una máquina con sistema operativo Ubuntu 22.04.2 LTS de 2 núcleos, 2G de memoria RAM y 20G de disco duro HDD. Posteriormente se pidió una ampliación de la memoria RAM a 8G para asegurar el buen funcionamiento del sistema.

Docker

Se ha elegido para el montaje y despliegue del servidor web en la máquina virtual usar Docker [24]. Docker es una tecnología de contenedores de código abierto que permite el desarrollo, el envío y la ejecución de aplicaciones en diferentes entornos. Se tomó esta decisión ya que permite crear, probar e implementar de manera sencilla y rápida aplicaciones.

Docker permite virtualizar un sistema operativo, proporcionando un nivel mayor de abstracción utilizando características y espacios de nombre que permite que los varios contenedores o sistemas se ejecuten y desplieguen en una sola instancia de Linux. El sistema Docker se compone de dos elementos :

- **Imágenes**: es una plantilla ligera, independiente y ejecutable que incluye todo lo necesario para ejecutar un pedazo de software, incluyendo código, tiempo de ejecución, bibliotecas del sistema, variables de entorno y archivos de configuración [26].
- **Contenedores**: es una instancia en tiempo de ejecución de una imagen Docker. Se puede crear, iniciar, detener, mover o borrar un contenedor. Varios contenedores de una misma imagen pueden ser ejecutados al mismo tiempo y contienen todo lo necesario para que las aplicaciones puedan ejecutarse correctamente [27].

Algunas de las ventajas más importantes de usar Docker son [29]:

- **Aislamiento de recursos/Modularidad**: `Docker` se asegura de que las aplicaciones y sus dependencias estén aisladas unas de otras, lo que implica que si una aplicación necesita una versión específica de una biblioteca se puede instalar esa versión en el contenedor de la aplicación sin que afecte a otras aplicaciones.
- **Rápidez y ligereza**: Los contenedores `Docker` son mucho más rápidos y ligeros que las máquinas virtuales tradicionales, ya que los contenedores comparten el mismo sistema operativo mientras que las máquinas virtuales necesitan cada una de su propio sistema operativo.
- **Facilidad de manejo y mantenimiento**: `Docker` simplifica el proceso de configuración y mantenimiento de las aplicaciones y sus dependencias, ya que permite codificar la configuración y las dependencias de las imágenes. También dispone de control de versión de imágenes, lo que permite restaurar o modificar la imagen en caso de ser necesario.

`Docker` usa una arquitectura cliente-servidor como la que se encuentra detallada en la siguiente figura. El cliente habla con el daemon `Docker` que hace el trabajo de construir, ejecutar y distribuir los contenedores `Docker`. El cliente `Docker` y el daemon pueden ejecutarse en el mismo sistema, o puede conectar un cliente `Docker` a un daemon remoto. Se comunican mediante una API REST, a través de sockets UNIX o una interfaz de red. [26]

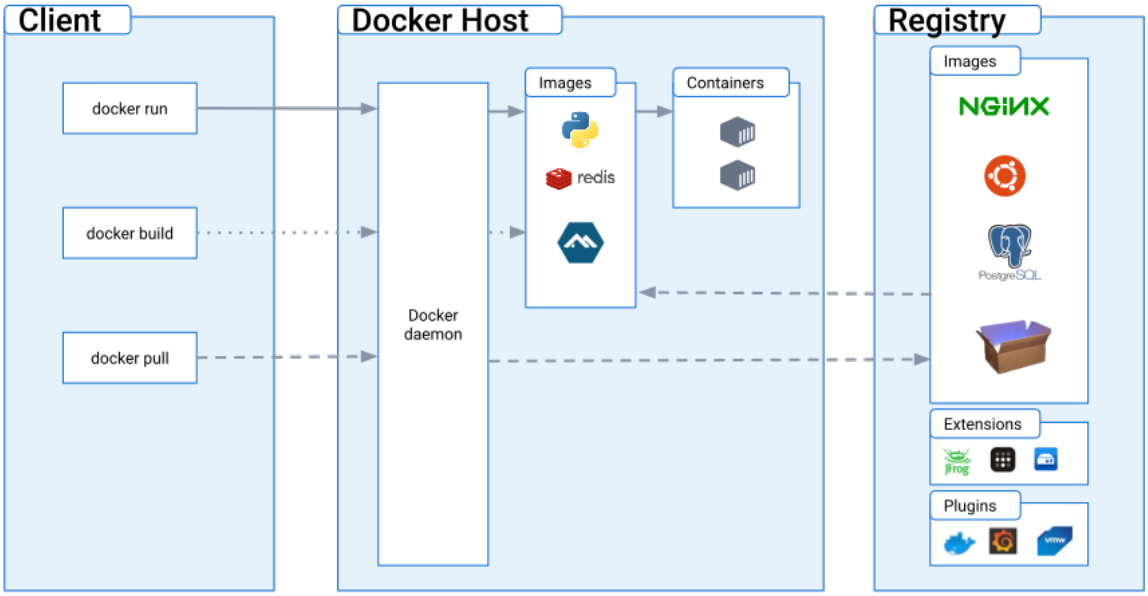


Figura 7.1: Arquitectura `Docker` [26]

Las aplicaciones `Docker` se crean a partir de un script llamado Dockerfile, que es un

archivo de texto que contiene las instrucciones para construir una imagen `Docker`. Entre las instrucciones de un Dockerfile hay que destacar [28]:

- **FROM**:descripción de la imagen que se está construyendo.
- **RUN**: ejecución de comandos como en la línea de comandos de la consola.
- **COPY**: añadir ficheros con el código que se desea ejecutar.
- **EXPOSE**: instrucción que informa a `Docker` que el contenedor está escuchando en un puerto de red específico.
- **WORKDIR**: establecer el directorio de trabajo.
- **CMD**: comando que se ejecuta la lanzar el `Docker`.

Docker Compose

En el proyecto, se ha usado a mayores la herramienta `Docker Compose` que permite definir y ejecutar aplicaciones de múltiples contenedores `Docker` de manera más sencilla que contruir, desplegar y conectar `Docker` individuales.

En `Docker Compose` aparece el concepto ”Servicio”que es una abstracción que representa un contenedor `Docker` y su configuración. A mayores, nos permite crear los ”Volúmenes”dentro de los servicios para la persistencia de datos entre reinicios de los contenedores [25].

En nuestro caso, al desplegar el `Docker Compose` se lanzaban los siguientes 3 servicios:

- **Base de datos PostgreSQL**: se inicializa el servicio PostgreSQL y se establecen los volúmenes con la base de datos donde se van a guardar las noticias que se van extrayendo.
- **App Scraping**: lanzado mediante el dockerfile para la ejecución de los scrapers (definido en Apéndice B), que depende y está conectado al volumen de la base de datos.
- **App `Flask` Web**: lanzado mediante el dockerfile para la ejecución de la propia plataforma web (definido en Apéndice B), que depende y está conectada al volumen de la base de datos.

Capítulo 8

Pruebas

8.1 Pruebas *Web Scrapers*

Se han realizado diferentes pruebas para verificar el correcto funcionamiento del sistema de *Web Scraping*. Las pruebas se han realizado para los diferentes *Web Scrapers* y fuentes de noticias planteadas.

- **Pruebas de integración**: las primeras pruebas realizadas fueron las pruebas de integración, donde se verificó que todas las partes de los Scrapers (como visto en el apartado de Diseño 6.1) funcionaban correctamente, es decir, que el programa es capaz de extraer los datos pedidos de una web de noticias y puede almacenarlos correctamente en la base de datos.

- **Pruebas de carga**: con el fin de hacer que el proceso de extracción de noticias fuera lo más eficiente y llevara el menor tiempo posible, se planteó el lanzamiento de forma concurrente de todos los *Web Scrapers*. En las pruebas de carga se comprobó que podían funcionar de manera simultánea sin tener un impacto en el rendimiento o en los datos extraídos.

- **Pruebas de recuperación de fallos**: una parte importante de los *Web Scrapers* era la capacidad de recuperarse de fallos que pudieran producirse durante la extracción de noticias, como por ejemplo un corte de conexión con la base de datos, una caída de la web a extraer o un cambio en la estructura de dicha web.

- **Pruebas de aceptación**: en este último grupo de pruebas se validó que el sistema cumple con los requisitos planteados en el apartado 5.2.1.

8.1.1 Listado y descripción de las pruebas

Test 1.1	**Extracción del titular de una noticia**
Tipo(s)	Prueba de integración, prueba de aceptación
Descripción	El Scraper navega por la web y extrae el titular de una noticia.
Resultado esperado	Titular extraído y guardado en la base de datos
Resultado obtenido	Resultado esperado

Cuadro 8.1: Test 1.1

Test 1.2	**Extracción del cuerpo de una noticia**
Tipo(s)	Prueba de integración, prueba de aceptación
Descripción	El Scraper navega por la web y extrae el cuerpo de una noticia.
Resultado esperado	Cuerpo extraído y guardado en la base de datos
Resultado obtenido	Resultado esperado

Cuadro 8.2: Test 1.2

Test 1.3	**Extracción del fecha de publicación y fecha de recolección de una noticia**
Tipo(s)	Prueba de integración, prueba de aceptación
Descripción	El Scraper navega por la web y extrae la fecha de publicación de una noticia.
Resultado esperado	Fecha de publicación extraído y guardado en la base de datos. Fecha de recolección guardada en la base de datos
Resultado obtenido	Resultado esperado

Cuadro 8.3: Test 1.3

Test 1.4	**Extracción del enlace de una noticia**
Tipo(s)	Prueba de integración, prueba de aceptación
Descripción	El Scraper navega por la web y extrae el enlace de una noticia.
Resultado esperado	Enlace extraído y guardado en la base de datos.
Resultado obtenido	Resultado esperado

Cuadro 8.4: Test 1.4

Test 1.5	**Extracción del tema o tópico de una noticia**
Tipo(s)	Prueba de integración, prueba de aceptación
Descripción	El Scraper navega por la web y extrae el tópico de una noticia.
Resultado esperado	Tópico extraído y guardado en la base de datos.
Resultado obtenido	Resultado esperado

Cuadro 8.5: Test 1.5

Test 1.6	**Extracción de la fuente de una noticia**
Tipo(s)	Prueba de integración, prueba de aceptación
Descripción	El Scraper guarda la fuente de la noticia.
Resultado esperado	Fuente guardada en la base de datos.
Resultado obtenido	Resultado esperado

Cuadro 8.6: Test 1.6

Test 1.7	**Extracción del autor de una noticia**
Tipo(s)	Prueba de integración
Descripción	El Scraper navega por la web y extrae el autor de una noticia.
Resultado esperado	Autor extraído y guardado en la base de datos.
Resultado obtenido	Resultado esperado

Cuadro 8.7: Test 1.7

Test 1.8	**Extracción de etiquetas de una noticia**
Tipo(s)	Prueba de integración
Descripción	El Scraper navega por la web y extrae las etiquetas o tópicos menores de una noticia.
Resultado esperado	Etiquetas extraídas y guardadas en la base de datos.
Resultado obtenido	Resultado esperado

Cuadro 8.8: Test 1.8

Test 1.9	**Lanzamiento simultáneo de dos o más *Web Scrapers***
Tipo(s)	Prueba de carga
Descripción	Se lanzan dos o más *Web Scrapers* de forma simultánea y concurrente.
Resultado esperado	Las noticias se extraen y se guardan en la base de datos sin problema.
Resultado obtenido	Resultado esperado

Cuadro 8.9: Test 1.9

Test 1.10	**Caída del servidor de la base de datos**
Tipo(s)	Prueba de recuperación de fallos
Descripción	Se produce un cambio de red, desconexión o caída del servidor donde se aloja la base de datos
Resultado esperado	Se detiene la extracción y se queda a la espera de reanudar, sin provocar un colapso del Web Scraper.
Resultado obtenido	Resultado esperado

Cuadro 8.10: Test 1.10

Test 1.11	**Cambio en el la estructura DOM de una web**
Tipo(s)	Prueba de recuperación de fallos
Descripción	Se modifica la estructura de una de las webs a extraer
Resultado esperado	El Scraper atrapa el error correctamente sin parar el programa y continua la extracción.
Resultado obtenido	Resultado esperado

Cuadro 8.11: Test 1.11

8.2 Evaluación de la web

Para hacer una evaluación general de algunos aspectos importantes de la plataforma web como rendimiento, accesibilidad o SEO (Search Engine Optimization) se ha usado una herramienta llamada **`Lighthouse`** [18], una herramienta de código abierto creada por Chrome Developers que permite automatizar estos aspectos mencionados anteriormente dando una métrica cuantitativa basada en sus propios test internos. Además de proporcionar las métricas, `Lighthouse` te proporciona en su informe consejos, errores y buenas prácticas que se pueden intentar aplicar para mejorar la página.

8.2.1 *Landing page* y listado de noticias

Usando `Lighthouse` en la *landing page* de la plataforma hemos obtenido los siguientes resultados:

- **Rendimiento**: 93/100. La página web tiene buen rendimiento, tarda en cargar el contenido 2496 milisegundos (2.5 segundos aproximadamente). `Lighthouse` recomienda que se permita la compresión de texto y que se optimicen las imágenes trabajando con formatos de imagen de siguiente generación como WebP o AVIF ya que ofrecen mejor compresión de imagen.
- **Accesibilidad**: 98/100. `Lighthouse` recomienda que se mejore el contraste de los botones que aparecen en la página ya que el texto blanco sobre azul puede dificultar la lectura de algunos usuarios.
- **Buenas prácticas**: 83/100. este apartado permite ver la limpieza general del código de la web. `Lighthouse` señala dos cuestiones que considera críticas y a mejorar.

  La primera que señala es el uso de HTTPS. En nuestro caso, por la naturaleza de la máquina virtual proporcionada no se dispone de certificado digital ya que para ello los técnicos tendrían que haber pasado la clave privada del certificado con todos los posibles problemas que ello podría plantear. Se plantearon dos opciones alternativas. La primera era pedir un alias para la máquina virtual que sería pasado por el proxy y funcionaría mediante SSL. La segunda alternativa era usar certificados digitales gratuitos. Finalmente se decidió que no se usaría HTTPS por lo que no se usó ninguna de estas opciones.

  La segunda cuestión que señala `Lighthouse` es el uso de las librerías `Bootstrap` 4.0.0 y `jQuery` 3.2.1 que tienen varias vulnerabilidades de seguridad conocidas.
- **SEO**: 92/100. `Lighthouse` solo señala un fallo en el cual la página no tiene una descripción meta, lo cual afecta en los resultados de búsqueda ya que no se puede resumir la página de forma concisa.

A mayores, `Lighthouse` señala que la aplicación no se encuentra optimizada para dispositivos móviles, lo cual es cierto ya que la aplicación está pensada para ser usada en un ordenador.

8.2.2 Página de una noticia

Usando `Lighthouse` en una de las páginas que contienen una noticia hemos obtenido los siguientes resultados:

- **Rendimiento**: 84/100. La página tarda en cargar el primer contenido 2496 milisegundos (2.5 segundos aproximadamente), aunque tarda en cargar por completo

unos 13842 milisegundos (13.8 segundos). `Lighthouse` recomienda lo mismo que en la *landing page*, pero a mayores señala como un error el tiempo inicial de respuesta del servidor que es muy elevado. En este caso, con los recursos computacionales y de tiempo disponibles no podemos implementar una solución a esto, ya que la razón del retraso en la respuesta es por los modelos que carga `Pysentimiento` para el análisis de sentimiento del texto de la noticia y los modelos de `Spacy` para el reconocimiento de entidades. Una de las opciones que se podrían plantear para arreglar esto sería almacenar en una base de datos el resultado de el análisis en lugar de calcularlo siempre al cargar la página.

- **Accesibilidad**: 93/100. `Lighthouse` señala de nuevo el contraste de los botones pero esta vez a mayores señala que los elementos de cabecera no están ordenados en orden descendente ya que ayuda a navegar y entender el texto cuando se usan asistencias técnicas en la lectura.
- **Buenas prácticas**: 83/100. `Lighthouse` señala de nuevo los mismo errores que en la *landing page*.
- **SEO**: 92/100. `Lighthouse` señala de nuevo el mismo error que en la *landing page*.

A mayores, Lighthouse señala que la aplicación no se encuentra optimizada para dispositivos móviles, lo cual es cierto ya que la aplicación está pensada para ser usada en un ordenador.

Capítulo 9

Conclusiones

La principal conclusión de este Trabajo Fin de Grado es la cobertura completa y consecución de todos los objetivos, tanto principales como secundarios, definidos en apartado 1.3.

Se han desarrollado los *Web Scrapers* para la extracción de noticias de webs y blogs de noticias, así como la aplicación web para mostrar los resultados de la extracción y del posterior procesamiento del texto de dichas noticias que se ha implementado usando técnicas de Procesamiento de Lenguaje Natural.

En cuanto a las tecnologías experimentadas hay que destacar el aprendizaje experimentado por el desarrollador ya que se ha tenido que trabajar con tecnologías de desarrollo web básicas que hasta la realización de ese proyecto desconocía o solamente conocía superficialmente. También hay que señalar el aumento en la familiaridad con el lenguaje `Python` y algunas de las librerías que se han usado para la consecución de los objetivos como las librerías de *Scraping Web* (`Selenium` y `BeautifulSoup`), la librería del reconocimiento de entidades nombradas (`SpaCy`) y la librería de análisis de sentimientos (`Pysentimiento`).

Cabe destacar la evolución en el desarrollo de la aplicación web usando el *framework* `Flask` y el despliegue del producto mediante el uso de la máquina virtual y el entorno y arquitectura `Docker`.

9.1 Trabajo futuro y posibles ampliaciones

Existen muchas mejoras para los diferentes apartados implementados y muchas líneas de trabajo futuras, en parte muchas de ellas impulsadas por el crecimiento que han sufrido las tecnologías del procesamiento de lenguaje en el año 2023 gracias a la

popularización de la IA generativa (Inteligencia Artificial generativa).

- **Extracción de las imágenes de las noticias**: en un futuro en la parte de *Web Scrapers* se podría añadir la capacidad de extraer las imágenes de cabecera que acompañan e ilustran la noticia para su posible y posterior análisis.
- **Detección de contenido generado con IA**: una línea de trabajo muy interesante sería añadir en la parte de procesamiento de la noticia la detección de contenido sintético, ya que en este año 2023 la aparición de este tipo de contenido se encuentra en constante crecimiento y saber detectarlo podría ayudar a identificar la veracidad de una noticia. También se podría aplicar esta parte de detección de contenido sintético a la detección de imágenes extraídas como hemos mencionado en el punto anterior. Actualmente ya hay alguna herramienta que detecta texto generado con inteligencia artificial como Turnitin [89] o el Classifier de OpenAI [1], pero no existe a día de hoy ningún detector con fiabilidad suficiente para imágenes generadas por IA con los modelos nuevos de difusión.
- **Introducción de otras métricas de análisis**: en este Trabajo Fin de Grado se ha trabajado con el análisis de sentimientos y el reconocimiento de entidades nombradas, pero podrían aplicarse muchas otras técnicas de análisis, como por ejemplo el análisis gramatical de una noticia.
- **Mejora de modelos y fine-tuning**: recientemente se ha popularizado el uso de Modelos de Lenguaje Grandes (LLM, *Large Language Models* en inglés), que son modelos de lenguaje pre-entrenados con miles de millones de parámetros y que pueden afrontar una amplia variedad de tareas, incluyendo el análisis de texto. Se podría usar alguno de estos modelos para mejorar la parte de análisis de entidades y sentimientos, e incluso se podría realizar un fine-tuning para su uso con noticias en castellano para unos mejores resultados.
- **Mejora de la interfaz web**: como se ha señalado en el capítulo anterior 8.2, sería posible implementar nuevas mejoras en la interfaz para mejorar la accesibilidad. Un ejemplo de esto sería cambiando los colores de ciertos componentes a colores con más contraste.
- **Uso de HTTPS para la web**: actualmente la web se encuentra alojada en `http://virtual.lab.inf.uva.es/20072` que usa HTTP en vez de HTTPS. Sería conveniente por seguridad, instalar certificados HTTPS.

Apéndices

Apéndice A

Manual de Usuario

En este apéndice se presentará el modo de acceso a la plataforma web, así como las principales acciones que puede realizar el usuario.

A.1 Acceso Web

La dirección donde se encuentra alojada la plataforma web es:

`http://virtual.lab.inf.uva.es/20072`

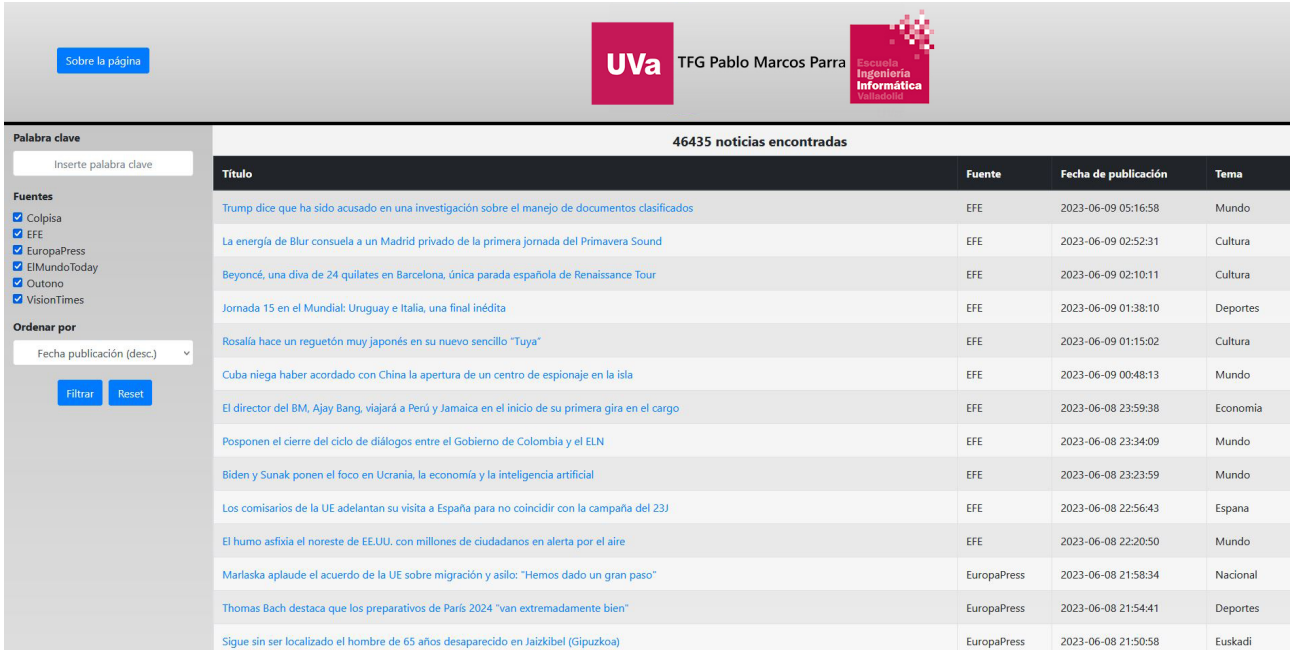


Figura A.1: *Landing page* (09/06/2023)

Parte del código y documentación de la web se encuentra alojado en:

`https://gitlab.inf.uva.es/pabmarc/tfg-informatica-pablomarcos`

A.2 Acciones del usuario

- Si el usuario pulsa con el ratón el icono de la aplicación web, con los logos de la Universidad de Valladolid y la Escuela de Ingeniería Informática de Valladolid, que se encuentra visible en todas las páginas de la web en el centro arriba de la pantalla, el usuario es llevado de vuelta a la *landing page*.

Figura A.2: Icono aplicación web

- En la parte superior izquierda de la *landing page*, hay un botón ”Sobre la página”que lleva a una página en la que se detalla el contenido y forma de la página web. Ese botón aparece a la derecha en la página donde se muestra una noticia. Una vez en esa página, se puede pulsar el botón ”Volver”para regresar a la página de procedencia.

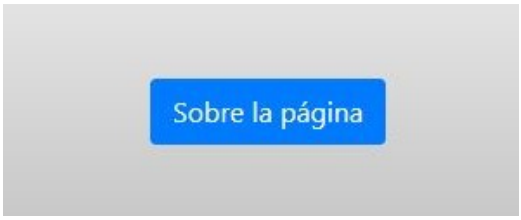


Figura A.3: Botón ”Sobre la página”

- En la parte izquierda de la *landing page* se encuentra el panel de los filtros que se pueden aplicar al listado de noticias.
  - Filtrar por palabra clave: el usuario puede pulsar con el ratón en el cuadro de búsqueda e introducir por teclado la o las palabras claves que quiere encontrar en los titulares de las noticias.
  - Fuentes: el usuario puede pulsar con el ratón las fuentes que desea o no desea ver en el listado. Al filtrar, se mostrarán las noticias de aquellas fuentes que tengan el icono marcado.

- Ordenar por: se puede ordenar el listado de noticias de 6 formas distintas: por fecha de publicación descendente (de más nueva a más antigua), por fecha de publicación ascendente (de más antigua a más nueva), por fecha de extracción descendente (de más nueva a más antigua), por fecha de extracción ascendente (de más antigua a más nueva), por título de la A a la Z y finalmente por título de la Z a la A.

Para aplicar los filtros seleccionados hay que pulsar el botón ”Filtrar”. En caso de querer resetear los filtros y volver a la página de inicio hay que pulsar el botón Reset”.

Figura A.4: Panel con los filtros y sus correspondientes botones

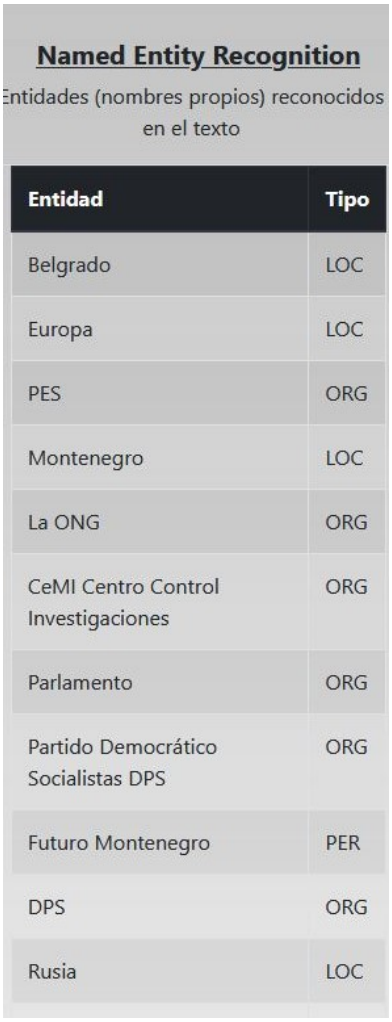
- Para visualizar el contenido completo de una noticia, basta con pulsar con el ratón el título de la noticia que se desea visualizar. Esto llevará al usuario a otra página donde se muestra en el texto de la noticia el sentimiento de cada frase coloreando el texto.

Figura A.5: Ejemplo de noticia de prueba

A la izquierda de la pantalla, aparece el sentimiento predominante de la noticia con la correspondiente leyenda de colores para el texto.

Figura A.6: Resultado y leyenda de colores del análisis de sentimiento

A la derecha, aparece una tabla con el listado de entidades reconocidas en la noticia con su clasificación correspondiente.



Named Entity Recognition

Entidades (nombres propios) reconocidos en el texto

Entidad	Tipo
Belgrado	LOC
Europa	LOC
PES	ORG
Montenegro	LOC
La ONG	ORG
CeMI Centro Control Investigaciones	ORG
Parlamento	ORG
Partido Democrático Socialistas DPS	ORG
Futuro Montenegro	PER
DPS	ORG
Rusia	LOC

Figura A.7: Resultado del análisis de entidades

Para volver a la página anterior, basta con pulsar el botón ”Volver”que se encuentra arriba a la izquierda de la pantalla.

- La plataforma se encuentra páginada y se muestran 25 noticias por página. Para pasar a otra página para ver más noticias, hay que pulsar los botones que aparecen abajo del todo en la página con los controles de la página.

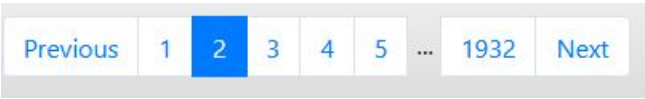


Figura A.8: Resultado del análisis de entidades

Apéndice B

Manual de Instalación

En este apéndice se presentará brevemente cómo instalar y configurar la aplicación web a nivel servidor (como administrador de la aplicación)

B.1 Requisitos

Para poder desplegar la aplicación se necesita disponer de los siguientes requisitos:

- Sistema Operativo GNU/Linux x86_64.
- 8 GB de memoria RAM.
- 20 GB de disco duro disponible.
- Docker [24].
- Docker Compose [25].
- (Opcional) Tarjeta gráfica dedicada.

B.2 Despliegue

Con el fichero `Docker Compose` (*docker-compose.yml*) se pueden lanzar los 3 servicios con el comando

```
docker compose up --build --no-deps --force-recreate -d
```

Se pueden modificar los parámetros de lanzamiento de cada uno de los servicios

Base de datos con PostgreSQL

La base de datos en PostgreSQL se inicializa de la siguiente manera en el Compose:

```
db:
    image: postgres:15
    environment:
      POSTGRES_USER: USERNAME
      POSTGRES_PASSWORD: PWDDB
      POSTGRES_DB: NOMBREDB
    ports:
      - "5432:5432"
    volumes:
      - db-data:/var/lib/postgresql/data
      - ./init.sql:/docker-entrypoint-initdb.d/init.sql
```

Como se ve, se carga la imagen de PostgreSQL versión 15 y se monta el volumen. La base de datos escucha en el puerto 5432 que es el puerto por defecto de PostgreSQL pero se puede cambiar. En init.sql se declaran las tablas para la base de datos:

```
CREATE TABLE news_item(news_id CHAR (36) PRIMARY KEY NOT NULL,
headline VARCHAR (2048), author VARCHAR (256),
publication_date TIMESTAMP, collection_date TIMESTAMP,
topic VARCHAR (128), link VARCHAR (256), content TEXT);

CREATE TABLE source(news_id CHAR (36) NOT NULL, name VARCHAR (128) NOT NULL,
PRIMARY KEY (news_id, name), FOREIGN KEY (news_id)
REFERENCES news_item (news_id));

CREATE TABLE tags(news_id CHAR (36) NOT NULL, tag VARCHAR (128) NOT NULL,
PRIMARY KEY (news_id, tag), FOREIGN KEY (news_id)
REFERENCES news_item (news_id));
```

Scrapers

Para lanzar el proceso de *scraping* de los 6 *scrapers* se muestra a continuación el contenido del Dockerfile:

```
FROM python:3.9

RUN wget -q -O - https://dl-ssl.google.com/linux/linux_signing_key.pub
```

```
| apt-key add -

RUN sh -c 'echo "deb [arch=amd64] http://dl.google.com/linux/chrome/deb/
stable main" >> /etc/apt/sources.list.d/google-chrome.list'

RUN apt-get -y update
RUN apt-get install -y google-chrome-stable

RUN apt-get install -yqq unzip
RUN wget -O /tmp/chromedriver.zip
http://chromedriver.storage.googleapis.com/`curl -sS chromedriver.storage.googleap

RUN unzip /tmp/chromedriver.zip chromedriver -d /usr/local/bin/

ENV DISPLAY=:99

WORKDIR /Scraping

COPY requirements.txt .
RUN pip install -r requirements.txt

COPY . .

CMD ["python","-u", "Scraping1week.py"]
```

Como se puede observar, se lanza un proceso Python tras instalar las librerías de las que depende que están en *requirements.txt* y a mayores se instala el *driver* de Chrome para que la librería `Selenium` funcione correctamente.

Web app con `Flask`

Para lanzar la aplicación web con `Flask` se muestra a continuación el contenido del `Dockerfile`:

```
FROM python:3.9

WORKDIR /app

COPY requirements.txt .

RUN pip install --no-cache-dir -r requirements.txt
```

```
COPY . .

EXPOSE 80

CMD ["gunicorn", "--bind", "0.0.0.0:80", "app:app", "--workers", 5]
```

Como se puede observar, se lanza la aplicación usando `Gunicorn` que es un servidor HTTP WSGI (*Web Server Gateway Interface*), siguiendo ese estándar WSGI que se usa en Python para la interfaz entre aplicaciones web y servidores web. En el archivo se instalan las librerías de las que depende la aplicación y posteriormente se lanza el proceso de `Gunicorn`.

En esa declaración en `CMD` vemos que con la opción `--bind` señala la IP y el puerto en el que la aplicación está escuchando las peticiones de la red, con la opción `--workers` señalamos el número de hilos con el que va a estar trabajando la aplicación. Este número de hilos óptimo depende de la cantidad de núcleos que posea el procesador del servidor siguiendo la fórmula [45]: *Nº de hilos = 2 x Nº de nucleos + 1*.

Bibliografía

[1] Ene. de 2023. URL: https://beta.openai.com/ai-text-classifier.

[2] Media Bias Fact Check (MBFC). *Vision Times: Bias and Fact Checking*. Última consulta: 10/04/23. URL: https://mediabiasfactcheck.com/vision-times/.

[3] Shivaji Alaparthi y Manit Mishra. «Bidirectional Encoder Representations from Transformers (BERT): A sentiment analysis odyssey». En: *arXiv preprint arXiv:2007.01127* (2020).

[4] Davey Alba. *Epoch Media Casts Wider Net to Spread Its Message Online*. Última consulta: 10/04/23. URL: https://www.nytimes.com/2021/03/09/technology/epoch-media-right-wing-disinformation.html.

[5] Portal de Archivos Españoles (PARES). *Institución - Europa Press (España)*. Última consulta: 24/03/23. URL: http://pares.mcu.es/ParesBusquedas20/catalogo/autoridad/145797.

[6] Astah. *About Astah* Professional*. Última consulta: 12/03/23. URL: https://astah.net/products/astah-professional/.

[7] Vimala Balakrishnan y Ethel Lloyd-Yemoh. «Stemming and lemmatization: A comparison of retrieval performances». En: vol. Proceedings of SCEI Seoul Conferences. Seoul, Korea, 2014.

[8] Balsamiq. *About Balsamiq*. Última consulta: 12/06/23. URL: https://balsamiq.com/company/.

[9] Balsamiq. *Balsamiq Wireframes*. Última consulta: 12/06/23. URL: https://balsamiq.com/wireframes/.

[10] Stefan Behnel et al. *About Cython*. Última consulta: 17/05/23. URL: https://cython.org/.

[11] S. Bird, E. Klein y E. Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. O'Reilly Media, Inc., 2009.

[12] María José Canel Crespo. *El País, ABC y El Mundo: tres manchetas, tres enfoques de las noticias*. Servicio Editorial de la Universidad del País Vasco/Euskal Herriko, 1999.

[13] José Cañete et al. «Spanish pre-trained bert model and evaluation data». En: *Pml4dc at iclr* 2020.2020 (2020), págs. 1-10.

[14] Steve Cohen. *What challenges has Pinterest encountered with Flask?* Última consulta: 17/05/23. URL: https://www.quora.com/What-challenges-has-Pinterest-encountered-with-Flask.

[15] Colpisa. *Colpisa: quienes somos*. Última consulta: 24/03/23. URL: https://www.colpisa.com/quienes-somos/.

[16] Colpisa. *Web de Colpisa*. Última consulta: 24/03/23. URL: https://www.colpisa.com/.

[17] Kushal Das. *https://pymbook.readthedocs.io/en/latest/flask.html*. Última consulta: 17/05/23. URL: https://web.archive.org/web/20160604162342/http://mitsuhiko.pocoo.org/flask-pycon-2011.pdf.

[18] Chrome Developers. *Lighthouse Overview*. Última consulta: 09/06/23. URL: https://developer.chrome.com/docs/lighthouse/overview/.

[19] Jacob Devlin et al. «Bert: Pre-training of deep bidirectional transformers for language understanding». En: *arXiv preprint arXiv:1810.04805* (2018).

[20] BeautifulSoup Devs. *BeautifulSoup4 Module Documentation*. Última consulta: 20/02/23. URL: https://beautiful-soup-4.readthedocs.io/en/latest/.

[21] NLTK Devs. *NLTK FAQ*. Última consulta: 09/05/23. URL: https://github.com/nltk/nltk/wiki/FAQ.

[22] NLTK Devs. *nltk.sentiment.vader*. Última consulta: 09/05/23. URL: https://www.nltk.org/_modules/nltk/sentiment/vader.html.

[23] Request Module Devs. *Request Module Documentation*. Última consulta: 19/02/23. URL: https://requests.readthedocs.io/en/latest/.

[24] Docker. *Docker*. Última consulta: 07/06/23. URL: https://www.docker.com/.

[25] Docker. *Docker Compose Model*. Última consulta: 07/06/23. URL: https://docs.docker.com/compose/compose-file/02-model/.

[26] Docker. *Docker Overview*. Última consulta: 07/06/23. URL: https://docs.docker.com/get-started/overview/.

[27] Docker. *Docker Overview: What is a container?* Última consulta: 07/06/23. URL: https://docs.docker.com/get-started.

[28] Docker. *Dockerfile Reference*. Última consulta: 07/06/23. URL: https://docs.docker.com/engine/reference/builder/.

[29] Docker. *Why Docker?* Última consulta: 07/06/23. URL: https://www.docker.com/why-docker/.

[30] P.J. Eby. *PEP 3333 – Python Web Server Gateway Interface v1.0.1*. Última consulta: 17/05/23. URL: https://peps.python.org/pep-3333/.

[31] Agencia EFE. *Archivo histórico de noticias*. Última consulta: 24/03/23. URL: https://efs.efeservicios.com/.

[32] Agencia EFE. *Web de la Agencia EFE*. Última consulta: 24/03/23. URL: https://efe.com/.

[33] Elentir. *Outono (Contando Estrelas)*. Última consulta: 11/04/23. URL: https://www.outono.net/elentir/.

[34] Elentir. *Sobre Outono*. Última consulta: 11/04/23. URL: https://www.outono.net/elentir/sobre-contando-estrelas/.

[35] ElMundoToday. *Quienes somos*. Última consulta: 12/04/23. URL: https://www.elmundotoday.com/quienes-somos/.

[36] ElMundoToday. *Web de ElMundoToday*. Última consulta: 12/04/23. URL: https://www.elmundotoday.com/.

[37] Jefatura del Estado. *Ley de prensa 1966 (BOE-A-1966-3501)*. Última consulta: 24/03/23. URL: https://www.boe.es/buscar/act.php?id=BOE-A-1966-3501&b=4&tn=1&p=19660319#asegundo.

[38] explosion. *Spacy 101*. Última consulta: 17/05/23. URL: https://spacy.io/usage/spacy-101.

[39] explosion. *Spacy Github Repo*. Última consulta: 17/05/23. URL: https://github.com/explosion/spaCy.

[40] Django Software Foundation. *Django at a glance*. Última consulta: 17/05/23. URL: https://docs.djangoproject.com/en/4.2/intro/overview/.

[41] Git. *About Git*. Última consulta: 10/03/23. URL: https://git-scm.com/about.

[42] Github. *About Github*. Última consulta: 10/03/23. URL: https://github.com/about.

[43] Google. *Angular Docs*. Última consulta: 17/05/23. URL: https://angular.io/docs.

[44] Alex Graves. «Generating sequences with recurrent neural networks». En: *arXiv preprint arXiv:1308.0850* (2013).

[45] Gunicorn. *Gunicorn - WSGI server*. Última consulta: 15/06/23. URL: https://docs.gunicorn.org/en/stable/.

[46] Sepp Hochreiter y Jürgen Schmidhuber. «Long Short-term Memory». En: *Neural computation* 9 (dic. de 1997), págs. 1735-80. DOI: 10.1162/neco.1997.9.8.1735.

[47] HuggingFace. *Task guides: Summarization*. Última consulta: 08/02/23. URL: https://huggingface.co/docs/transformers/tasks/summarization.

[48] IBM. *¿Qué son las redes neuronales recurrentes?* Última consulta: 23/05/23. URL: https://www.ibm.com/es-es/topics/recurrent-neural-networks.

[49] IBM. *Natural Language Processing*. Última consulta: 06/02/23. URL: https://www.ibm.com/topics/natural-language-processing.

[50] Instituto de Ingeniería del Conocimiento. *Transformers en Procesamiento del Lenguaje Natural*. Última consulta: 23/05/23. URL: https://www.iic.uam.es/innovacion/transformers-en-procesamiento-del-lenguaje-natural/.

[51] Air Institute. *Proyecto Ethical News*. Última consulta: 30/01/23. URL: https://ethicalnews.air-institute.com/.

[52] Air Institute. *Web de Air Institute*. Última consulta: 13/02/23. URL: https://air-institute.com/.

[53] Peter Jackson e Isabelle Moulinier. *Natural language processing for online applications: Text retrieval, extraction and categorization; Chapter 5*. Vol. 5. John Benjamins Publishing, 2007.

[54] JetBrains. *Documentación Pycharm*. Última consulta: 10/03/23. URL: https://www.jetbrains.com/help/pycharm/quick-start-guide.html.

[55] Jupyter. *About Jupyter*. Última consulta: 10/03/23. URL: https://jupyter.org/about.

[56] Daniel Jurafsky y James H Martin. *Speech and Language Processing: An introduction to speech recognition, computational linguistics and natural language processing, 2nd ed.; Chapter V: Applications*. 2008.

[57] Bing Liu. *Sentiment analysis: Mining opinions, sentiments, and emotions, Chapter 1*. Cambridge university press, 2020.

[58] Yinhan Liu et al. «Roberta: A robustly optimized bert pretraining approach». En: *arXiv preprint arXiv:1907.11692* (2019).

[59] Microsoft. *Documentación Visual Studio Code*. Última consulta: 10/03/23. URL: https://code.visualstudio.com/docs.

[60] R. Mitchell. *Web scraping with Python: Collecting more data from the modern web*. O'Reilly Media, Inc., 2018.

[61] miyakogi. *Pyppeteer Docs*. Última consulta: 12/03/23. URL: https://pyppeteer.github.io/pyppeteer/.

[62] El Mundo. *La infanta Elena pide que la imputen, según dos medios colombianos*. Última consulta: 12/04/23. URL: https://www.elmundo.es/elmundo/2013/04/04/comunicacion/1365072049.html.

[63] Tetsuya Nasukawa y Jeonghee Yi. «Sentiment analysis: Capturing favorability using natural language processing». En: *Proceedings of the 2nd international conference on Knowledge capture*. 2003, págs. 70-77.

[64] NewsAPI. *Documentación de NewsAPI*. Última consulta: 29/01/23. URL: https://newsapi.org/docs.

[65] Dat Quoc Nguyen, Thanh Vu y Anh Tuan Nguyen. «BERTweet: A pre-trained language model for English Tweets». En: *arXiv preprint arXiv:2005.10200* (2020).

[66] Octoparse. *5 anti-scraping techniques you may encounter*. Última consulta: 04/02/23. URL: https://www.octoparse.com/blog/5-anti-scraping-techniques-you-may-encounter.

[67] OpenAI. *Open AI: GPT-3 y ChatGPT*. Última consulta: 08/02/23. URL: https://openai.com/.

[68] Oracle. *What is a chatbot*. Última consulta: 06/02/23. URL: https://www.oracle.com/chatbots/what-is-a-chatbot/.

[69] David Ownby, James R Lewis y Jesper Aagaard Petersen. «The Falun Gong: A new religious movement in post-Mao China». En: *Controversial new religions* (2005), págs. 195-214.

[70] Jinja Pallets. *About Jinja*. Última consulta: 17/05/23. URL: https://jinja.palletsprojects.com/en/3.1.x/intro/.

[71] Werkzeug Pallets. *About Werkzeug*. Última consulta: 17/05/23. URL: https://werkzeug.palletsprojects.com/en/2.3.x/.

[72] Parsehub. *Web Scraping vs Web Crawling*. Última consulta: 06/02/23. URL: https://www.parsehub.com/blog/web-scraping-vs-web-crawling/.

[73] Sociedad Estatal de Participaciones Industriales (SEPI). *Sobre la Agencia EFE*. Última consulta: 24/03/23. URL: https://www.sepi.es/es/sectores/agencia-efe.

[74] Juan Manuel Pérez, Juan Carlos Giudici y Franco Luque. «pysentimiento: A python toolkit for sentiment analysis and socialnlp tasks». En: *arXiv preprint arXiv:2106.09462* (2021).

[75] plightbo et al. *Selenium Docs*. Última consulta: 12/03/23. URL: https://www.selenium.dev/selenium/docs/api/py/index.html.

[76] PostgreSQL. *About PostgreSQL*. Última consulta: 08/06/23. URL: https://www.postgresql.org/about/.

[77] Europa Press. *Web de EuropaPress*. Última consulta: 24/03/23. URL: https://www.europapress.es/.

[78] Mariano Rivera. *El Problema del Gradiente Evanescente*. Última consulta: 23/05/23. URL: http://personal.cimat.mx:8181/~mrivera/cursos/aprendizaje_profundo/evanescente/evanesce_grad.html.

[79] Armin Ronacher. *Opening the Flask: How an April Fools' Joke became a Framework with Good Intentions*. Última consulta: 17/05/23. URL: https://web.archive.org/web/20160604162342/http://mitsuhiko.pocoo.org/flask-pycon-2011.pdf.

[80] Rachel Sanders. *Developing Flask Extensions - PyCon 2014*. Última consulta: 17/05/23. URL: https://www.youtube.com/watch?v=OXN3wuHUBP0#t=46.

[81] Victor Sanh et al. «DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter». En: *arXiv preprint arXiv:1910.01108* (2019).

[82] D. Sarkar, R. Bali y T. Sharma. *Chapter 3, Practical machine learning with python: A problem-solver's guide to building real-world intelligent systems. 1st ed.* APRESS, 2017.

[83] Tokio School. *Redes neuronales recurrentes en Python, ¿cómo funcionan?* Última consulta: 23/05/23. URL: https://www.tokioschool.com/noticias/redes-neuronales-recurrentes/.

[84] De S Sirisuriya et al. «A comparative study on web scraping». En: vol. Proceedings of 8th International Research Conference. Sri Lanka: KDU, 2015.

[85] Statista. *Principales agencias de noticias en función de su facturación en España en 2020*. Última consulta: 24/03/23. URL: https://es.statista.com/estadisticas/961271/ranking-de-las-principales-agencias-de-noticias-espana/.

[86] New York Times. *Documentación de las APIs del periódico New York Times*. Última consulta: 29/01/23. URL: https://developer.nytimes.com/apis.

[87] Vision Times. *Vision Times en español*. Última consulta: 10/04/23. URL: https://es.visiontimes.com/.

[88] Vision Times. *Vision Times: About Us*. Última consulta: 10/04/23. URL: https://www.visiontimes.com/about-us.

[89] Turnitin. *Turnitin y la escritura con IA*. Última consulta: 09/06/23. URL: https://www.turnitin.com/es/soluciones/escritura-con-ia.

[90] Twitter. *About Bootstrap*. Última consulta: 17/05/23. URL: https://getbootstrap.com/docs/4.1/about/overview/.

[91] Ashish Vaswani et al. «Attention is all you need». En: *Advances in neural information processing systems* 30 (2017).

[92] Vocento. *Periódicos Grupo Vocento*. Última consulta: 24/03/23. URL: https://www.vocento.com/nosotros/prensa.

[93] Wikipedia. *Angular (web framework)*. Última consulta: 17/05/23. URL: https://en.wikipedia.org/wiki/Angular_(web_framework).

[94] Wikipedia. *Web scraping techniques*. Última consulta: 04/02/23. URL: https://en.wikipedia.org/wiki/Web_scraping#Techniques.

[95] Simon Willison. *What is the history of the Django web framework? Why has it been described as 'developed in a newsroom'?* Última consulta: 17/05/23. URL: https://www.quora.com/What-is-the-history-of-the-Django-web-framework-Why-has-it-been-described-as-developed-in-a-newsroom/answer/Simon-Willison.

[96] Thomas Wood. *What is a Transformer Neural Network?* Última consulta: 23/05/23. URL: https://deepai.org/machine-learning-glossary-and-terms/transformer-neural-network.

Glosario

A

API

Son las siglas de *Application Programmming Interface* (Interfaz de Programación de Aplicaciones en castellano), es un mecanismo que permite a dos componentes de sofware comunicarse entre sí mediante un conjunto de definiciones y protocolos. 4

C

Computer Vision

Es un campo de la inteligencia artificial que permite a los ordenadores extraer información a partir de imágnes digitales. 21

D

Deep Learning

Es un subcampo del aprendizaje automático (Machine Learning) que se centra en el desarrollo y uso de redes neuronales artificiales. Estas redes neuronales son capaces de aprender automáticamente representaciones jerárquicas de datos complejos a través de la experiencia y los datos de entrenamiento. Algunas de las arquitecturas más comunes en el Deep Learning. 27, 68, 104

Deepfake

Es una forma de manipulación de medios digitales mediante técnicas de inteligencia artificial para crear contenidos multimedia falsificados o falsos. Se usa en vídeos, imágenes, audio y texto. 38

E

ETL

Son las siglas de *extract, transform, load* (extraer, transformar, cargar en castellano). Son una serie de procesos que, al ser usados, permiten obtener información de varias fuentes o bases de datos y permite que se almacene en un solo repositorio con los datos debidamente formateados y listos para ser utilizados. 5

F

fine-tuning

El fine-tuning es un proceso del aprendizaje automático en el cual se toma un modelo pre-entrenado y se ajusta para que realice una tarea específica, permitiendo aprovechar los patrones y características ya aprendidos por el modelo y obtener un mejor rendimiento que con un modelo entrenado desde cero. 84

I

IA generativa (Inteligencia Artificial generativa)

Es un subcampo de la inteligencia artificial que se centra en la creación de contenido. Se usan algoritmos y técnicas de aprendizaje automático, especialmente Deep Learning, para generar contenido como texto, imágenes, música, voz y otros tipos de medios. 84

IDE

Integrated Development Environment o Entorno de Desarrollo Integrado en español. Es una herramienta usada para el desarrollo de sofware que incluye funciones para escribir, depurar y ejecución de código de forma más rápida y sencilla para el usuario. 61

M

Markdown

Markdown es un lenguaje de marcado ligero utilizado para formatear texto de una manera fácil y rápida. Utiliza caracteres especiales para añadir estilo al texto. 62

N

NLP

Natural Language Processing o Procesamiento del Lenguaje Natural (PLN). 24, 26, 27, 30, 67, 68

P

Parse

Es el proceso de analizar y convertir un programa a un formato interno que el entorno de desarrollo pueda ejecutar. 20

Pipelines

Son una estructura en la que se aplican diferentes procesos a una serie de etapas de procesamiento de datos que se conectan en secuencia, donde la salida de una etapa se convierte en la entrada de la siguiente etapa. El concepto de pipeline se basa en

el principio ”Divide y Vencerás” (divide-and-conquer) que es la idea de dividir un proceso complejo en subprocesos más simples y administrables, que se ejecutan de manera secuencial o en paralelo. 68

S

SEO (Search Engine Optimization)

Es el proceso de mejorar la visibilidad de un sitio web en los resultados de un buscador o motor de búsqueda. 80

Softmax

La función softmax es una función matemática que transforma un conjunto de números en probabilidades. Cada número se convierte en una probabilidad, dando la suma de todas las probabilidades. Es usada a menudo en problemas de aprendizaje automático. Su fórmula matemática es $softmax(x)_i = \frac{e^(x_i)}{\sum_j e^(x_j)}$. 30, 32

Stopwords

Son un conjunto de palabras que carecen de sentido al aparecer solas. También se suelen considerar stopwords a conjunciones, artículos, preposiciones y adverbios. 24

U

User Agent (Agente de Usuario)

Un *User Agent* (Agente de Usuario) es una cadena de texto que los clientes web envían a los servidores para identificar el *software* y el *hardware* que usa el usuario como por ejemplo el nombre y versión del navegador y sistema operativo entre otros. 73