TRABAJOS FIN DE GRADO

Universidad de Jaén

Escuela Politécnica
Superior de Jaén

Recuperación y Ampliación Automática de Noticias con Tecnologías de PLN

Autor: Miguel Montoya Gavilán

Grado en Ingeniería Informática

Director: Prof. D. Manuel García Vega; Subdirector: Prof. D. Eugenio Martínez Cámarara
Departamento del director: Departamento de Informática

Fecha: Septiembre, 2024

AGRADECIMIENTOS

Para este apartado de agradecimientos me gustaría agradecer en primer lugar a mis padres y a mi hermano, me han aportado y ayudado mucho en esta etapa de 4 años, sobretodo cuando me daban el día libre de TFG para ayudar en el trabajo.

Agradecer a mis abuelos, que son unos máquinas, no entienden mucho lo que estoy haciendo, pero saben que le puedo poner el fútbol o configurarle el móvil en un santiamén.

Dar las gracias también, como no, a mis tutores de este proyecto, que sin ellos nada hubiera sido posible, Manolo y Eugenio, por haberme ayudado y guiado en el desarrollo de este proyecto y por esas risas en las tutorías que teníamos. Gracias Manolo por haber sido mi tutor durante estos años de carrera, cualquier duda o problema que me surgía acudía a ti y tu me ayudabas, gracias.

Me gustaría dar las gracias también a Rafa, mi compañero de estudios y mi mejor amigo, por todas esas noches que hemos estudiado mucho, y las que no tanto también.

Por último dar gracias a Celia, mi compañera de vida, por haber estado conmigo en los buenos y en los malos momentos. Me has ayudado más de lo que tú te crees, en nada te estoy viendo graduarte, te quiero guapa.

Bueno, una vez dados los agradecimientos, vamos a la chicha.

Índice

Capítulo 1

Introducción y Estructura del Proyecto

1.1. Introducción

En un contexto donde la cantidad de información disponible crece de manera exponencial, la habilidad para recopilar, organizar y analizar datos se ha vuelto esencial. Este proyecto se propone desarrollar un dossier de noticias enfocado en "La Universidad de Jaén"(UJA), con el objetivo de centralizar y estructurar la información relevante sobre esta institución.

El propósito del dossier es ofrecer una herramienta efectiva para cualquier persona interesada en la UJA, facilitando el acceso a noticias publicadas sobre la universidad en una amplia gama de fuentes en línea.

Lo que distingue a este proyecto no es solo la recopilación de información de diversas fuentes, sino la implementación de técnicas avanzadas de Procesamiento del Lenguaje Natural (PLN) y modelos de aprendizaje automático para clasificar y jerarquizar la información. Esto permitirá que los datos presentados sean no solo relevantes, sino también organizados y fácilmente accesibles.

La información recopilada se integrará en una aplicación web con una interfaz de usuario intuitiva y fácil de usar. La interfaz estará diseñada para ser .ªmigable", inspirándose en el estilo visual de la página oficial de la Universidad de Jaén, y ofrecerá opciones de filtrado para que el usuario pueda explorar las noticias de su interés de manera eficiente.

1.2. Motivación

La motivación principal detrás de este proyecto es la necesidad de contar con un sistema que permita a nuestra comunidad universitaria y a cualquier persona interesada en la Universidad de Jaén mantenerse informados sobre sus acontecimiento mas relevantes.

En la actualidad, si algún miembro de nuestra universidad quisiera estar al tanto de los eventos que ocurren en ésta tendría que buscar por múltiples fuentes en línea y sobre temas muy variados.

Lo que se busca con este proyecto es ahorrarle ese tiempo a este prototipo de usuario recolectando noticias de las fuentes en línea más relevantes de nuestra provincia, de forma que se recopile la mayor cantidad de noticias fiables y de calidad para que éstas estén reunidas en un mismo espacio, y además, la tarea encontrar las noticias de interés sea mucho más fácil.

1.3. Objetivos

Se describen a continuación los objetivos del proyecto:

- **Desarrollo del Sistema de Recolección de Noticias:** Implementar un sistema automático que sea capaz de recopilar noticias relacionadas con la Universidad de Jaén desde diversas fuentes en línea.
- **Clasificación Automática de Noticias:** Utilizar técnicas basadas en PLN y modelos de aprendizaje automático para clasificar las noticias en diferentes categorías
- **Interfaz de Usuario Intuitiva:** Crear una interfaz de usuario que sea fácil de usar, permitiendo a los usuarios navegar de forma intuitiva y en un entorno cómodo.
- **Implementación de un filtrado de Noticias:** Crear un sistema de filtrado donde el usuario pueda filtrar por los aspectos de su interés y de esta forma encontrar noticias más facilmente.

1.4. Hipótesis

Se plantea que la implementación y desarrollo de un sistema automatizado de recolección, búsqueda y clasificación de noticias tendrá los siguientes efectos:

- **Eficiencia en la Recolección de Datos:** Un sistema automatizado será capaz de recopilar noticias de manera más rápida y exhaustiva que un proceso manual.
- **Precisión y Relevancia en la Clasificación de Noticias:** Las noticias clasificadas mediante modelos de aprendizaje y técnicas de PLN pueden clasificar con un gran grado de precisión, identificando patrones y tendencias relevantes.
- **Usabilidad y Satisfacción del Usuario:** Una interfaz de usuario bien diseñada y fácil de usar incrementará la satisfacción y eficiencia de los usuarios al interactuar con el sistema.

- **Robustez y Escalabilidad del Sistema:** El sistema será capaz de manejar grandes volúmenes de datos y adaptarse a cambios en las estructuras de los sitios web fuente.

1.5. Estructura

La estructura del proyecto consta de la siguiente manera:

- **Capítulo 2: Fase de Análisis**
  En este capítulo se realiza un análisis detallado de los requisitos del sistema. Se identifican las necesidades de los usuarios y se definen las funciones clave que el sistema debe cumplir para resolver el problema planteado.

- **Capítulo 3: Contexto relacionado con el problema a resolver**
  Este capítulo ofrece un contexto teórico y técnico relacionado con el problema a resolver. Aquí se introduce el Procesamiento de Lenguaje Natural (PLN) y sus aplicaciones. Además, se abordan técnicas como la clasificación de temas (topic modeling) y el scraping, que son fundamentales para el desarrollo del sistema.

- **Capítulo 4: Fase de Diseño**
  En este capítulo se describe el diseño del sistema basado en el análisis previo. Se detalla la arquitectura del sistema así como el diagrama de despliegue y el diseño de la base de datos.

- **Capítulo 5: Marco de Evaluación de la Clasificación**
  Este capítulo se centra en la evaluación del sistema de clasificación desarrollado. Se establecen los criterios de evaluación, las métricas a utilizar, y se realiza un análisis de los resultados obtenidos en las pruebas del sistema.

- **Capítulo 6: Implementación**
  Aquí se describe el proceso de implementación del sistema, detallando las herramientas y tecnologías utilizadas, los pasos seguidos para construir el sistema, y los desafíos encontrados durante la implementación.

- **Capítulo 7: Conclusiones**
  En el capítulo final, se presentan las conclusiones del proyecto, se resume el trabajo realizado, y se discuten los logros alcanzados.

De esta forma queda detallada la estructura de la memoria, proporcionando una descripción clara y concisa de cada capítulo que compone este proyecto.

Con esto, se sienta la base para los capítulos siguientes, donde se profundizará en cada una de las fases del proyecto, comenzando por fase de análisis en el siguiente capítulo.

Capítulo 2

Fase de Análisis

2.1. Requisitos funcionales del proyecto

2.1.1. Introducción

En esta sección, se detallan los requisitos funcionales que definen el comportamiento y las funcionalidades que deberá cumplir el sistema de dosier de noticias. Estos requisitos son fundamentales para garantizar que el sistema pueda recopilar, clasificar y filtrar noticias de manera eficiente, permitiendo a los usuarios acceder rápidamente a la información relevante sobre la Universidad de Jaén. La correcta implementación de estos requisitos permitirá desarrollar un sistema robusto, capaz de manejar grandes volúmenes de datos y proporcionar una experiencia de usuario satisfactoria.

2.1.2. Requisitos funcionales

- El sistema debe ser capaz de recopilar noticias automáticamente desde múltiples fuentes en línea, extrayendo y almacenando información relevante como el título, autor, fecha y contenido de las noticias.

- El sistema debe permitir a los usuarios filtrar noticias almacenadas, ofreciendo opciones de filtrado por fecha, categoría, y relevancia para facilitar el acceso a la información más pertinente.

- El sistema debe mostrar detalles completos de una noticia seleccionada por el usuario, incluyendo su título, fecha, autor, y contenido, de manera clara y organizada.

- El sistema debe poder clasificar automáticamente las noticias en categorías relevantes utilizando técnicas de Procesamiento de Lenguaje Natural. Esto incluye el uso de

un modelo de aprendizaje no supervisado que permita identificar patrones y mejorar continuamente la precisión en la clasificación.

2.1.3. Requisitos no funcionales

- El sistema debe ser capaz de manejar grandes volúmenes de datos y realizar operaciones de scraping, procesamiento y búsqueda de manera eficiente, sin comprometer la velocidad o la calidad de los resultados, ya que la información que tenemos en la web (periódicos) es inmensa.
- Se debe optimizar el rendimiento del sistema para garantizar tiempos de respuesta rápidos, previendo períodos de alta carga de trabajo.
- El sistema debe ser robusto y resistente a fallos, capaz de manejar situaciones inesperadas, como cambios en la estructura del sitio web (diferencia de estructura entre periódicos).
- La interfaz de usuario debe ser intuitiva y fácil de usar, con opciones claras para visualizar las noticias y acceder a funciones adicionales del sistema, es decir, poder acceder al sistema de filtrado de noticias junto con la consulta.

2.2. Casos de uso

2.2.1. Introducción

Se presentan los casos de uso del proyecto, destacando las interacciones clave entre los usuarios y el sistema, así como los resultados esperados de dichas interacciones.

Se mostrarán, además, los respectivos **diagramas de casos de uso** para un mejor entendimiento de los casos de uso del sistema.

El diagrama está compuesto por los siguientes elementos:

- **Actores:** Representados como figuras humanas (usuarios o sistemas externos) que interactúan con el sistema.
- **Casos de uso:** Representados por óvalos, cada uno de los cuales indica una funcionalidad específica del sistema.
- **Relaciones:** Las relaciones entre los actores y los casos de uso están representadas por líneas. Las relaciones de inclusión («Include») y extensión («Extend») están indicadas donde es relevante.

Los casos de uso que compondrán el sistema serán los siguientes:

- Recopilar Noticias
- Filtrar Noticias
- Visualizar Detalles de una Noticia
- Clasificar Noticias

Para comenzar este apartado se mostrará un diagrama de casos de uso general, donde se mostrarán los actores que interaccionan con estas funcionalidades.

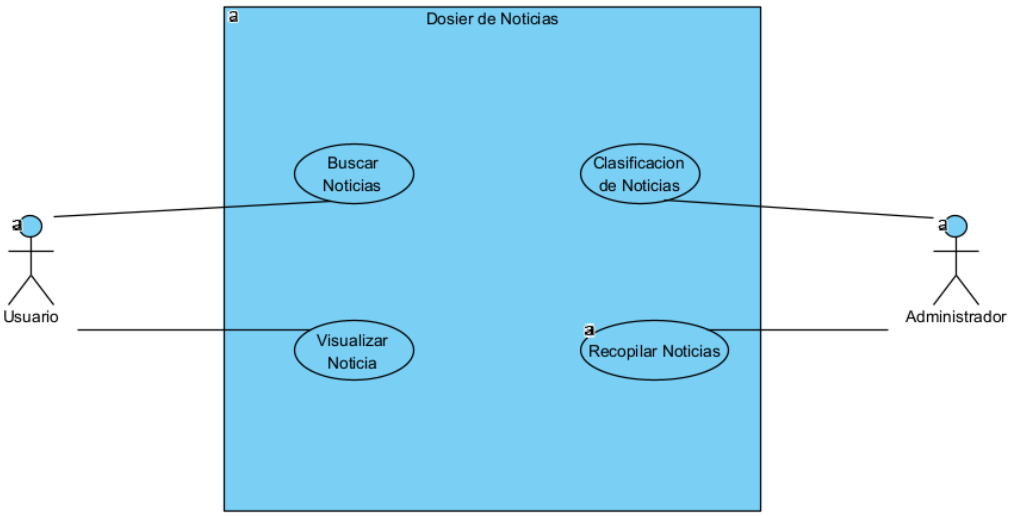

Figura 2.1: *Diagrama General*

Como se puede observar hay dos actores en el escenario, el Usuario y el Administrador. El primer actor, el Usuario interactúa con los siguientes caso de uso:

- Filtrar Noticias
- Visualizar Detalles de una Noticia

Mientras que el segundo actor, el Administrador, interactúa con los siguientes:

- Recopilar Noticias
- Clasificar Noticias

A continuación, se desarrollara cada caso de uso expuesto.

2.2.2. Recopilar noticias

- **Actor primario:** Sistema
- **Sistema:** Dosier de Noticias
- **Nivel:** Subfunción

Contexto: Para poder mostrar al usuario las noticias que son de su interés primero es necesario recopilarlas de diversas fuentes. Este caso de uso se encarga de recorrer varios periódicos recopilando las noticias.

Descripción del Caso de Uso:

- **Inicio:** Se inicia el script que se encarga de recopilar noticias.
- **Acción Principal:** El script recorre todo el periódico descargando el código HTML de todas sus páginas.
- **Procesamiento:** El script se encarga de extraer la información necesaria de cada código HTML para luego almacenarla.

Objetivo del Usuario: Tener todas las noticias posibles almacenadas.

Interacciones clave:

1. El script descarga el contenido de las páginas de cada periódico.
2. Se procesa dicho código almacenando la información importante.

Resultados Esperados: Varios archivos en formato CSV con toda la información de cada periódico, un archivo para cada periódico.

El diagrama de caso de uso sería el siguiente:

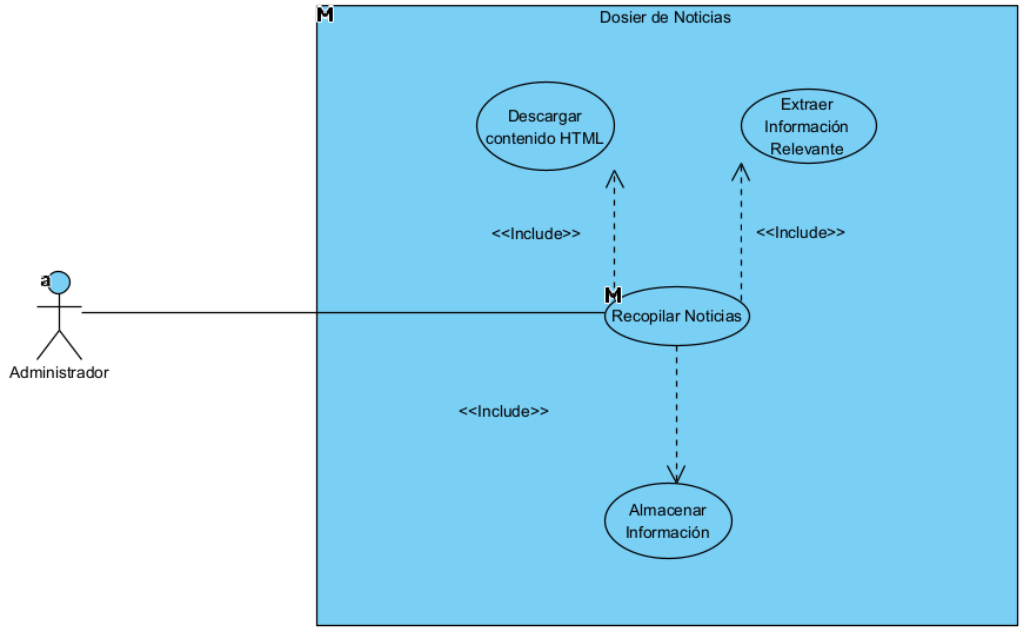

Figura 2.2: *Diagrama Recopilar Noticias*

Este diagrama muestra como el administrador interactúa con el caso de uso 'Recopilar Noticias'. Se relaciona con otras tres funcionalidades del sistema mediante la relación «Include», estas funcionalidades son:

- Descargar contenido HTML.
- Extraer Información Relevante.
- Almacenar Información.

2.2.3. Filtrar Noticias

- **Actor primario:** Usuario
- **Sistema:** Dosier de Noticias
- **Nivel:** Objetivo Usuario

Contexto: En el sistema, los usuarios necesitan buscar noticias relevantes para mantenerse informados sobre temas específicos. Este caso de uso permite a los usuarios encontrar rápidamente noticias relacionadas con sus intereses.

Descripción del Caso de Uso:

- **Inicio:** El usuario accede al sistema a través de su interfaz.
- **Acción Principal:** El usuario realiza un filtrado para buscar noticias relevantes.
- **Procesamiento:** El sistema recibe la información del filtrado y la procesa mostrando las noticias relevantes.

Objetivo del Usuario: Encontrar noticias relevantes de manera rápida y precisa.

Interacciones clave:

1. El usuario abre el sistema y accede a la función de búsqueda.
2. Filtra las noticias según sus intereses.
3. El sistema procesa y muestra los resultados en una lista ordenada.

Resultados Esperados: Una lista de noticias relevantes basada en el filtrado proporcionado por el usuario, la lista está ordenada por relevancia.

El diagrama de caso de uso sería el siguiente:

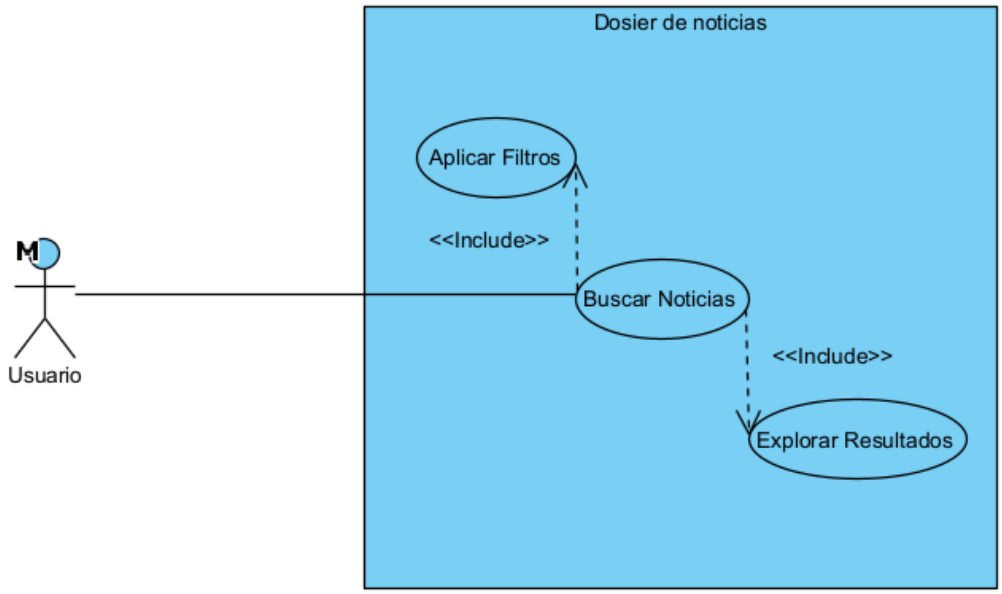

Figura 2.3: *Diagrama Filtrar Noticias*

En este diagrama el actor principal es el Usuario, se relaciona con el caso de uso 'Buscar Noticias'. Se añaden también otras funcionalidades mediante «Include» como:

- Aplicar Filtros.
- Explorar Resultados.

2.2.4. Visualizar Detalles de una Noticia

- **Actor primario:** Usuario
- **Sistema:** Dosier de Noticias
- **Nivel:** Objetivo Usuario

Contexto: Una vez que el usuario encuentra una noticia de interés, necesita acceder a los detalles completos para obtener información completa sobre el evento o tema.

Descripción del Caso de Uso:

- **Inicio:** El usuario ha encontrado una noticia de interés en la lista de resultados.
- **Acción Principal:** El usuario selecciona la noticia para ver más detalles.
- **Procesamiento:** El sistema muestra el contenido completo de la noticia seleccionada.

Objetivo del Usuario: Acceder a la información detallada de una noticia específica previamente elegida.

Interacciones clave:

1. El usuario encuentra una noticia de interés en los resultados de búsqueda.
2. Selecciona la noticia para visualizarla en detalle.
3. El sistema muestra el contenido completo de la noticia.

Resultados Esperados: La visualización completa de la noticia seleccionada, con su información detallada y relevante.

El diagrama de caso de uso sería el siguiente:

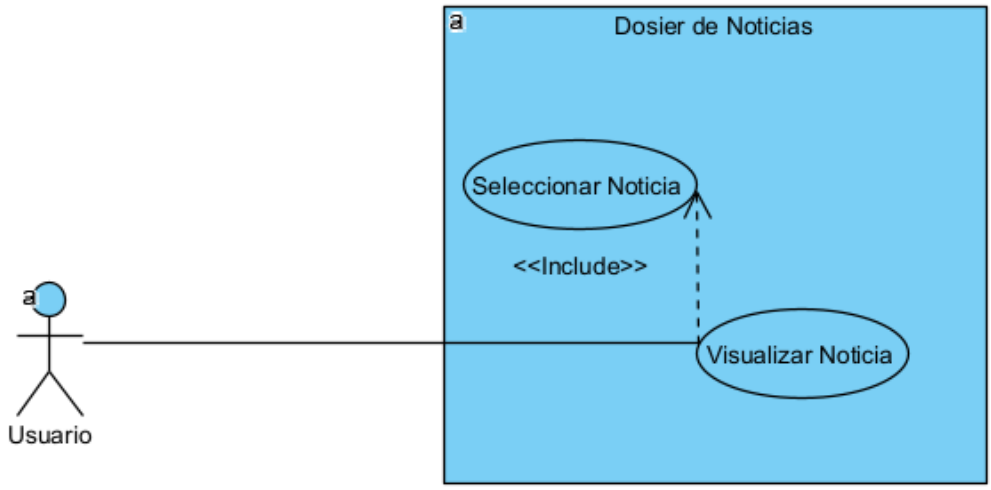

Figura 2.4: *Diagrama Visualizar Detalles de una Noticia*

En este diagrama también actúa el Usuario relacionándose con Visualizar Noticia, este es un diagrama de casos de uso muy simple ya que con la única funcionalidad con la que se relaciona mediante «Include» es 'Seleccionar Noticia'.

2.2.5. Clasificar Noticias

- **Actor primario:** Sistema
- **Sistema:** Dosier de Noticias
- **Nivel:** Subfunción

Contexto: El sistema debe organizar automáticamente las noticias recopiladas para facilitar la búsqueda y el acceso a la información. En este caso de uso se describe cómo el sistema clasifica las noticias.

Descripción del Caso de Uso:

- **Inicio:** El sistema recopila noticias de diversas fuentes.
- **Acción Principal:** El sistema analiza y clasifica automáticamente las noticias.

- **Procesamiento:** Aplica un modelo de aprendizaje no supervisado para clasificar las noticias recopiladas.

Objetivo del Usuario: Clasificar automáticamente las noticias para mejorar la organización y accesibilidad de la información.

Interacciones clave:

1. Se aplica técnicas de PLN para preprocesar el texto de las noticias obtenidas.
2. Se utiliza un modelo de aprendizaje no supervisado para identificar patrones y clasificar un conjunto inicial de noticias.

Resultados Esperados: Las noticias clasificadas automáticamente en diferentes categorías o etiquetas, mejorando la organización y acceso a la información.

El diagrama de caso de uso sería el siguiente:

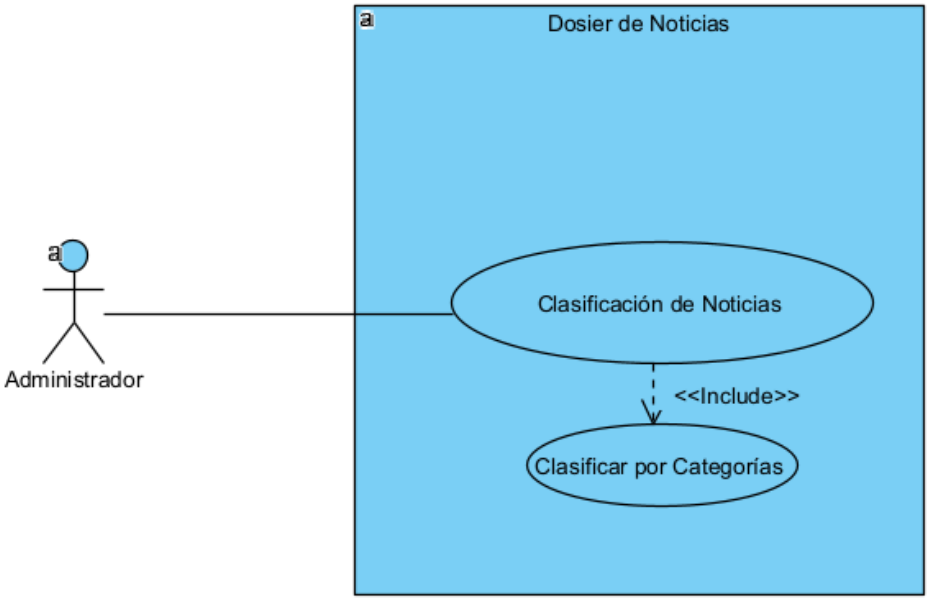

Figura 2.5: *Diagrama Clasificar Noticias*

En este diagrama se puede observar como el actor principal Administrador se relaciona con el caso de uso 'Clasificación de Noticias'.

El diagrama, además, se compone de dos relaciones «Include» y una «Extend», las dos relaciones de «Include» son con las funcionalidades 'Aplicar Técnicas de PLN' y 'Clasificar por categorías' mientras que la funcionalidad 'Analizar Noticias' se relaciona mediante «Extend».

2.3. Interfaz

Se procede con el apartado de la interfaz que expondrá el Dosier de Noticias, dicho sistema constará de dos interfaces principales:

- La primera interfaz será la **página de inicio** donde el usuario tendrá un formulario de filtrado el cual podrá rellenar a su gusto y una vez rellenado se le mostrarán las noticias recuperadas en base a la información introducida por el usuario.

  La información que se mostrará de las noticias recopiladas será escueta, ya que si el usuario tiene interés en alguna noticia se le podrá hacer clic para redirigirlo a la siguiente interfaz.

- Esta interfaz es la encargada de mostrar todos los **detalles de la noticia** que se han recopilado, así como el periódico recopilado, título de la noticia, fecha etc...

  Dispondrá además de un botón que le permite al usuario volver a la interfaz previamente mencionada para mejorar la experiencia de usuario.

Como se puede entender, las interfaces serán muy sencillas, lo que hace que mejore la experiencia de usuario haciendo que la interfaz sea muy intuitiva y sencilla de usar.

Se ha seleccionado una paleta de colores que refleja la identidad visual de la Universidad de Jaén. Esta decisión tiene como objetivo crear una experiencia coherente y familiar para el usuario, alineada con la imagen institucional que ya conocen sobre esta entidad. Al utilizar los colores corporativos oficiales de la universidad, se busca que los usuarios se sientan en un entorno visualmente consistente con el de la página web oficial de la Universidad de Jaén.

En el desarrollo de la interfaz los sketchs y wireframes juegan un papel fundamental en la planificación y conceptualización del diseño por lo que para un mejor entendimiento se procede a exponer los sketchs y wireframes de cada una de las dos interfaces expuestas, de forma que se pueda crear una idea general de como será el diseño final del Sistema.

2.3.1. Sketch Página Inicio

Se expone a continuación el sketch realizado para un mejor entendimiendo inicial de la página de incio:

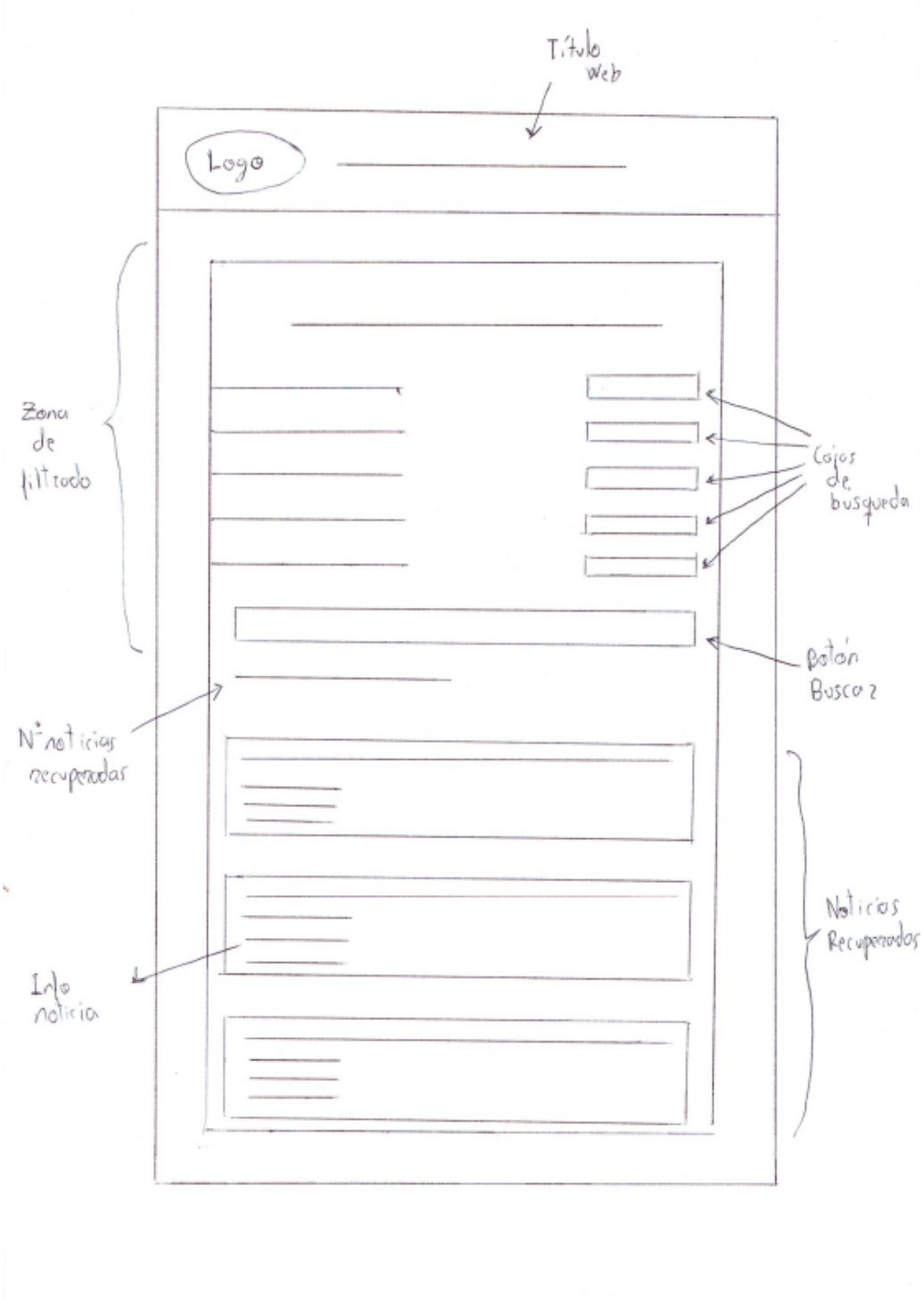

Figura 2.6: *Sketch Página Inicio*

Como se puede observar quedan bien diferenciadas las partes del encabezado, zona de filtrado y zona de noticias recuperadas lo que ayuda a mejorar la experiencia de usuario.

2.3.2. Wireframe Página Inicio

Se procede con el diseño del wireframe el cual tiene como objetivo diseñar los espacios entre los diferentes elementos de la interfaz y las funcionalidades principales del sistema.

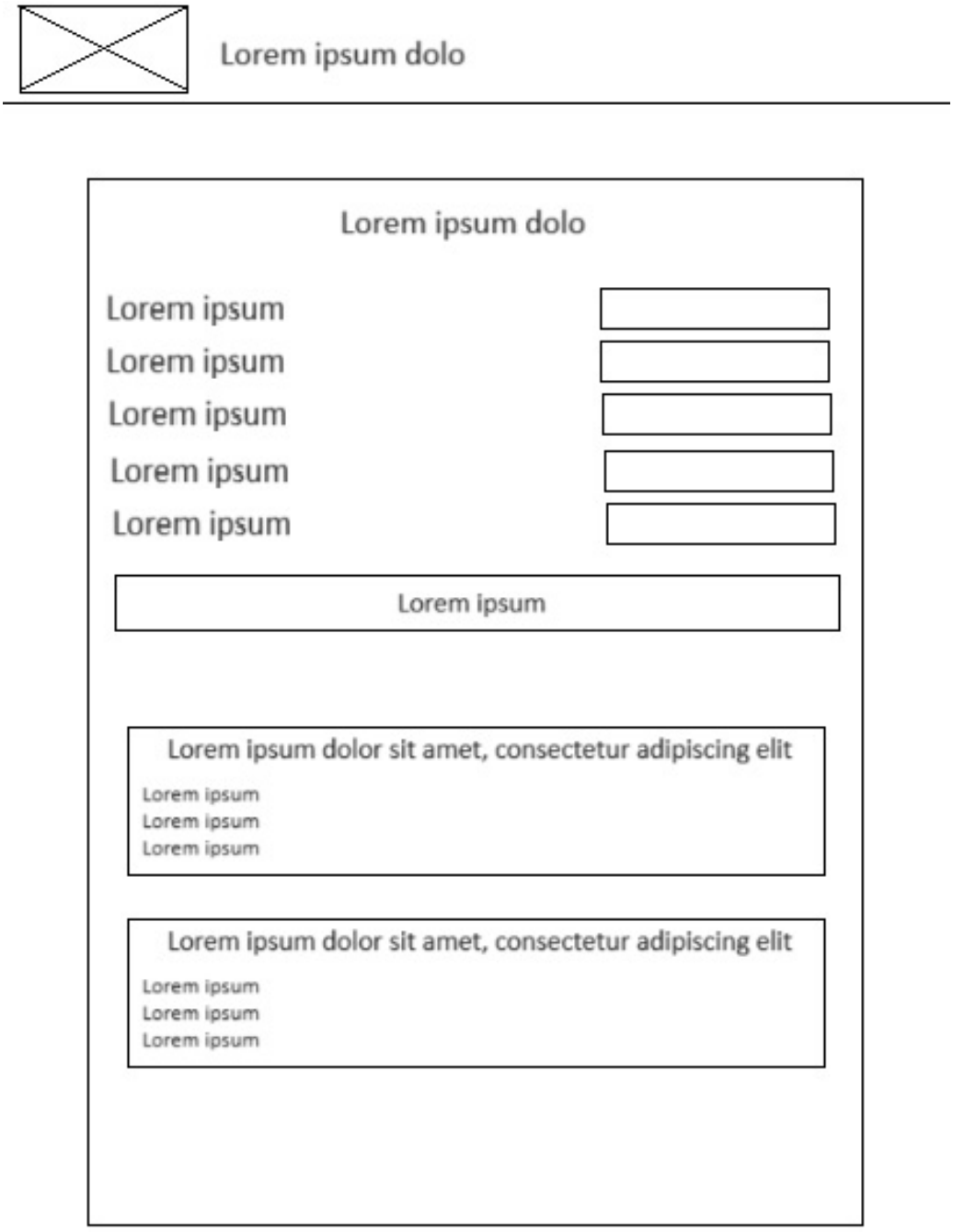

Figura 2.7: *Wireframe Página Inicio*

El diseño es similiar a creado en el sketch pero con un nivel de detalle mayor.

2.3.3. Sketch Detalles Noticia

Se procede ahora con el diseño del sketch de la página que muestra los detalles de la noticia.

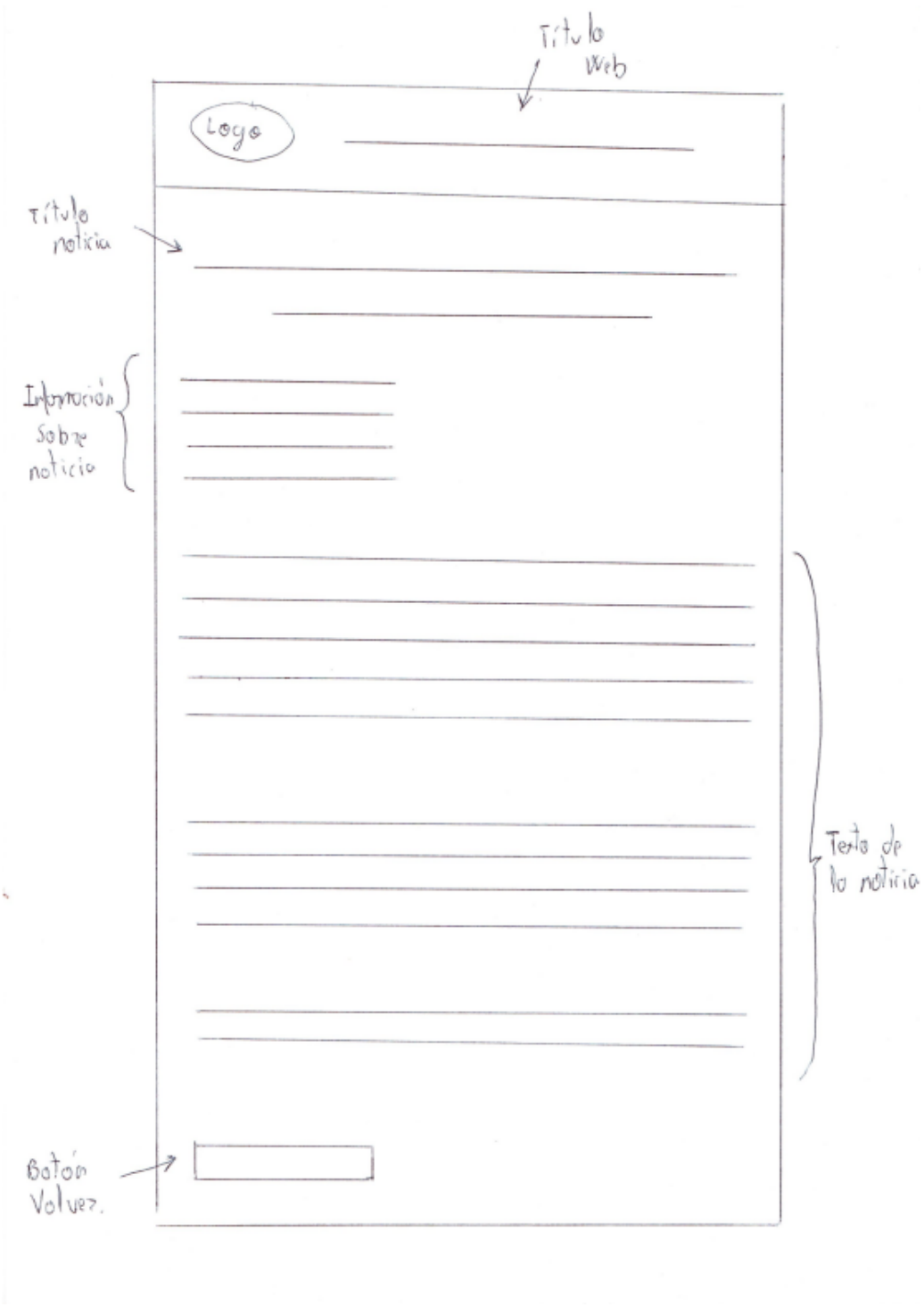

Figura 2.8: *Sketch Detalles Noticia*

Esta página web es mucho más sencilla en cuanto a diseño ya que solo se muestran los detalles de las noticias y un botón para volver a la página principal.

2.3.4. Wireframe Detalles Noticia

El diseño del wireframe es el siguiente:

Figura 2.9: *Wireframe Detalles Noticia*

De esta forma se han realizado los primeros diseños para la creación final de la interfaz de usuario del Sistema.

Una vez terminados los bocetos de cada una de las páginas se da por finalizada la fase de desarrollo de la interfaz mostrando a continuación la interfaz web final que tendrá el programa:

2.3.5. Interfaz Final Página Inicio

La interfaz final de la página de Inicio es la siguiente:

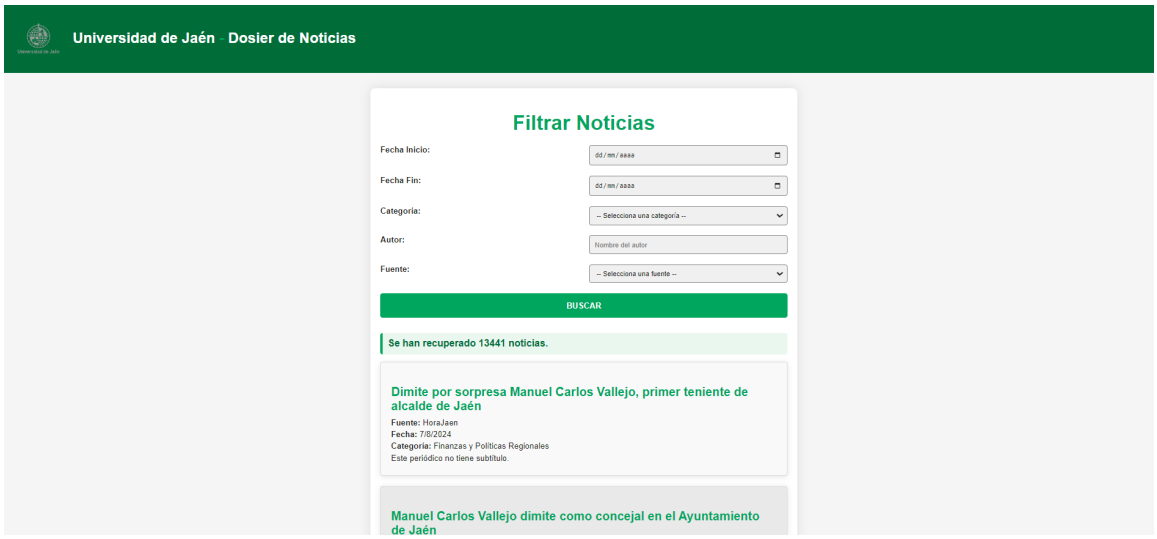

Figura 2.10: *Interfaz Final Página Inicio*

Como se puede observar, el sistema permite poder filtrar las noticias para poder recuperar las noticias a gusto del usuario, por lo que se haced referencia a al requisito funcional que lo especifica en el siguiente ***requisito***..

2.3.6. Interfaz Final Detalles Noticia

La interfaz final de la página en la que se muestran los detalles de las noticias es la siguiente:

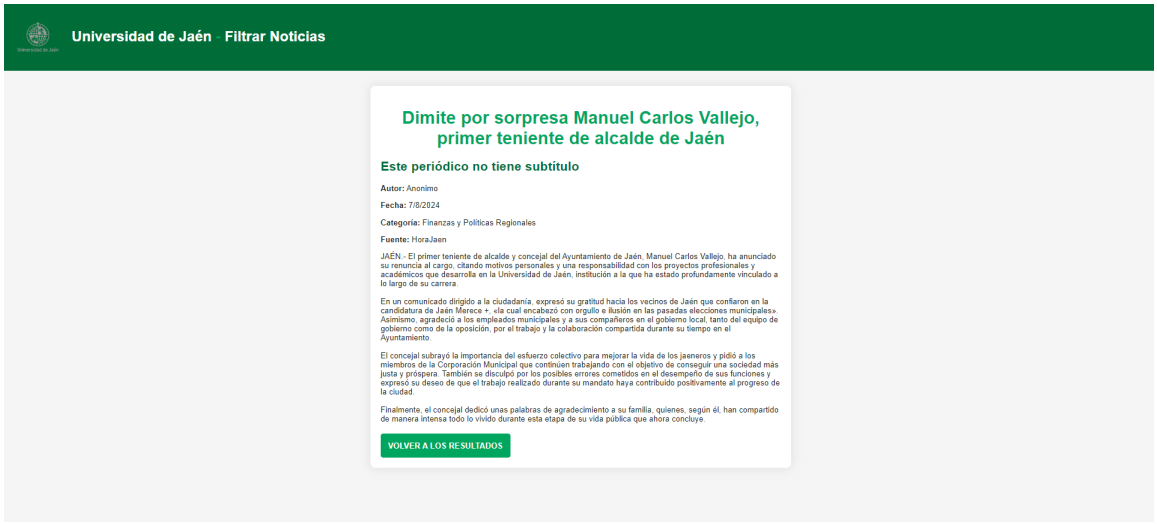

Figura 2.11: *Interfaz Final Detalles Noticia*

La implementación se ha hecho utilizando HTML y CSS, se ha elegido estos lenguajes de programación ya que son bastante fáciles de usar y ofrecen muchas herramientas visuales para adaptar la página web a como se desea.

Al diseñar la interfaz de noticias para la Universidad de Jaén, uno de los aspectos clave fue la selección del formato y la paleta de colores. Estos elementos no solo determinan la estética de la plataforma, sino que también juegan un papel fundamental en la experiencia del usuario, influyendo en su comodidad y familiaridad al interactuar con el sitio.

El formato elegido para la interfaz como se puede ver sigue las directrices visuales ya establecidas por la Universidad de Jaén en su página web. Al utilizar un formato que refleja el estilo visual oficial de la universidad, se consigue una transición más natural para los usuarios que están acostumbrados a navegar por dicha web oficial. Esta consistencia no solo refuerza la marca institucional, sino que también mejora la usabilidad al proporcionar un entorno familiar al usuario.

La paleta de colores utilizada, como el formato, se basa en los colores institucionales de la Universidad de Jaén. La elección de estos colores, no solo responden a cuestiones estéticas, sino también a consideraciones psicológicas que influyen en la percepción y el comportamiento del usuario en la web al sentirse muy familiarizado con la interfaz web.

Toda esta información se ha sacado de un recurso oficial de la Universidad de Jaén, dicho recurso proporciona la paleta de colores que se utiliza en la web oficial así como unas cuantas instrucciones a tener en cuenta.

La adopción del formato y la paleta de colores de la Universidad de Jaén no es solo una cuestión de estilo, sino una estrategia consciente para mejorar la experiencia del usuario. Al replicar el entorno visual con el que los estudiantes, profesores y demás miembros de la comunidad universitaria ya están familiarizados, se crea un sentido de pertenencia.

2.4. Diagrama de secuencias

2.4.1. Introducción

En esta sección, se presentan los diagramas de estado del proyecto. Cada diagrama ilustra los diferentes estados de un objeto, así como las transacciones que pueden ocurrir debido a algún evento en específico.

A continuación se muestran los diagramas de estado correspondientes a los principales escenarios del sistema:

- Recopilar Noticias
- Realizar Búsqueda de Noticias
- Visualizar Detalles de una Noticia
- Clasificar Noticias

2.4.2. Recopilar Noticias

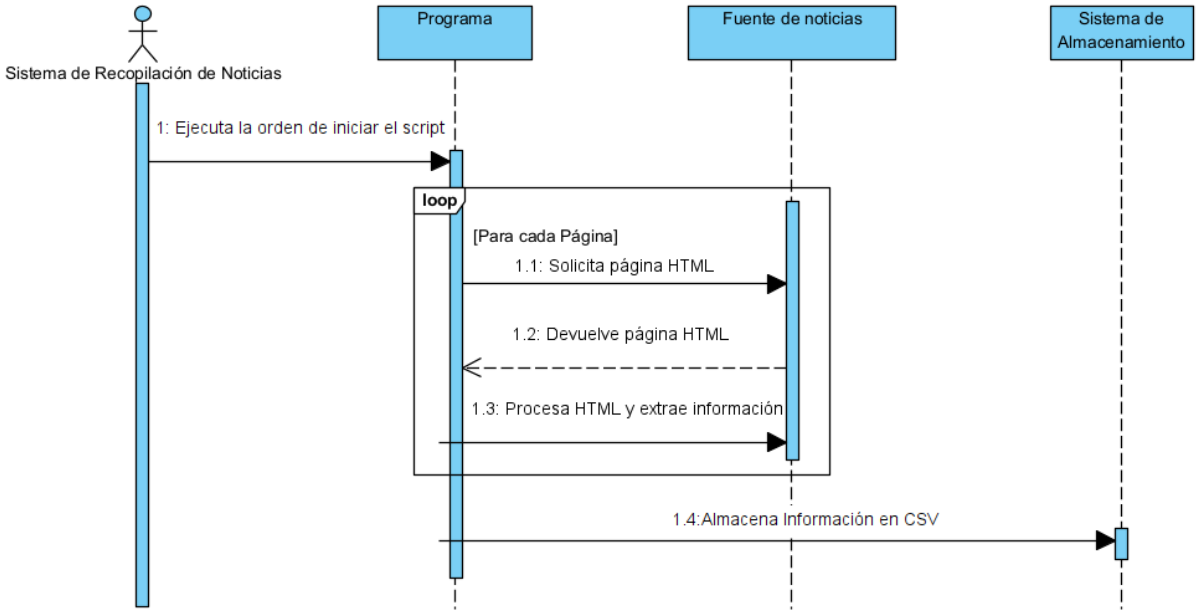

Figura 2.12: *Imagen recopilación de noticias*

En este diagrama se puede observar como es una subfunción del sistema ya que no interviene el usuario en este escenario. Simplemente el sistema de recolección inicia el script de recolección y este se encarga de recolectar las noticias para más tarde almacenarlas.

Para recolectar las noticias se mete en un bucle que recorre todo la fuente de noticias página por página.

2.4.3. Filtrar Noticias

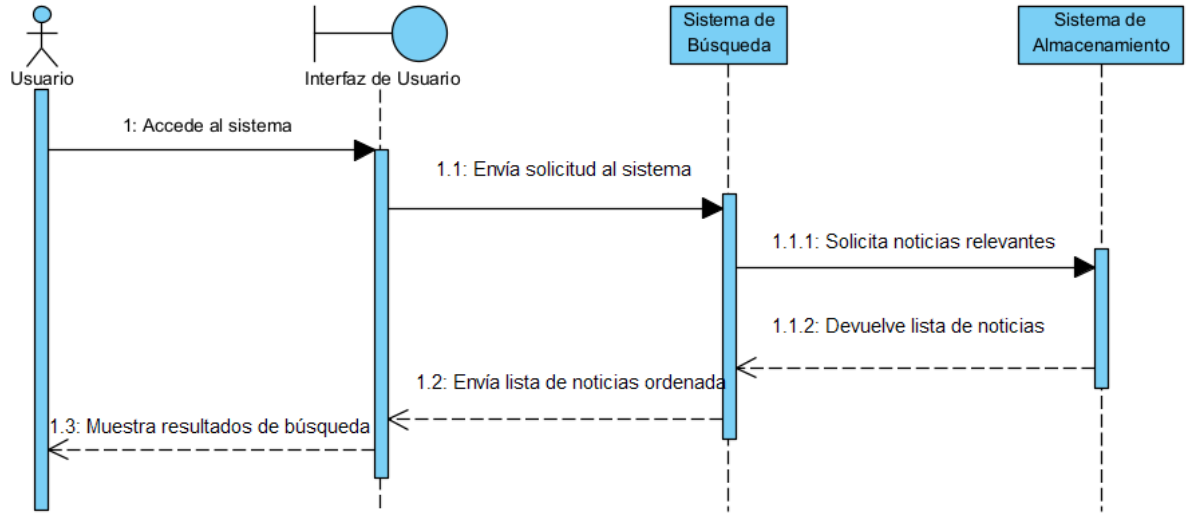

Figura 2.13: *Imagen Filtrar Noticias*

En este diagrama de secuencias si que interviene el usuario y como se puede observar envía el formulario rellenado con los filtros de búsqueda de la interfaz al sistema de búsqueda, este sistema se comunica con la base de datos y recupera de dicha base las noticias relevantes para los filtros que ha introducido el usuario.

Una vez se tienen las noticias recuperadas se muestra en pantalla al usuario.

2.4.4. Visualizar Detalles de una Noticia

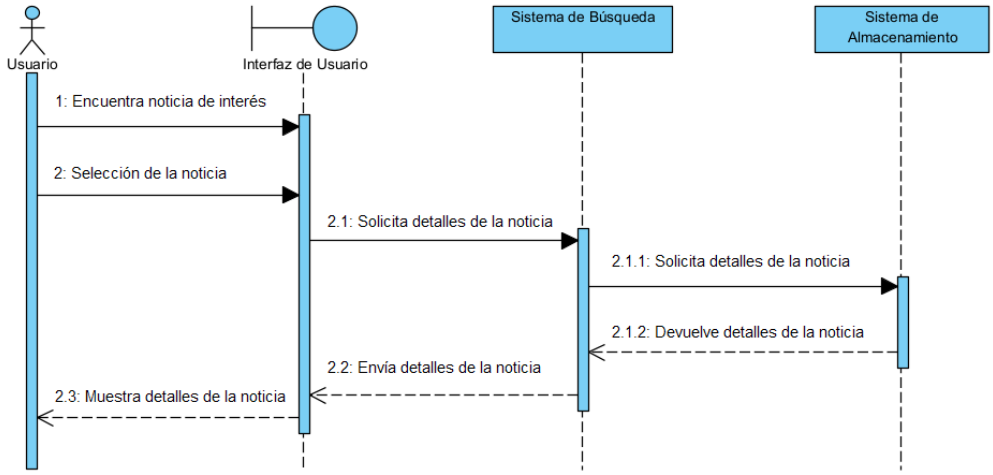

Figura 2.14: *Imagen visualización de noticias*

Cuando el usuario encuentra una noticia de interés el usuario accede a dicha noticia, el sistema de búsqueda recupera de la base de datos todos los datos referentes a la noticia y se muestran por pantalla al usuario.

2.4.5. Clasificar Noticias

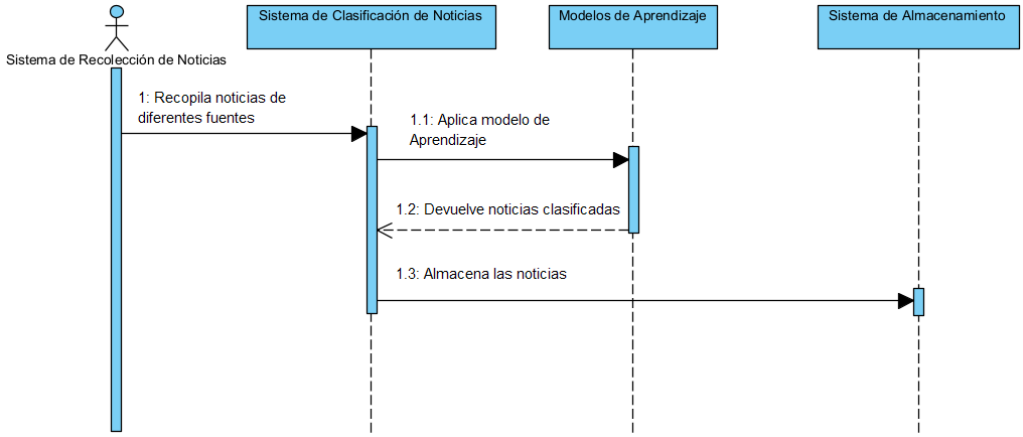

Figura 2.15: *Imagen categorización de noticias*

Este diagrama muestra otra subfunción del sistema, el sistema de recolección de noticias recopila las noticias de diversas fuentes, a estas noticias el modelo de aprendizaje las clasifica según los textos que tienen estas.

Una vez se clasifican las noticias se guardan en la base de datos como indica el diagrama.

2.5. Conclusiones

En esta fase de análisis, se han asentado las bases para desarrollar un sistema que recopile, clasifique y muestre noticias relacionadas con la Universidad de Jaén. Durante este proceso, se ha identificado lo que necesita el sistema y cómo debería funcionar para cumplir con las expectativas de los usuarios.

Entender bien estos requisitos es esencial para que las siguientes fases, como el diseño y la implementación, se lleven a cabo sin problemas.

También se ha explorado cómo podemos utilizar técnicas avanzadas, como el Procesamiento de Lenguaje Natural y el aprendizaje automático, para que el sistema clasifique automáticamente las noticias. Esto ayudará a organizar mejor la información y a hacerla más accesible.

Por último, se ha tenido en cuenta la importancia de que el diseño de la interfaz esté alineado con la identidad visual de la Universidad de Jaén. Elegir bien los colores, la tipografía y el diseño es clave para que la experiencia de los usuarios sea agradable y coherente con la imagen de la universidad.

Para un correcto desarrollo del proyecto, según el análisis general que se ha hecho, se necesitan una serie de módulos.

Dicho módulos dividen las distintas funciones que el proyecto necesita para poder funcionar y todos ellos deben de trabajar sincronizadamente para que el resultado obtenido sea el más óptimo.

Estos módulos se detallarán en el capítulo número 4, Fase de Diseño.

Capítulo 3

Contexto relacionado con el problema a resolver

Se procede con la explicación de los puntos más importantes a desarrollar en este proyecto.

Para un mejor entendimiento se hará una breve introducción al campo en el que se sitúa este TFG para más tarde dar detalle sobre los dos apartados en los que se basa todo el proyecto.

3.1. ¿Qué es el PLN?

El Procesamiento del Lenguaje Natural (PLN) es una rama de la inteligencia artificial que se enfoca en la interacción entre las computadoras y el lenguaje humano. Su objetivo principal es permitir que las máquinas no solo comprendan, sino que también generen y respondan en lenguaje natural, es decir, en el idioma que utilizamos los seres humanos en nuestra comunicación diaria.

Esto implica el desarrollo de algoritmos y modelos que puedan procesar y analizar grandes volúmenes de datos textuales o verbales, extrayendo significado y patrones de manera que las máquinas puedan interactuar con las personas de manera más intuitiva y efectiva.

En este proyecto, el PLN juega un papel crucial en la automatización de diversas tareas que están directamente relacionadas con la recolección, análisis y clasificación de noticias. Estas tareas incluyen, entre otras, la identificación de temas relevantes, la categorización de contenidos, la detección de tendencias y la generación de resúmenes automáticos de noticias.

Gracias al PLN, es posible manejar grandes cantidades de información textual de manera eficiente, permitiendo a las máquinas interpretar el contenido de las noticias de forma más precisa y coherente con el entendimiento humano.

3.1.1. Evolución del PLN

A lo largo de las últimas décadas, el campo del Procesamiento del Lenguaje Natural ha experimentado un desarrollo significativo, impulsado por los avances en las capacidades computacionales, la disponibilidad de grandes volúmenes de datos y la mejora en las técnicas de aprendizaje automático. Inicialmente, el PLN se basaba en reglas y modelos estadísticos simples que permitían realizar tareas básicas como la corrección ortográfica o la clasificación de documentos.

Con el tiempo, y especialmente con el auge del aprendizaje profundo y las redes neuronales, el PLN ha alcanzado niveles de sofisticación que eran impensables hace solo unos años. Modelos como los Transformers y, en particular, la aparición de modelos de lenguaje como GPT y BERT, han revolucionado el campo, permitiendo que las máquinas puedan identificar patrones complejos, entender contextos más amplios y generar texto con un grado de coherencia y fluidez que se aproxima al lenguaje humano.

En resumen, la evolución del PLN ha permitido que hoy en día podamos desarrollar aplicaciones capaces de identificar patrones o temas subyacentes en grandes volúmenes de texto de manera automatizada, facilitando así la extracción de información relevante y la toma de decisiones basada en datos.

3.1.2. Importancia del PLN en la clasificación de noticias

El PLN es muy importante en la clasificación de noticias, ya que permite procesar grandes cantidades de texto y extraer información relevante de manera eficiente.

Al analizar un corpus de noticias recolectadas de diversos periódicos, las técnicas de PLN pueden identificar automáticamente las palabras y frases mas relevantes de cada noticia, como 'olivar', 'curso', o 'baloncesto'.

De esta forma, al saber las palabras o frases con más peso en el artículo se pueden agrupar los artículos más similares. Por ejemplo, como es el caso del proyecto, si se recolectan 10.000 artículos de diferentes fuentes sobre la Universidad de Jaén, el PLN ayuda a agrupar automáticamente estos artículos en subcategorías como 'cultura', 'política', 'deportes' etc.

3.1.3. Desafíos del PLN en la recolección y clasificación de noticias

El principal desafío en la aplicación del PLN se trata de la gran diversidad lingüística que se puede encontrar en los artículos recopilados. Las noticias suelen estar escritas en un lenguaje formal, pero con variaciones significativas dependiendo por ejemplo de la fuente o el autor.

Otros desafíos importantes a los que hay que hacer frente también es la existencia de ambigüedad o ironía en los textos ya que pueden complicar la tarea de clasificar de manera

más precisa los temas de las noticias.

Se procede con la explicación de los dos apartados base en el desarrollo del tfg los cuales son:

- **Topic Modeling**
- **Scraping**

A continuación el primer apartado.

3.2. Clasificación de temas (topic modeling)

La clasificación de temas o topic modeling es una técnica utilizada para descubrir automáticamente los temas ocultos en un conjunto de documentos.

En el contexto de este proyecto, se emplea para analizar y clasificar noticias recolectadas de diversos periódicos, utilizando la técnica de Latent Dirichlet Allocation (LDA) la cual en capítulos mas tarde se detallará su funcionamiento.

3.2.1. Objetivos del Topic Modeling

Como se acaba de mencionar, esta técnica tiene como objetivo principal encontrar temas subyacentes en un conjunto de documentos como son las noticias.

Se busca clasificar noticias en función de los diferentes temas que abordan. Cada noticia se modela como una mezcla de varios temas, lo que permite una clasificación más precisa y contextual.

Al tener una noticia varios temas se le asignara aquel que tenga mayor peso en el conjunto total de temas a los que pertenece.

Cabe destacar que un buen uso del Topic Modeling es importante utilizar otras técnicas de PLN para poder preprocesar los textos que se van a clasificar y que los resultados obtenidos sean mucho mejores.

3.2.2. Desafíos y limitaciones de LDA en la clasificación de noticias

LDA es una herramienta poderosa pero presenta ciertas limitaciones. La interpretación de los temas generados, como se verá mas adelante, requiere de una validación manual y ajuste de los parámetros del modelo.

LDA divide las noticias en categorías, pero dichos temas no tienen título como tal, el título que se le asigna a cada tema se hace manualmente en base a los tópicos más representativos de uno.

El aprendizaje no supervisado, como el utilizado en LDA, también presenta sus propias limitaciones. Al no estar guiado por etiquetas o categorías predefinidas, el modelo puede generar agrupaciones de datos que no siempre son intuitivas o directamente relevantes para los objetivos específicos del análisis. Además, la falta de supervisión significa que la precisión del modelo depende en gran medida de la calidad y representatividad de los datos de entrada, lo que puede llevar a resultados menos consistentes o útiles si los datos no están bien distribuidos o si los patrones son difíciles de identificar sin intervención humana.

3.3. Scraping

El scraping es una técnica esencial en este proyecto, ya que permite la recolección automática de noticias desde diversos sitios web de periódicos. Dado que muchas fuentes de noticias, por no decir todas las de las provincia, no ofrecen APIs robustas para el acceso a sus contenidos, el scraping se convierte en una herramienta vital para obtener los datos necesarios.

3.3.1. ¿En qué consiste el scraping?

El scraping web consiste en utilizar scripts automatizados para navegar por páginas web y extraer información específica, como el título, la fecha o el contenido de las noticias. En este TFG se emplea la técnica de scraping para recolectar noticias de varios periódicos, almacenarlas en una base de datos, y luego utilizarlas en el análisis y clasificación mediante LDA.

3.3.2. Herramientas y técnicas de Scraping

Existen varias herramientas y técnicas que se pueden utilizar para el scraping de noticias, las dos que se han utilizado en este proyextos son:

- **BeautifulSoup:** Una librería de Python que facilita la extracción de datos de archivos HTML y XML.
- **Selenium:** Herramienta que permite la automatización de navegadores web.

Las aplicaciones que se le han dado a estas técnicas se detallarán en el capítulo de implementación, el capítulo 6.

3.3.3. Desafíos técnicos del Scraping en sitios de noticias

El principal desafío al que se ha hecho scraping es la capacidad de detección de bots que tienen algunos de los periódicos por lo que se han tenido que emplear técnicas algo más complejas así como la utilización de la técnica que se acaba de mencionar llamada 'Selenium'.

Otros desafío importante es la necesidad de manejar contenido dinámico y la posible modificación de la estructura del sitio web lo que conlleva a la modificación y ajuste de código.

Capítulo 4

Fase de diseño

4.1. Introducción

En esta fase del proyecto, se detalla el diseño del sistema desarrollado para la recolección y clasificación de noticias utilizando técnicas de scraping y topic modeling mediante LDA como se acaba de mencionar.

El diseño de este sistema incluye la planificación de la arquitectura general, los componentes involucrados, y la manera en que se comunican entre sí estos componentes. En este capítulo también se describe en detalle el despliegue de la arquitectura, lo que permite una mejor comprensión sobre cómo se estructuran y operan las distintas partes del proyecto.

4.2. Diseño de la arquitectura del sistema

El sistema para la recolección y clasificación de noticias consta de varios componentes clave, obviamente cada uno de ellos cuenta con una función específica dentro del flujo de procesamiento.

4.2.1. Componentes principales

- **Módulo de crawler y scraping**
  - **Función:** Se encarga de la recolección automática de noticias desde las webs de los periódicos.
  - **Entrada:** URLs de las páginas web objetivo.
  - **Salida:** Información de cada noticia recopilada.
- **Módulo de Preprocesamiento**

- **Función:** Realiza el preprocesamiento de los textos recolectados para prepararlos para la clasificación. Elimina las noticias que no son de interés para la clasificación y procesa los textos así como la eliminación de stopwords, lematización y tokenización.
- **Entrada:** Textos de las noticias en bruto.
- **Salida:** Textos limpios y listos para su clasificación.

- **Módulo de clasificación**
  - **Función:** Aplicar el algotritmo LDA utilizando la librería de Gensim para identificar y clasificar los temas presentes en las noticias.
  - **Entrada:** Textos de las noticias procesados.
  - **Salida:** Noticias clasificadas según el tema más predominantes en sus textos.
- **Módulo de Almacenamiento**
  - **Función:** Gestiona el almacenamiento de las noticias clasificadas, en MongoDB. Se usa también MongoDB Compass para visualizar mejor la base de datos.
  - **Entrada:** Resultados de la ejecución del algoritmo LDA.
  - **Salida:** Base de datos con artículos clasificados.
- **Módulo de Visualización y Consulta**
  - **Función:** Proporciona una interfaz web para que los usuarios puedan consultar los resultados obtenidos de la clasificación. De forma que aplicando algunos filtros se puedan mostrar las noticias relevantes para esos filtros.
  - **Entrada:** Datos almacenados en MongoDB.
  - **Salida:** Interfaz web.

4.2.2. Diagrama de despliegue

El diagrama de despliegue muestra como se implementan físicamente los componentes del sistema en un entorno local siguiendo la arquitectura cliente-servidor:

Como se puede ver, la interfaz está dentro del cliente, y el servidor compone los módulos importantes del sistema como almacenamiento, el cual interactúa con la interfaz, clasificación, Preprocesamiento y Crawler y Scraping.

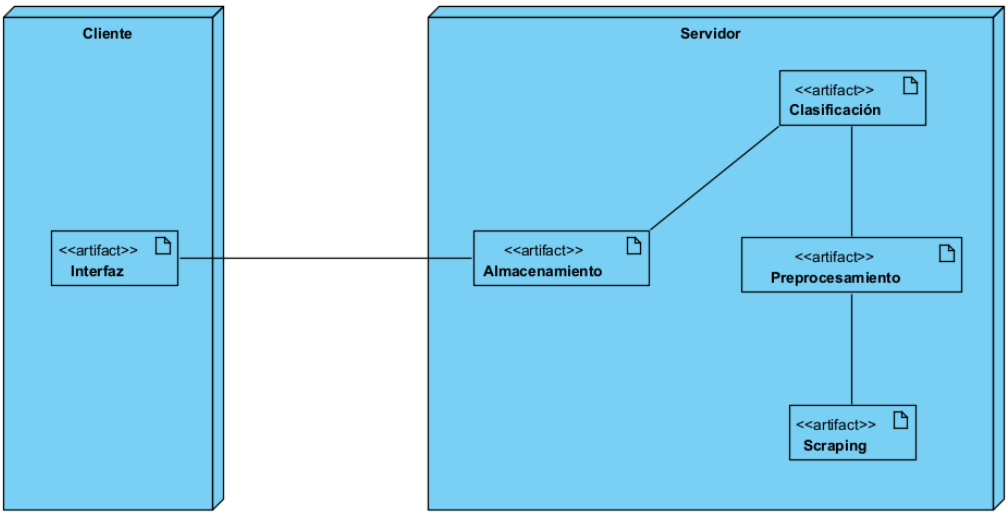

Figura 4.1: *Diagrama de Despliegue*

4.3. Diagrama de clases software

A continuación se presenta el diagrama de clases software el cual representa las clases que se han implementado así como el módulo en el que se encuentran:

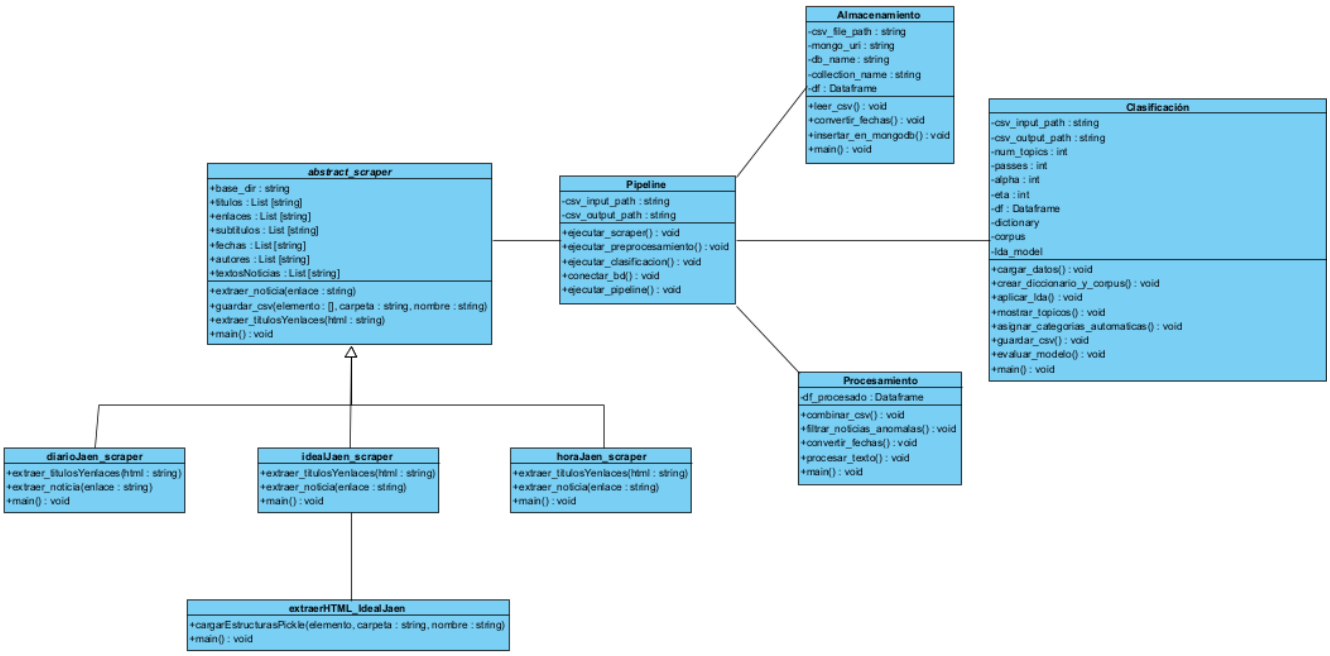

Figura 4.2: *Diagrama de clases software*

Como se puede observar se tiene una clase padre llamada 'Pipeline' la cual se encarga de llamar cada uno de los módulos independientes para que el sistema tenga un correcto desarrollo.

Esta clase se asocia con todas las demás ya que es la clase que 'controla' todo el sistema, de forma que el administrador, solo tendría que ejecutar esta clase para que todo el programa funcionara y se ejecutara correctamente.

Se ha creado una clase abstracta, que más tarde se detallará en el apartado de implementación, de forma que los scrapers implementados hereden de esta clase y sea un scraper para cada periódico en específico. Estas clases pertenecerían al módulo de crawler y scraping.

Las demás clases como Almacenamiento, Clasificación y Procesamiento pertenecen a su respectivo módulo.

De esta forma, además, se puede tener una visión más general sobre los métodos y atributos que compondrán las clases que formarán este sistema.

4.4. Diseño de datos

En el desarrollo del proyecto, uno de los aspectos claves es la estructura y almacenamiento de forma eficiente de las noticias recopiladas.

Las noticias consituyen datos no estructurados o en algunos casos semi-estructurados por

lo que se ha optado por la utilización de una base de datos NoSQL, concretamente MongoDB.

Se detalla primeramente porqué se ha elegido utilizar una base de datos NoSQL en vez de una SQL:

- **Modelo de Datos Flexible:** Una de las principales razones por las que se ha elegido NoSQL es la flexibilidad que ofrece en el modelo de datos. Las bases de datos SQL necesitan que los datos se estructures en tablas con esquemas rígidos y predefinidos. Sin embargo, las noticias que se recolectan pueden tener estrucutura variable, con campos que pueden estar presentes o no dependiendo del periódico.
- **Manejo de Datos No Estructurados:** Como se ha mencionado anteriormente, aunque las bases de datos SQL son capaces de manejar este tipo de datos, no son la opción más adecuada para hacerlo. NoSQL está diseñado específicamente para manejar datos no estructurados, permitiendo almacenar y consultar las noticias de manera eficiente.
- **Agilidad en el Desarrollo:** En un proyecto como este, donde la estructura de las noticias puede evolucionar y cambiar con el tiempo, es importante adaptar rápidamente el sistema de almacenamiento sin la necesidad de realizar cambios complejos en la base de datos. Con NoSQL es posible agregar o modificar campos en los documentos sin necesidad de alterar la base de datos, lo que permite desarrollar el proyecto de una forma más ágil.

  Obviamente, en una base de datos SQL, cualquier cambio en la estructura de los datos requiere modificaciones en el esquema, lo que complica todo aún más, especialmente si ya hay noticias recolectadas.
- **Relaciones No Esenciales:** En este proyecto, las noticias se manejan como entidades independientes, sin una necesidad crítica de mantener relaciones complejas entre diferentes conjuntos de datos. En un contexto donde cada noticia es tratada como un documento independiente, la ausencia de relaciones entre tablas o documentos no afecta la funcionalidad ni el rendimiento del sistema. Esto refuerza la decisión de optar por una base de datos NoSQL, donde la prioridad es la flexibilidad y la eficiencia en el manejo de grandes volúmenes de datos no estructurados, más que la gestión de relaciones entre datos.

Una vez expuestos los motivos por los que se ha optado por utilizar NoSQL se explican las razones por las que se ha utilizado MongoDB:

- **Escalabilidad Horizontal:** MongoDB es conocido por la capacidad de escalibilidad horizontal que tiene, lo que significa que puede manejar grandes volúmenes de datos distributendolos a través de servidores o nodos.

Dado que se prevé la recoleccion continua de noticias de diversas fuentes, el volumen de datos crecerá con el tiempo pero esto no será problema para MongoDB.

- **Almacenamiento y Consultas Eficientes de Datos No Estructurados:** Las noticias, como ya se sabe, son un tipo de dato no estructurado, lo cual se adapta perfectamente al modelo de documentos de MongoDB.

  Además, MongoDB ofrece capacidades avanzadas para consultas sobre datos no estructurados, permitiendo búsquedas eficientes en campos de texto, fechas y otros elementos que no siguen un formato fijo.

- **Alta Disponibilidad y Tolerancia a Fallos:** MongoDB ofrece características avanzadas de alta disponibilidad a través de la replicación automática y creación de réplicas de datos. Esto garantiza que, en casos de fallos, los datos estén seguros y accesibles.

- **Facilidad de Integración con Aplicaciones Modernas:** Otra capacidad de MongoDB que interesa mucho para el correcto funcionamiento del sistema es la capacidad que tiene para integrarse con integrarse con diversas plataformas y lenguajes de programación. Además, tiene una API que es perfecta para una conexión con sistemas externos que vendría genial en caso de escalar el proyecto en un futuro y llevarlo al siguiente nivel.

Una vez expresados todos los motivos de las elecciones tomadas referentes al aspecto de almacenamiento de las noticias se procede con el siguiente capítulo, el cual se habla sobre la clasificación automática de noticias.

Capítulo 5

Marco de evaluación del clasificación

5.1. Diseño del clasificador con LDA.

En este apartado se encuentra toda la información referente a la clasificación de noticias que se ha llevado a cabo.

La clasificación se ha implementado con el algoritmo LDA (Latent Dirichlet Allocation), se trata de un algoritmo de aprendizaje automático no supervisado.

LDA es un algoritmo muy usado en el ámbito del PLN para descubrir temas subyacentes en un determinado conjunto de documentos. Se trata de una técnica de modelado de tópicos que asume que cada documento es una mezcla de varios temas, y cada tema una distribución de palabras.

Algunas de las aplicaciones que tiene este algoritmo son:

- **Análisis de texto**
- **Recomendación de contenido**
- **Organización de información**

Los principales motivos por los que se ha escogido este algoritmo para la clasificación de noticias son los siguientes:

- **Descubrimiento de temas ocultos:** LDA tiene una gran capacidad para identificar temas, es muy efectivo para descubrir temas subyacentes en un gran conjunto de noticias, como es el caso.

  Cada tema representa un conjunto de palabras que suelen aparecer juntas, lo que permite identificar automáticamente los diferentes temas de los que tratan las noticias deportes, cultura, política etc... , sin necesidad de etiquetas previas.

Sí que es verdad, que en este algoritmo, es necesario inicializar el número de temas que se desea que encuentre el algoritmo, es decir, antes de la ejecución del algoritmo se debe especificar el número de temas en los que el algoritmos basará su clasificación.

- **Enfoque en el contenido textual:** Este algoritmo, a diferencia de otros, se basa exclusivamente en el contenido contextual de las noticias para clasificar las noticias, lo que es perfecto cuando, como es el caso, se tiene acceso directo al texto pero no a etiquetas de clasificación preexistentes.
- **No supervisado:** Al ser un algoritmo no supervisado significa que no se requiere de un conjunto de entrenamiento etiquetado previamente, es genial ya que es muy útil especialmente cuando no se dispone de una clasificación previa de las noticias.
- **Análisis inicial del corpus:** LDA al clasificar las noticias del corpus, permite obtener una visión general de los diferentes temas que tratan, por lo que si en un futuro se quiere hacer un análisis de las noticias, así como saber cuantas noticias tenemos referenciando cada tema, LDA sería estupendo.
- **Escalable a grandes volúmenes de datos** El algoritmos es eficaz incluso cuando el corpus tiene un gran volumen de datos.

  Como los datos que se están clasificando son noticias y estas surgen más y más cada día, el corpus se va engrandeciendo, pero al utilizar este algoritmo no hay problema en la cantidad de datos que se tienen ya que es capaz de operar con una gran cantidad de noticias sin ningún inconveniente.

Una vez se han expuesto las aplicaciones, motivos y funcionamiento de este algoritmo se procede con los resultados que de este mismo se han obtenido.

El resultado, como era de esperar, ha sido la creación de un archivo en formato '.csv' con las noticias recopiladas y un campo añadido llamado 'Categoría'.

Tras probar varias ejecuciones, se ha llegado a la conclusión de que el número de temas que mejor se adaptaba al número de noticias recopiladas era 9.

Tópicos

Los tópicos que se han clasificado se pueden observar en lo que devuelve la ejecución del programa:

```
(0, '0.012*"patrimonio" + 0.009*"museo" + 0.007*"arqueología" + 0.006*"zona"')

(1, '0.012*"investigación" + 0.010*"investigadores" + 0.007*"universidades" + 0.006*"g
```

```
(2, '0.017*"aceite" + 0.016*"oliva" + 0.012*"investigación" + 0.011*"salud"')

(3, '0.006*"cultura" + 0.005*"maría" + 0.005*"josé" + 0.004*"juan"')

(4, '0.010*"provincia" + 0.008*"rector" + 0.008*"juan" + 0.007*"internacional"')

(5, '0.020*"estudiantes" + 0.013*"curso" + 0.010*"grado" + 0.007*"alumnos"')

(6, '0.008*"años" + 0.005*"ahora" + 0.004*"aunque" + 0.004*"hacer"')

(7, '0.013*"euro" + 0.009*"millones" + 0.008*"junta" + 0.008*"proyecto"')

(8, '0.010*"persona" + 0.009*"programa" + 0.009*"actividades" + 0.007*"social"')
```

Como consecuencia se le han asignado los siguientes temas respectivamente en función de las palabras más representativas de cada tema:

```
0: "Patrimonio y Cultura",
1: "Investigación y Proyectos Universitarios",
2: "Aceite de Oliva y Salud",
3: "Cultura y Premios",
4: "Provincia y Desarrollo Internacional",
5: "Estudiantes y Educación",
6: "Otras categorías",
7: "Financiamiento y Proyectos Regionales",
8: "Programas y Actividades Sociales"
```

Es de destacar que haya una categoría que sea 'Aceite de Oliva y Salud' ya que esto no es muy normal a noticias referentes de a una Universidad. Esto sucede ya que la provincia de Jaén, es la mayor exportadora de aceite del mundo, por lo que en los periódicos hay muchas noticias referentes al aceite y a la investigaciones y proyectos que se hacen de este mismo en la Universidad de Jaén. Es por este motivo por el cual hay una categoría enfocada al aceite.

Las demás categorías son mas normales así como 'cultura y premios' o 'Estudiantes y Educación', que son categorías que obviamente están plenamente relacionadas con una universidad.

5.2. Evaluación del clasificador.

Una vez se han clasificado las noticias es necesario evaluar la clasificación realizada por el modelo.

Así a priori, como se puede observar, las palabras más representativas de cada tópico son palabras coherentes y que tienen un sentido común y relación entre las que son del mismo tópico.

Es por ello que la tarea de asignarle un tema a cada tópico sabiendo solo las palabras más representativas ha sido fácil.

Para evaluar el modelo utilizaremos las siguientes técnicas:

- **Coherencia de Tópicos:** Se trata de una métrica popular para evaluar modelos de tópicos como es el caso del LDA. La métrica se encarga de medir el grado de similitud entre las palabras existentes en un mismo tópico. Esta métrica se ha implementado con la librería específica de 'gensim' denominada 'CoherenceModel' la cual es capaz de calcular la medida de coherencia entre los tópicos.

- **Perplejidad:** Esta es una métrica estándar que mide como de bien un modelo predice una muestra. Para evaluar el modelo de esta forma se ha implementado una función que trae por defecto el modelo LDA, en concreto 'lda_model.log_preplexity()' que permite extraer directamente el grado de perplejidad del corpus.

- **F1-Score** Esta métrica es muy usada y es especialmente útil cuando se está trabajando con problemas de clasificación y se quiere medir la precisión y exhaustividad de las categorías que se han clasificado automáticamente. Para usar esta métrica es necesario tener un pequeño conjunto de datos etiquetados. En este caso no se tiene ningún dato etiquetado ya que se parte desde cero a la hora de recolección de noticias.

  Para ello se han etiquetado manualmente 50 noticias, de esta forma comparando estas noticias clasificadas manualmente y las clasificadas automáticamente se podrá obtener una visión general de lo bien que ha funcionado el sistema de categorización si se aplica esta métrica.

Los resultados obtenidos han sido los siguientes:

```
Coherencia de tópicos: 0.510
Perplejidad: -8.965
F1-Score: 0.885
```

Coherencia de tópicos: Esta métrica se mide entre 0 y 1, siendo 1 una máxima coherencia y 0 mínima coherencia. Se obtiene una coherencia de *0.51* la cual es razonablemente buena.

Indica que los temas generados por el modelo LDA son moderadamente coherentes, es decir, las palabras en cada tema tienen un cierto grado de relación semántica.

Perplejidad: La perplejidad como se ha mencionado anteriormente, mide en términos generales la incertidumbre o desajuste del modelo, cuanto menor sea el valor, mejor habrá sido el ajuste a los datos.

Teniendo en cuenta la cantidad de datos que ha manejado el modelo, un valor de *-8.9* sugiere que el modelo tiene un buen ajuste a los datos y que es capaz de trabajar bien con el gran volumen de datos que existente en el corpus.

En general, el algoritmo ha hecho un buen trabajo de clasificación ya que los tópicos son moderadamente coherentes, y el modelo está ajustado a los datos.

F1-Score: Antes de empezar a analizar el resultado, es importante saber que no se ha aplicado esta métrica a las noticias clasificadas con la categoría 'Otra categoría' ya que no se cree que tenga mucho sentido incluir esta categoría dentro de las que se tienen en cuenta para la métrica. El resultado ha sido 0.885, este resultado sale de aplicar la métrica F1-Score a cada categoría para más tarde utilizar Micro-Averaging.

Es decir, se suman todos los True Positives, False Positives y False Negatives para aplicar a esto la métrica F1-Score global.

Se muestra los datos para las 8 categorías:

Categoría	TP	FP	FN
0: Patrimonio y Cultura	12	1	1
1: Investigación y Proyectos Universitarios	11	2	2
2: Aceite de Oliva y Salud	10	2	1
3: Cultura y Premios	11	1	1
4: Provincia y Desarrollo Internacional	10	1	2
5: Estudiantes y Educación	9	2	2
6: Financiamiento y Proyectos Regionales	8	2	1
7: Programas y Actividades Sociales	12	1	2

Figura 5.1: *Imagen F1-Score categorías*

La suma global es la siguiente:
TP global: 92 *FP global:* 12 *FN global:* 12
Por lo que una vez aplicada la siguiente fórmula:

$$\mathrm{F1} = \frac{2 * \mathrm{TP}}{2 * \mathrm{TP} + \mathrm{FP} + \mathrm{FN}}$$

Figura 5.2: *Imagen Fórmula F1-Score*

Se obtiene el resultado de 0.885 lo que indica que el modelo tiene un buen desempeño general en términos de precisión y recuperación. El valor está bastante cerca de 1.0 que es el límite por lo que se podría decir que significa que el modelo está haciendo un buen trabajo en general al clasificar las instancias en todas las categorías.

Capítulo 6

Implementación

6.1. Descripción General

Para la implementación del sistema, primero, se ha llevado a cabo un proceso de selección sobre las fuentes en línea (periódicos), de los cuales se han seleccionado aquellos, que, a juicio de un buen desarrollo del proyecto, son los más relevantes para la provincia. Este proceso también tuvo en cuenta la calidad de las noticias y la accesibilidad de las mismas, con el objetivo de facilitar la creación de scrapers eficientes. Así, se evitó la necesidad de desarrollar programas excesivamente complejos.

Para la implementación del sistema, inicialmente se consideraron varios periódicos, estos periódicos fueron los siguientes:

- Diario Jaén
- Jaén Hoy
- Extra Jaén
- Hora Jaén
- LaContraDeJaén
- Ideal Jaén

Todos estos periódicos se analizaron, y tras evaluar la relevancia, calidad y accesibilidad de las noticias ofrecidas por cada uno de ellos, se decidió continuar con los siguientes:

- Diario Jaén
- Hora Jaén

- Ideal Jaén

De esta forma se garantiza que si los periódicos son de calidad, sus noticias por ende, serán no solo relevantes sino que además serán fiables y de interés para aquellos usuarios que se quieran informar sobre la Universidad de Jaén. Al enfocarse en fuentes que son de confianza, se obtiene información veraz y actualizada, por lo que la información que se expondrá en el dosier estará contrastada por estos periódicos.

Estos, a su vez, se han elegido por la facilidad con la que se puede ejecutar la tarea de desarrollar los correspondientes scrapers , ya que las páginas web de estos periódicos tienen una estructura más clara y accesible, reduciendo la complejidad del código necesario para extraer la información.

Se tratará a continuación el proceso seguido para la implementación de los diferentes scraper que se ha llevado a cabo una vez se ha dejado claro cuales son los periódicos que se han tenido en cuenta para la recolección de noticias.

El **propóstio general** del sistema de recolección de noticias es automatizar y optimizar la captura de información relevante sobre la Universidad de Jaén. El sistema está diseñado para:

- **Centralizar la Información:** Recopilar noticias de diferentes fuentes en un solo lugar, de forma que todos lo eventos relevantes que hayan ocurrido o que ocurrirán sobre la Universidad de Jaén estén recopilados en un solo lugar de información.
- **Automatizar el Proceso de Recolección:** Utilizar scrapers para realizar la recolección de noticias de manera automática y periódico. De esta forma se reduce la necesidad de intervención manual ya que se asegura la recolección de datos en tiempo real y de manera actualizada.
- **Garantizar la Calidad y Relevancia de los Datos:** Al seleccionar los periódicos que se han mencioando anteriormente se verifica que las noticias recopiladas sean relevantes y fiables.
- **Proporcionar una Solución Escalable y Mantenible:** A la hora de crear el diseño del sistema se le ha dado mucha importancia a la escalabilidad de este para que sea más fácil, por ejemplo, si se quisiera añadir otro periódico a la hora de recolectar las noticias.

Como lenguaje de programación que se ha usado para la implementación del sistema se ha optado por Python por estas tres principales causas:

- **Simplicidad y Legibilidad del Código:** Python ofrece una sintaxis clara y concisa, ideal para programar el código de scrapers, facilitando la comprensión y modificación del código incluso para quienes no están muy familiarizados con él.

- **Bibliotecas y Frameworks Especializados:** Para la tarea de recopilación de noticias es imprescindible trabajar con unas herramientas adecuadas, las bibliotecas que proporciona Python se adaptan muy bien a los requisitos de nuestro programa, se han utilizado bibliotecas como 'requests' y 'BeautifulSoup' entre otras, las cuales se detallaran más adelante.

- **Facilidad para la Integración y Escalabilidad:** Uno de los objetivos clave en el sistema es desacoplar la API de la interfaz, lo que permitiría una arquitectura más modular y flexible. Python facilita esta separación al ofrecer una integración sencilla con bases de datos, APIs y servicios en la nube. Esta capacidad para conectar y gestionar diversas fuentes de datos y volúmenes de información no solo favorece una arquitectura escalable, sino que también permite que el sistema se adapte con agilidad a diferentes necesidades y configuraciones.

6.2. Arquitectura del sistema

En cuanto a la arquitectura del sistema se ha optado por el diseño de un sistema modular basado en una clase abstracta llamada 'AbstractScraper' que permite la incorporación de nuevos periódicos sin necesidad de rediseñar el sistema desde cero. Esta clase proporciona una base común y define los métodos esenciales que deben ser implementados por cada scraper.

Se ha implementado esta clase abstracta ya que permite mucha flexibilidad y eficacia a la hora de programar los respectivos scraper de cada periódico.

Los motivos por los que se ha decidido usar una clase abstracta para que los scrapers de los periódicos hereden de ella son los siguientes:

1. **Estandarización del Diseño:** Utilizar una clase abstracta permite definir un conjunto de métodos y propiedades para que todas las subclases, es decir, los scrapers, deben implementar. Esto asegura que cada scraper específico para un periódico siga esta estructura y funcionalidad.

2. **Reusabilidad del Código:** Una clase abstracta permite encapsular el código común que será reutilizado por las subclases. Se evita de esta forma la duplicación de código. Métodos que son para almacenar en memoria la información recopilada son comunes a todos los scrapers.

3. **Facilidad de Mantenimiento y Expansión:** La clase abstracta además facilita el mantenimiento del sistema. Modificar una funcionalidad se convierte en una tarea mucho más sencilla con esta clase abstracta. Asimismo, si se desean agregar nuevos scrapers,

no habría inconveniente alguno, ya que solo sería necesario crear nuevas subclases que hereden de esta clase. De esta forma, se asegura que nuestro sistema sea escalable.

4. **Seguridad y Robustez:** Se garantiza que las subclases implementen los métodos necesarios, por lo que se previene errores en tiempo de ejecución por métodos no implementados. Esto añade más robustez y seguridad en el sistema.

Se hace referencia aquí al UML detallado anteriormente para ver con má claridad lo explicado y lo que se detaallará a continuación. El diagrama se encuentra en este ***enlace***.

Este diseño modular se ajusta a la perfección al propósito del sistema.

6.3. Clase Pipeline

Antes de empezar a detallar la clase abstracta se comentará la clase que llama a esta clase, la clase Pipeline. Esta clase, como se ha mencionado anteriormente es la clase que controla todo el sistema ya que es la encargada de realizar las llamadas respectivas a cada módulo para que ejecute las clases correspondientes.

La clase tiene una instancia de cada módulo, es decir, de cada clase, y métodos que hacen la llamada respectiva a cada uno de los módulos correspondientes.

El código implementado se encuentra en el siguiente ***enlace***.

Es importante que la secuencia de llamadas que se hacen estén en orden ya que así se asegura que cada etapa se realice correctamente antes de pasar a la siguiente. Esto es clave para el correcto funcionamiento del sistema.

Se procede con la explicación de la clase abstracta.

6.4. AbstractScraper Componentes Principales

A continuación, se detalla la arquitectura y funcionalidad de esta clase abstracta.

Las librerías usadas para la creación de la clase AbstractScraper son las del siguiente ***enlace***.

La clase AbstractScraper utiliza el módulo *'abc'* que proporciona Python para definir métodos abstractos, que aseguran que todas las subclases implementen ciertas funcionalidades, es clave para la tarea de recolección de noticias.

1. **Inicialización de Variables:** Se hace mediante el siguiente método:

- **__init__(self, base_dir):** Este método juega un papel muy importante ya que es el constructor de la clase y se encarga de inicializar las variables que se utilizarán para almacenar la información extraída de los periódicos. El código está se encuentra en el ***enlace***.

  La varible 'base_dir (String)' establece el directorio donde se almacenará la información recopilada al final del programa, mientras que las demás listas son las específicas para guardar cada tipo de información que se recoge, título, enlace, el texto de la noticia etc.

  Estas listas permiten que la información recolectada sea organizada de manera estructurada y accesible para su posterior procesamiento y almacenamiento.

2. **Métodos Abstractos:** Estos métodos se detallarán más sobre ellos en cada scraper, y son:

   - **extraer_titulosYenlaces(soup):** Método abstracto que debe de ser implementado por las subclases para extraer solo los títulos y enlaces desde el contenido HTML de las páginas por las que va navegando el scraper. El código se encuentra en el siguiente ***enlace***.
   - **extraer_noticia(enlace):** Se trata de otro método abstracto que también debe de ser implementado por las subclases, esté método se encarga de navegar noticia por noticia recopilando todo lo interesante así como la fecha, la noticia en sí etc. El código se encuentra en este ***enlace***.

3. **Métodos para Guardar Datos:** Se dispone además, ya que es necesario, de un método que permita almacenar en memoria estas listas anteriormente mencionadas.

   El formato de almacenamiento que se ha elegido en un principio (luego se usará una base de datos) es en formato '.csv', mediante la librería pandas. Esta librería que proporciona Python es muy sencilla de utilizar, además de ser flexible y escalable. Permite manejar grandes volúmenes de datos, por lo que se piuede almacenar de una manera eficiente las noticias recopiladas para su posterior procesamiento, pandas es ideal para esta tarea.

   El método con el cual se almacena las noticias en memoria es el siguiente:

   - **guardar_csv(self, carpeta, nombre):** Método que permite almacenar las noticias en memoria mediante el uso de la librería pandas. Los parámetros respectivamente son:
     - **carpeta (String):** Nombre de la carpeta donde se guardarán la información recopilada.

- **nombre (String):** Nombre que tendrá el archivo que contendrá la información.

Como carácter separador se ha empleado el ';' y simplemente se hace uso de la librería pandas para poder almacenar el archivo '.csv' en memoria mediante lña función 'dataframe.to_csv'

El código implementado se encuentra en el siguiente ***enlace***.

4. **Método Principal:** Por último se tiene el método principal que también debe de ser programado en las subclases, este método es:

   - **main(self):** Un método placeholder con el que se fuerza la implementación de un método principal en cada subclase, de esta forma se asegura que cada scraper tenga un punto de entrada definido.

     El código es muy sencillo y se encuentra en el siguiente ***enlace***.

     Se hace uso de 'raise' el cual se utiliza para generar una excepción en Python. La excepción que se lanza es 'NotImplementedError' al cual es una excepción específica que se utiliza para indicar que un método o función no ha sido implementado.

Se da por terminada así la explicación del código de la clase principal de nuestro programa, la clase abstracta llamada AbstractScraper.

Se procede con la explicación sobre cómo se ha programado la recolección de noticias en cada periódico.

6.5. Diario Jaén

6.5.1. Introducción al Diario Jaén

Diario Jaén es uno de los periódicos más relevantes de la provincia de Jaén. Es un periódico con bastante experiencia ya que fue fundado en 1941 y ha sido, desde entonces, una fuente confiable de noticias locales, regionales e incluso nacionales. Es un periódico que abarca una amplia gama de temas, incluyendo noticias sobre política, economía, deportes y como no, de la Universidad de Jaén. La elección de este periódico como una de las fuentes para nuestro sistema de scraping se debe a la relevancia y popularidad que tiene.

6.5.2. Motivos para la la Elección del Diario Jaén

Se detallan a continuación los principales motivos de la elección de este periódico.

- **Variedad de Contenidos:** Proporciona una amplia gama de noticias que cubren todos los temas de interés relacionados con la Universidad de Jaén por lo que cualquier noticia publicada sobre la UJA será de nuestro interés.
- **Relevancia Local:** Como se ha comentado anteriormente Diario Jaén es una de las fuentes clave de información de la provincia de Jaén.
- **Calidad Periodística:** Destaca por su rigor y calidad en la producción de noticias.
- **Facilidad para implementar el scraper:** Tras una análisis profundo de muchas de sus noticias se ha podido obtener múltiples patrones en su código HTML, por lo que a la hora de programar el scraper se puede hacer uso de dichos patrones para recopilar la información que se necesita para nuestro sistema de recopilación de noticias.

6.5.3. Desafíos del Scraping del Diario Jaén

Scrapear un sitio web de noticias como Diario Jaén no ha sido tan fácil como se esperaba en un primer lugar, ya que ha surgido múltiples obstáculos a la hora de recorrer el sitio web. Se muestran los desafíos técnicos que se han tenido que superar para la programación de dicho scraper:

- **Ausencia de una página con todas las noticias:** La principal idea que se tenía a la hora de programar el scraper era que recorriera una página web del periódico y recopilar absolutamente todas las noticias que este periódico ha lanzado a lo largo de su vida. Una vez se recopilaran todas las noticias de este periódico se procedería con un procesado de todas esas noticias filtrando y eliminando aquellas que no son de nuestro interés.

  Por ejemplo se eliminaría todas las noticias que no contuvieran en su título o en el cuerpo de la noticia las palabras 'Universidad de Jaén' de esta forma se aseguraría que todas las noticias son relevantes (directa o indirectamente) para la Universidad de Jaén.

  Esto no ha sido posible, ya que el periódico carece de una página web de este estilo, lo más que tiene es una sitio donde se encuentran tres páginas con las noticias más recientes. Esto no es suficiente para el sistema de recolección por lo que finalmente se ha optado por utilizar el buscador interno que proporciona dicho periódico.

  Cuando se hace una consulta en dicho periódico lleva a un sitio donde se muestran todas las noticias que contienen en su cuerpo de noticia, o en su título, la consulta que se ha proporcionado, así como el número de noticias total que se han encontrado.

  Esto también ha sido un reto ya que de primeras el buscador no funcionaba del todo bien, se ha estado analizando y buscando información por internet para entender como funciona este buscador y poder hacer un uso del buscador mucho más eficiente.

Un ejemplo de un problema que se tuvo es el siguiente:

Figura 6.1: *Ilustración buscador Diario Jaén*

Como se puede observar, tras buscar con la consulta 'Universidad de Jaén' se obtuvieron unos diez mil resultados, se analizaron varias de las noticias que se recuperaron y se vio que muchas de ellas no tenía 'Universidad de Jaén' en los cuerpos de sus noticias, por lo que se llegó a la conclusión de que el buscador recopilaba las noticias en forma de OR, es decir, hacían una recuperación de las noticias, en este caso, que tuvieran la palabra 'Universidad' o 'Jaén', 'de' al ser una stopword el buscador lo elimina automáticamente.

Se buscó por información por internet para entender el problema y como solucionar este, se llegó a la siguiente conclusión:

Esta vez, hacer uso de unas comillas al principio y al final hace que el buscador recupere las noticias que hablan sobre la 'Universidad de Jaén'. Se puede observar que recupera unas siete mil noticias, tres mil menos que con la anterior consulta, es decir, se han eliminado aquellas que no contienen la frase 'Universidad de Jaén' en sus noticias.

Se analizaron las noticias recuperadas y efectivamente todas ellas eran relevantes y de interés para la Universidad de Jaén.

- **Diversas páginas a recorrer:** Como se menciona anteriormente, cuando se utiliza el

Figura 6.2: *Ilustración buscador Diario Jaén*

buscador del Diario Jaén para recuperar noticias, el sistema redirige a una página web que contiene los primeros resultados de la búsqueda.

Esta página inicial muestra un conjunto limitado de noticias, generalmente cada página muestra unas diez noticias, por lo que hay que seguir recorriendo las siguientes páginas hasta que las noticias se agoten y no se hayan publicado más.

A la hora de explicar el código se hará mención de esto mismo y de como se ha resuelto el problema.

6.5.4. Implementación del Scraper Diario Jaén

La implementación del scraper se ha hecho con un estructura clara y organizada lo cual permite que se extraigan noticias de una forma eficiente, se explica el código a continuación:

1. **Librerías importadas:** Se han importado las siguientes librerías, las cuales se encuentran en el siguiente ***enlace***.

   Todas ellas son necesarias para el correcto funcionamiento del sistema y se verán en los métodos como se hace uso de estas.

2. **Método extraer_titulosYenlaces(soup):** El código implementado para este método está el siguiente ***enlace***.

Este método se utiliza para extraer los títulos y enlaces, como su propio nombre indica, de las noticias que hay en cada página que se recorre. Se hace uso de la librería BeautifulSoup para encontrar lo que se desea en el código HTML parseado de la página web, de esta forma podemos extraer del código los títulos y los enlaces de interés.

Simplemente estudiando y analizando el código HTML de las diversas páginas del periódico se puede obtener en qué contenedores está la información que se desea, y simplemente bastaría con ajustar el código para extraiga la información que hay dentro de dichos contenedores.

Se tuvo un pequeño problema con los enlaces y es que a la hora de extraerlos habría que añadirle 'https://www.diariojaen.es' al principio del enlace extraído, esto es debido a que solo se extraía el final y no llevaba a ninguna página, por lo que para recorrer las noticias recopiladas y acceder a su contenido dicho enlace este no sería válido.

En el caso en el caso en el que haya algún problema y no se encuentre el título o el enlace se añadirá con el término 'ERROR'.

Al llegar a una página que no tiene más noticias el bucle for no se ejecuta ya que 'artículos' está vacío por lo que se sale de la función devolviendo una lista de títulos y enlaces vacía.

Guardamos los enlaces y títulos de cada página en sus respectivas listas y lo devolvemos al finalizar la función.

3. **Método extraer_noticia(enlace):** Este método se encarga de, dado un enlace de una noticia que se le pasa por cabecera, recorrer el cuerpo de la noticia recopilando los datos de interés, el código se encuentra en el siguiente ***enlace***.

   El código del método como se puede comprobar es extenso ya que abarca varias posibilidades de que ocurra algún error en el servidor.

   En variadas ocasiones, al entrar en el enlace de una noticia, el servidor no responde bien, y en vez de procesar la solicitud y responder a esta con un código HTML algunas veces devuelve un código sin estructura el cuál no es posible parsear. Es por ello que se ha implementado al principio de la función unas línea de código que comprueba que el código proporcionado es realmente código HTML.

   Esto se hace gracias a 'resultado.headers.get('Content-Type', '')' el cual extrae la primera línea del código proporcionado y si esta línea no es 'text/html' significará que e código no es válido para nuestro sistema. Al no ser válido devuelve 'ERROR' en todas los campos de información para esa noticia en concreto

   Una vez comprobado que el código efectivamente es HTML se procede con la extracción de la respectiva información que se desea. Dicha información se vuelve a encontrar y a

extraer con la librería BeautifulSoup, igual que para los títulos y enlaces. Previamente se ha analizado la página web de varias noticias para ver sus patrones en común y usarlos para que la creación del método sea válido para todo tipo de noticias.

En el caso en el que no se encuentre un tipo de información por un error o directamente por su ausencia se le asignará un valor por defecto:

- subtítulo - 'Sin subtítulo'
- fecha - 'Error con la fecha'
- autor - 'Anónimo'
- noticia_completa - 'ERROR'

Donde más **problemas** se ha tenido a la hora de implementar el código ha sido en el apartado de extracción del cuerpo de la noticia. El cuerpo de la noticia generalmente se divide en varios apartados, normalmente hay más de un párrafo por noticia, por lo que hay que recorrer todos estos párrafos de información y añadirlos a una lista que los almacene.

Una vez recorrida la noticia entera hay que unir todos los párrafos en un mismo texto, esto se ha implementado haciendo uso de un 'join', de la siguiente manera '"\n".join(apartados)', de esta forma todos los textos se unen en un mismo string, y entre texto y texto hay un salto de línea.

Esto facilitará a la hora de mostrar el cuerpo de la noticia en la interfaz ya que sería solo agregar este texto para visualizar el cuerpo entero de la noticia.

4. **Método main():** Este es el método principal de este primer escraper. El código implementado está en el siguiente ***enlace***.

   Este sería el código del método 'main' el cual es el encargado de hacer las llamadas correspondientes a los anteriores métodos mencionados.

   El código del método 'main' comienza inicializando el enlace al sitio web del Diario Jaén, este enlace web va cambiando cada día, y lo curioso es que el cambia con respecto al día del año en el que nos encontramos. El enlace tiene incluido el día en el que se accede a dicho enlace, por lo que es necesario actualizar el enlace al día en el que se ejecuta el scraper.

   Para solucionar este problema de una forma automática se ha implementado en pocas líneas de código una efectiva solución, para ello se hace uso de la librería 'datetime' la cual ofrece la posibilidad de tener la fecha actual en el formato que se desee.

   Simplemente se crean unas variables con las fecha actual y en los distintos formatos necesarios para el correcto funcionamiento del enlace.

Una vez se tiene las fechas en el formato correcto, se puede incorporar dichas fechas dentro del enlace con una técnica que permite Python llamada f-strings. Esta técnica es muy sencilla y se basa en insertar variables directamente dentro de una cadena de texto.

Para usar esta técnica, simplemente se coloca la letra f antes de las comillas que delimitan el enlace y se utiliza las llaves '' para encerrar las variables que contienen las fechas que queremos inyectar. Python lo que hará será reemplazar las llaves con el valor de las variables correspondientes.

Se deja a continuación una imagen que lo explica mejor visualmente:

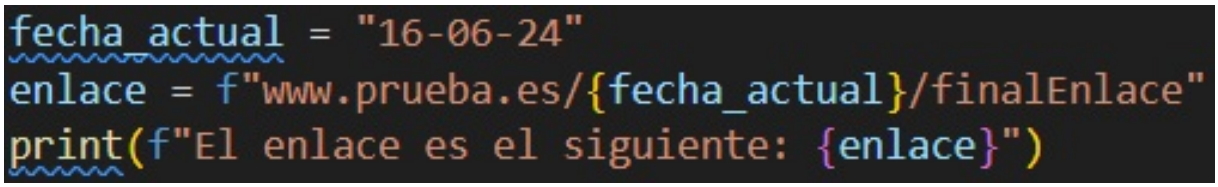

Figura 6.3: *Ilustración inyección fecha en enlace*

Con su respectiva salida:

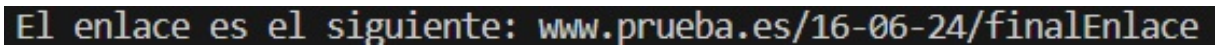

Figura 6.4: *Ilustración salida inyección fecha en enlace*

Una vez que se tiene el enlace del sitio web inicial, es necesario ir actualizando este enlace para que el scraper pueda recorrer todas las páginas con las noticias recuperadas. Cabe destacar que el enlace dirige directamente hacia donde el buscador interno redirige cuando se hace la consulta "Universidad de Jaén".

Para tratar el tema de recorrer las páginas, que se ha mencionado anteriormente, se ha solucionado de una forma muy sencilla. Simplemente se inicializa una variable llamada pagina a un número, en este caso a uno ya que queremos empezar el recorrido desde la primera página del periódico. Esta variable se insertará en el enlace de la siguiente forma, justo donde debería ir ese numero se sustituye por unas llaves esta manera '', con nada en su interior.

La forma en la que se inyecta esta variable en el enlace es diferente a la anterior mencionada. Se hace uso de la función '.format(pagina)' que proporciona Python, de esta forma se sustituye las llaves por el numero de página que se está recorriendo. Para entenderlo de una forma más visual:

```
pagina = 1
enlace = f"www.prueba.es/{{}}/finalEnlace"
enlace_con_pagina = enlace.format(pagina)
print(f"El enlace es el siguiente: {enlace_con_pagina}")
```

Figura 6.5: *Ilustración salida inyección página en enlace*

Con su respectiva salida:

```
El enlace es el siguiente: www.prueba.es/1/finalEnlace
```

Figura 6.6: *Ilustración salida inyección página en enlace*

Esta función ('.format(pagina)') se hace uso de ella cada vez que se inicia de nuevo el bucle while, de forma que se va actualizando la variable 'pagina' incrementándose en uno, y por ende actualizándose también el enlace, hasta que sale del bucle while por la causa de que ya no hay más noticias que recorrer.

Se espera que de esta forma se haya entendido cómo se recorren las páginas del sitio web.

Cada vuelta que da el bucle while es una página que se ha recorrido, el funcionamiento sobre como se extrae la información es el siguiente:

a) Se inicializa el enlace de la página de noticias al que se quiere acceder.

b) Se extrae y parsea el HTML de dicha página utilizando BeautifulSoup.

c) Se llama al método 'extraer_titulosYenlaces()' que recorre el HTML y devuelve una lista con los títulos y otra con los enlaces.

d) Se hace extienden dichas listas a la lista global de títulos y enlaces, se hace uso de '.extend()', de esta se añade esta lista al final de la global que tenemos de este periódico.

e) Se recorre la lista de enlaces con los sitios webs de las noticias con un bucle for (todo esto dentro del bucle while) en el que se va llamando a al método 'extraer_noticia()' que devuelve el subtítulo, autor etc. de cada noticia.

f) Se añade una a una la información extraídas con la función '.append()'.

g) Se comprueba que se han recopilado títulos, sino se utiliza un 'break' para salirse del bucle while ya que significaría que hemos llegado al fin de las páginas con noticias.

h) Se incrementa en uno la variable 'pagina'

Una vez se ha salido del bucle while quedaría por finalizar el programa guardar todos los datos recopilados. Para hacerlo simplemente se llama al método creado en la clase abstracta 'guardar_csv() ' al cual se le pasa por cabecera el nombre de la carpeta donde se quiere guardar la información y el nombre que tendrá el archivo guardado.

Gracias a la clase abstracta creada el proceso de acceder a las variables es automático ya que con 'self.titulos', por ejemplo, en la propia clase abstracta, se accede a todos los títulos recopilados.

Se da por finalizado el primer periódico que se ha elegido para el sistema y se procede con el segundo.

6.6. Ideal Jaén

6.6.1. Introducción al Ideal Jaén

El Ideal Jaén es un periódico que fue fundado en 1932, por lo que cuenta con unos años de experiencia en este sector. Es un periódico que fué fundado en Granada por lo que muchas de sus noticias están enfocadas a la provincia de Granada. Sin embargo, las noticias que se publican en él hablan de Andalucía en general, y como no, de Jaén. Tiene una sección es específica para la provincia de Jaén en la que se suben diariamente multitud de noticias acerca de esta maravillosa provincia.

Al tener una sección de Jaén en muchas de sus noticias, como era de esperar, se habla además de la Universidad de Jaén, que es la identidad de la cual interesa obtener mucho más información.

6.6.2. Motivos para la la Elección del Ideal Jaén

1. **Calidad de las noticias:** Al tener tantos años de experiencia se nota un plus en la calidad de sus noticias. Se incluyen muchas imágenes, subtítulos que describen bien la noticia etc. Esta calidad de noticias es la deseada para el sistema de recolección de noticias.

2. **Estructura:** El periódico aporta una estructura clara y muy ordenada, lo que, por supuesto, facilita la tarea de recolección de noticias ya que todas las noticias tienen la misma estructura interna.

3. **Una sola página con noticias:** Como ventaja, además, no tiene como tal un conjunto de páginas que habrá que recorrer para recolectar las noticias, sino que tiene todas las noticias en una misma página, más tarde se hablará de esto.

4. **Sección en específica para la Universidad de Jaén:** El periódico tiene una página en específico dedicada a las noticias publicadas sobre la Universidad de Jaén lo que facilita mucho el poder recolectar las noticias al tener todas las que se desean en un mismo espacio.

6.6.3. Desafíos del Scraping del Ideal Jaén

A la hora de implementar el scraper, el Ideal Jaén ha supuesto mucho más problemas e inconvenientes que otros scraper que se han programado. Principalmente estos son los desafíos que han supuesto el correcto desarrollo e implementación del scraper de este periódico:

- **Uso de Anti-Scraping:** Al empezar con la programación del scraper se siguió la misma estructura de código y se utilizaron las mismas librerías que el anterior scraper del Diario Jaén.

  Tras ajustar el código a este nuevo periódico y solucionar varios errores, el scraper no recogía ninguna noticia, daba siempre error y no se tenía solución.

  Esto es una cosa bastante curiosa ya que en el anterior scraper funcionaba pero en este no, hasta que se llegó a la conclusión de que, al ser este un periódico de más calidad y con mucha mas audiencia a nivel andaluz, el periódico estaba usando un algoritmo Anti-Scraping que impedía que el programa pudiera recolectar las noticias que se deseaban.

  Para solucionar este gran problema se investigó por internet hasta dar con una solución muy efectiva, la utilización de librería 'Selenium', de la cual, en la implementación se hablará detalladamente de los beneficios que aporta esta librería al sistema de recolección de noticias.

- **Buscador interno mal optimizado:** Al empezar a investigar en la página web del periódico no se contempló la posibilidad de usar la sección que ofrecía el periódico para buscar noticias de la Universidad de Jaén, no se contempló esta posibilidad debido a que no se descubrió esta sección hasta una etapa media de la implementación del scraper.

  Es por esto que se buscaron otras alternativas, alternativas que ya se habían explorado y usado en el anterior scraper, dicha alternativa era la de hacer uso del buscador interno del periódico.

El desafío llega ahora, cuando se comprueba que el buscador, al hacer uso de él y agregar la consulta 'Universidad de Jaén', este no está optimizado y no recuperaba bien las noticias que eran de interés.

Al tener la experiencia con el anterior periódico se optó por usar las mismas consultas, para comprobar si de esta forma, el buscador funciona de mejor manera. Se utilizaron consultas como:

Figura 6.7: *Ilustración buscador Ideal Jaén*

Como se puede observar se hizo uso de las comillas " en el buscador para comprobar si mejoraba su eficacia, pero el resultado fue negativo, no recuperaba bien las noticias el buscador.

Se indagó y exploró más detenidamente en el sitio web y se encontró la maravillosa sección de la que se ha hablado con anterioridad. De esta forma, en una sola página se incluían todas las noticias relevantes sobre la Universidad de Jaén lo que ha facilitado mucho el trabajo de encontrar las noticias que verdaderamente son relevantes.

Cabe desatacar que para el correcto funcionamiento del scraper se han programado dos módulos:

```
extraerHTML_IdealJaen.py
idealJaen_scraper.py
```

Ambos utilizan la librería que se ha mencionado llamada 'Selenium'.

El primer script se encarga de de extraer el HTML de la única página que proporciona el periódico con las noticias recuperadas. Se encarga de ir aumentando el HTML clicando automáticamente en un botón que dice 'Ver más noticias'. Una vez no hay más noticias se extrae el HTML y se almacena en memoria para ser tratado por el siguiente script. Esto se verá mas detalladamente más tarde.

El segundo script se encarga de extraer la información del HTML, así como títulos, enlaces, cuerpo de la noticia etc. para posteriormente cargarlo en memoria como se ha hecho en el periódico anterior con el método 'guardar_csv()'.

Vayamos ahora con la implementación la expoliación detallada de la implementación del scraper, se dividirá en dos partes, la primera se explica el código referente

al script llamado 'extraerHTML_IdealJaen.py' y la segunda parte al script 'idealJaen_scraper.py'.

6.6.4. Implementación del Scraper Ideal Jaén - 1er script

Se empieza detallando el código del primer script.
El script consta de los siguientes métodos y librerías:

- **Librerías importadas:** Estas son las librerías que se han importado, ***enlace***.

  La mayoría de las importaciones que se han hecho son de la librería 'Selenium'. Todas estas importaciones se usan y son necesarias.

- **Método cargarEstructurasPickle(elemento, carpeta, nombre):** El código implementado está en el siguiente ***enlace***.

  Esté método hace una función parecida al método ya antes comentado 'guardar_csv()'. Este método en líneas generales se encarga de cargar en memoria un elemento. Se le pasa como parámetros el elemento que se quiere guardar, el nombre de la carpeta donde se desea almacenar y el nombre que tendrá el archivo almacenado.

  El archivo almacenado tendrá la misma información que la variable elemento, pero estará almacenado en memoria en formato 'pickle', se ha elegido este tipo de formato por los siguientes motivos:

  - **Persistencia de Objetos Complejos:** La librería 'pickle' permite guardar objetos complejos de Python para poder cargarlos y seguir utilizándolo en otro momento, que es justo lo que se quiere.
  - **Compatibilidad con la mayoría de los tipos de datos de Python:** Pickle permite almacenar casi cualquier tipo de dato nativo en Python, por lo que, si por cualquier cosa el código se tuviera que actualizar con nuevos y diferentes contenedores, Pickle seguiría poder usándose.
  - **Eficiencia:** El tiempo que tarda el proceso existente para almacenar la variable en memoria es bastante rápido, lo que convierte a Pickle en una opción eficiente para guardar variables con mucho contenido, en el caso del script, para guardar el gran contenido que tiene el HTML de la página.
  - **Facilidad de Uso:** En apenas unas líneas de código se pueden almacenar y cargar variables en memoria. Se hace de una forma muy sencilla y directa lo que facilita el uso de esta librería.

  Se procede con el otro método implementado en este script en el cual se hace uso de esta función.

- **Método main():** El código programado a continuación se muestra en el siguiente *enlace*.

  En este método se hace un gran uso de la librería 'Selenium', por lo que antes de adentrarse en explicar el código se expondrán los beneficios y motivos por los que se ha optado por usar 'Selenium' para extraer el código HTML de la página web del periódico.

  Se muestran los **beneficios** a continuación:

  - **Automatización de Navegadores:** Selenium Permite automatizar tareas repetitivas sobre navegadores web, en el caso del sistema Chrome, lo que ahorra una cantidad significativa de tiempo y sobretodo esfuerzo.
  - **Interacción con Elementos de la Página:** Esta librería proporciona métodos para interactuar con varios elementos de una pagina web, un ejemplo en el sistema sería el uso de un botón, permitiendo realizar acciones como clics, envíos de formularios e incluso navegación.
  - **Esperas Explícitas e Implícitas:** Selenium aporta esperas explicitas e implícitas, lo que ayuda a manejar el comportamiento asincrónico de las aplicaciones web y se aseguro de esta forma que los elementos estén disponibles antes de interactuar con ellos para poder evitar algunos errores.
  - **Desplazar la vista del navegador:** A demás, Selenium permite controlar la vista del navegador, subiendo o bajando, mostrando en todo momento las interacciones que se hacen y evitando así otros errores por falta de que no aparezca el elemento en la vista.

  Una vez se ha hablado de los beneficios que ofrece esta gran librería, se detallan a continuación, los **motivos** por los cuales se ha optado por usar 'Selenium', más enfocado a la implementación del sistema.

  - **Automatización del Proceso de Navegación y Extracción de Datos:** El método necesita abrir una página web en específico, concretamente la página web a la que redirige el buscador cuando se selecciona que recupere las noticias relacionadas con el tema de la Universidad de Jaén.

    Aceptar cookies y hacer clic repetidamente en el botón 'Ver más noticias' es una de las acciones que se pueden automatizar, evitando la necesidad de tener que interactuar con la página web manualmente.
  - **Manejo de Interacciones Dinámicas:** La página contiene elementos interactivos y, como no, dinámicos, como el botón mencionado anteriormente 'Ver más noticias', que solo se vuelven clicables cuando han ocurrido ciertos eventos.

'Selenium' es capaz de manejar todas estas interacciones de manera eficiente utilizando esperas implícitas, y la forma de implementarlo es muy sencilla.

- **Obtención del HTML Completo de la Página:** Una vez que se ha cargado toda la página, esto quiere decir que se están mostrando todas las noticias y ya no existe el botón 'Ver más noticias', la librería permite obtener el HTML completo de la página, para su posterior procesamiento y extracción de datos en otro script.
- **Manejo de Excepciones y Continuidad del Proceso:** Gran parte del proceso de implementación se ha pasado solucionado errores que hacían que se cortara la ejecución del programa sin saber exactamente por qué.

  Selenium permite capturar la excepciones de una forma sencilla, haciendo, por ejemplo, que el navegador siguiera abierto cuando saltaba una excepción. De esta forma investigando en el navegador se pueden solucionar la mayoría de errores entendiendo el por qué de estos.

Una vez se ha hablado de las ventajas de usar esta librería, se procede con la explicación de código que se ha implementado.

El método main es el más importante del script ya que ejecuta todas las operaciones para poder extraer el HTML correctamente y poder cargarlo en memoria.

Lo primero es inicializar las variables para que el método funcione correctamente, así como inicializar la ruta donde está ubicado el WebDriver.

El WebDriver es un componente de 'Selenium', una suite de herramientas para la automatización de navegadores Web. El WebDriver actúa como un controlador el cual es cpaz de automatizar las interacciones que tiene 'Selenium' con el navegador.

Cada versión de Chrome tiene su respectiva versión de WebDriver por lo que es importante descargar aquella que es compatible con la versión que utiliza el navegador.

Una vez se configura el servicio, inicializando a ruta del WebDriver, se inicia este controlador. A continuación, todo el código estará en un bloque 'try' para poder capturar las excepciones en el caso en el que ocurra algún imprevisto.

Se inicializa el enlace con 'driver.get(enlace)' de forma que el navegador accede al navegador en el enlace que se le proporciona. Una vez abre la página web deberá aceptar las cookies haciendo en el botón 'Aceptar' para poder navegar y seguir recuperando noticias sin problemas.

Se ha añadido una funcionalidad al código para que espere a que este botón de aceptar cookies esté disponible con 'WebDriverWait', de forma que hasta que el botón no aparezca en pantalla no se la haga clic. Esto facilita mucho el trabajo y la resolución de errores.

Se crea un bucle forma que va de 0 a 500, que se encarga de hacer clic en el botón 'Ver más noticias', se ha puesto como límite 500 ya que más de 500 veces no se le hará clic al botón porque éste desaparecerá mucho antes al no haber más noticias que recolectar.

En esta parte del código surgieron varios problemas por lo que la implementación para hacer clic en este botón es la siguiente:

- Se asegura con la función anteriormente mencionada 'WebDriverWait' que el botón se le pueda hacer clic.
- Una solución a un error persistente que ocurría era poder bajar y mover la vista del navegador al botón mediante el módulo 'ActionChains'. Este módulo permite mover la vista directamente a un elemento, en este caso al botón.
- Otra solución a otro error que no permitía hacer clic normalmente, como el botón de aceptar las cookies, es la de ejecutar directamente el script del código ver más noticias. Cuando se hace clic en un botón lo que se hace es ejecutar el código que lleva implícito ese botón, bien, al no poder hacer clic directamente al botón la solución es ejecutar el script que lleva asociado dicho botón con la función '.execute_script'.

De esta forma se ejecuta el botón 'ver más noticias' hasta que no queden más noticias.

Una vez desaparezca el botón por que no hay más noticias, se captura la excepción que se produce y se guarda el código HTML mediante la función '.page_source'. Una vez guardado en la variable, se almacena en memoria con la llamada al método que se ha comentado antes '.cargarEstructurasPickle()'.

Esta ha sido la explicación al código implementado que almacena el código HTML en memoria. Se procede con la explicación del script que se encargada de procesar este código y extraer de él la información que se desea.

6.6.5. Implementación del Scraper Ideal Jaén - 2o script

Se procede con la explicación del código programado para la segunda parte del scraper:

- **Librerías importadas:** Las librerías utilizadas son las ***siguientes***.

  Todas estas librerías son imprescindibles en el código programado. En el código se hará mención a ellas.

- **Método extraer_titulosYenlaces(soup):** Este el código implementado, ***enlace***.

  Este código, como se puede observar, es muy similar al código implementado en el scraper correspondiente al Diario Jaén. Las únicas variaciones visuales que puede haber

en dicho código es los contenedores de donde se extrae la información de interés en el periódico.

El funcionamiento del código es el mismo que para el anterior scraper, extrayendo información de los diferentes contenedores y almacenándola en dos listas, una para los títulos y otra para los enlaces. En el caso que haya un error inesperado, se le adjudicará al título o enlace el texto 'ERROR'.

Esto a la hora de procesar las noticias recopiladas ayuda, ya que es facilita la eliminación de aquellas en las que ha surgido algún tipo de error y la información no está bien recogida.

A diferencia del método implementado en el scraper del Diario Jaén, este método se llamará solo una única vez, ya que el sitio web muestra en una sola página todas las noticias que se han recuperado. En el scraper del Diario Jaén se llamaba al método tantas veces como páginas habría que recorrer para la recolecta de noticias.

Una vez recopilados todos los títulos y enlaces, se devuelven las listas correspondientes que los contienen.

- **Método extraer_noticia(enlace):** A continuación, se muestra el ***código***.:

  Este es un método con un código similar al periódico anteriormente DiarioJaén.

  Este método, a diferencia del implementado en el otro periódico, se le pasa por cabecera directamente el código HTML, no un enlace.

  Este código HTML es el código de la pagina web de la noticia, por lo que, simplemente como se ha hecho anteriormente, se extrae la información que se desea investigando detenidamente el código HTML observando los contendores donde se encuentra dicha información.

  El método devuelve 4 String, con la información siguiente información extraída:

  - subtítulo.
  - fecha.
  - autor.
  - cuerpo de la noticia.

  De forma que si se produce algún error se devolverá el String 'ERROR'.

- **Método main():** Se expone el código implementado en el siguiente ***enlace***.:

  Este es el método más importante ya que se encarga de todas las operaciones, como cargar el HTML previamente almacenado en el anterior script y procesarlo extrayendo información recorriéndolo.

Lo primero es cargar el HTML con la misma librería con la que se almacenó, se está hablando de la librería 'pickle', mediante la función 'pickle.load(file)' se carga el HTML para procesarlo a continuación.

Como en los métodos anteriores, para procesar el HTML, primero es necesario parsearlo. Una vez parseado se le pasa por cabecera al método 'extraer_titulosYenlaces(soup)'.

Este método recorre el HTML parseado y devuelve la lista con los enlaces y títulos de las noticias que se han recuperado.

Esta listas se extienden a las variables globales con '.extend()'.

Para poder recorrer las noticias recuperadas, no es posible hacerlo pasándole el enlace directamente a la función 'extraer_noticia()' ya que, como se comenta en el anterior script, este periódico cuenta con sistemas Anti-Scraping y no es posible extraer el código HTML directamente como en el anterior periódico.

Para solucionar este problema se vuelve a optar por el uso de la librería 'Selenium'. De la misma forma que en el anterior script, se configura el servicio con la ruta del WebDriver para poder usar el controlador de Chrome.

Una vez está todo inicializado se recorre la lista enlaces, y con las líneas de código 'driver.get(enlace)' y 'html = driver.page_source' se extrae el html de cada noticia recuperada.

Una vez se tiene el código HTML simplemente se le pasa al método 'extraer_noticia(html)' y se obtiene la información de interés. Se añaden los String devueltos por esta función a las variables globales con '.append()'.

Una vez se ha terminado el bucle, la ejecución del código se deriva al bloque 'finally' y finalmente se guarda la información recopilada con el método 'guardar_csv()'.

6.7. Hora Jaén

6.7.1. Introducción al Hora Jaén

El periódico Hora Jaén es un periódico Local con fuerte relevancia en Jaén. Es conocido por la facilidad que tiene para poder informar sobre acontecimientos importantes y relevantes en esta provincia. Hoar Jaén, como los anteriores periódicos que se han mencionado, trata sobre una amplia cantidad de temas, desde política, economía cultural etc.

Hora Jaén, una de las ventajas que tiene, es que toma muy en cuenta a la Universidad de Jaén a la hora de publicar las noticias, lo cual es justo lo que se necesita para el sistema de recolección.

Este proporciona información detallada sobre las noticias que publica y cuida al lector ofreciéndole una visión cercana a ellas.

6.7.2. Motivos para la la Elección del Hora Jaén

Los motivos por lo que se ha optado por la elección del Hora Jaén como periódico para recolectar sus noticias son los siguientes:

- **Variedad de secciones:** Como se ha comentado anteriormente, el periódico Hora Jaén ofrece muchos tipos de temas en sus noticias, por lo que se publican noticias sobre la Universidad de Jaén de cualquier tipo de tema.
- **Cobertura de la Universidad de Jaén:** Como no, este periódico pone el punto de mira en la Universidad de Jaén, centrándose en ella para muchas de las noticias que se publican.
- **Calidad Periodística:** Como los anteriores periódicos, este no se iba a quedar atrás, dispone de una calidad periodística que muchos periódicos envidan, esto contribuye a que sea un periódico relevante, de calidad y fiable.
- **Experiencia y Trayectoria:** El Hora Jaén es unos de los periódicos con mayor trayectoria y experiencia en este sector, esto añade un punto de confianza y credibilidad a sus lectores, lo que es perfecto para mostrar las noticias en el dosier.

6.7.3. Desafíos del Scraping del Hora Jaén

La recolección en este periódico no ha sido lo fácil que se esperaba, estos son los desafíos que se han abordado para poder implementar el scraper del periódico Hora Jaén:

- **Buscador:** Como en todos los periódicos, el buscador a jugado un papel importante, pero en ninguno ha funcionado del todo bien.

  En este periódico, el buscador no funciona correctamente ya que se hicieron múltiples consultas pero no se obtuvieron resultados relevantes al respecto.

  Ya que se ha tenido experiencia en este tipo de problemas, se optó por usar las mismas soluciones que se utilizaron en los anteriores periódicos. De forma que se empezo a hacer uso de las comillas " para hacer la siguiente consulta 'Universidad de Jaén'.

  Al hacer este tipo de consultas, con las comillas, se obtuvieron resultados relevantes por lo que el scraping era posible utilizando el buscador interno del periódico y haciendo este tipo de consultas.

- **Tiempos de acceso largos:** Uno de los principales problemas que tiene este periódico es los tiempos de acceso que tiene para su página web.

  A diferencia de los demás periódicos, este para acceder a su página web se es capaz de tardar hasta 5 veces más. Es por eso que el scraping en este periódico se ha tardado bastante más.

  Se une la cantidad de noticias de la Universidad de Jaén que tiene con el tiempo que hay que esperar para acceder a cualquiera de sus sitios web.

- **Estructura HTML:** A la hora de recorrer las páginas principales y obtener los títulos y enlaces de las noticias ha habido problemas.

  En el código HTML hay implícitas algunas noticias que no son de interés pues no tratan sobre la Universidad de Jaén, Estas noticias se encuentran en el menú desplegable que ofrece el periódico.

  La solución para no recopilar esas noticias que no son de interés es investigar a fondo el código HTML para acceder a los contenedores correctos. Tras varias pruebas y analizar el código se encontraron los contenedores que contienen las noticias deseadas por lo que se ignoró el contenedor global y se centró en uno más específico para poder solucionar este problema.

6.7.4. Implementación del Scraper Hora Jaén

La implementación del scraper para este periódico es muy parecida a la del Diario Jaén, el periódico Hora Jaén tiene la misma estructura web por lo que el scraper tiene métodos muy similares.

Por esta razón no se explicará tan detalladamente como en los anteriores periódicos la implementación seguida.

- **Librerías importadas:** Las librerías utilizadas son las ***siguientes***.

  Todas estas librerías ya se han visto y detallado a lo largo de esta memoria del proyecto.

- **Método extraer_titulosYenlaces(self, soup):** Este es su respectivo ***código***.

  El código de este método es similar al de los otros periódicos, lo único que difiere es los contenedores donde se encuentran las noticias que se buscan, como se ha mencionado anteriormente, en esté periódico se ha tenido que buscar contenedores más específicos.

- **Método extraer_noticia(self, enlace):** El código implementado se encuentra en el ***enlace***.

  En este método no se programa nada nuevo en comparación con los otros dos scrapers por lo que se procede con el siguiente método.

- **Método main():** Se expone el ***código***.

  Cabe destacar que la estructura que tiene este periódico, en cuanto a noticias recuperadas, es la misma que la estructura del Diario Jaén. Cuando se recuperan las noticias, estas, en vez de estar en una misma página como el caso del Ideal Jaén, están esparcidas en varias páginas.

  Por lo que para recuperar todas las noticias relevantes sobre la Universidad de Jaén hay que recorrer cada una de las páginas con las noticias que se recuperan. Esto, como se ha comentado con anterioridad, se hace exactamente de la misma forma que se ha hecho con el periódico Diario Jaén.

  Una diferencia existente es el tiempo que tarda en recuperar un periódico y otro, el Hora Jaén, al tener más noticias y tardar más en cuanto a tiempo de acceso a las páginas se refiere, el tiempo total de recopilación es mucho mayor que si se compara con el del Diario Jaén.

  Tras indagar más sobre las noticias que recuperaba este periodico, se comprendió que buscaba diferentes noticias con la consulta 'Universidad de Jaén' y ' UJA '. Por lo que para asegurarse de que se recopilan todas las noticias relacionadas con la Universidad se recorren todas las noticias que recuperan estas dos consultas.

  El primer while se corresponde con las noticias recuperadas a partir de la consulta 'Universidad de Jaén', y el segundo con las noticias que se recuperan a partir de la consulta 'UJA'.

  En este periódico se recoge una gran cantidad de noticias al poder recopilar de esta forma con dos consultas en el buscador interno del periódico.

  Para las noticias repetidas (es posible que una noticia se recupere tanto con una consulta como para la otra) se eliminarán en un script que se detallará más tarde encargado de filtrar y limpiar todas las noticias recopiladas.

Resultados Scraping

Una vez ejecutados los tres scrapers se han recopilado en torno a 13800 noticias.

La ejecución de los scrapers tiene un tiempo total de recolección de unas 6 horas en recorrer y almacenar todas las noticias de los tres periódicos, dos horas de media para cada scraper.

Todas las noticias recopiladas no son válidas, ya que puede haber habido errores de recolección, noticias sin información etc.

Para tener en cuenta solo las noticias que son de interés se le aplica un preprocesamiento a las noticias para dejarlas listas para su clasificación.

En este apartado se ha detallado como el sistema es capaz de recolectar noticias y almacenar su información relevante por lo que se referencia al requisito funcional que especifica esta funcionalidad, requisito número 1, en el siguiente ***enlace***.

6.8. Implementación Preprocesamiento

Una vez recopiladas las noticias, hay que hacer un procesamiento de todas estas para el correcto funcionamiento del sistema.

Para ejecutar dicho procesamiento se ha implementado una clase que consta de los siguiente métodos:

- **combinar_csv**.
- **filtrar_noticias_anomalas**.
- **convertir_fechas**.
- **procesar_texto**.

Una vez nombrados los métodos se procede con la explicación de cada uno de ellos.

- **combinar_csv:** Este método, el cual se encuentra en el siguiente ***enlace***, se crea con la finalidad, como su propio nombre indica, de combinar los tres archivos en formato '.csv' de cada uno de los periódicos en un archivo solo.

  El funcionamiento del método es muy sencillo, consta de los siguientes pasos:

  1. Leer los tres archivos '.csv' y guardar cada uno de ellos en una variable.
  2. Se inserta en una sola variable cada uno de las tres variables mencionadas arriba, con la función '.append()'.
  3. Se combina los archivos con la función 'pd.concat()' que ofrece pandas.
  4. Se asigna al atributo dataframe de la clase el dataframe combinado.

  De esta forma se combinan los tres archivos en un solo y se le asigna al atributo de la clase.

- **filtrar_noticias_anomalas:** El código implementado se encuentra en el siguiente ***enlace***.

  Este método se encarga del procesamiento de las noticias de forma que elimina del dataframe aquellas noticias que se consideren anómalas.

Se consideran noticias anómalas aquellas que cumplen una de las siguientes características:

- Tener en la columna 'Fuente' el valor 'HoraJaén' y estar repetida.
- Tener algún campo vacío.
- Un texto de noticia de menos de 100 caracteres.
- El texto de noticia tiene más del 80 % de palabras repetidas.

El funcionamiento es simple, y es el siguiente:

1. Eliminar del dataframe las noticias repetidas del HoraJaén.
2. Crear la función que determinará si una noticia es o no anómala, devuelve True o False en el caso en el que lo sea o no dependiendo de las características que tenga ésta.
3. Crear un campo adicional llamado 'Anomala' y llamar a la función, previamente creada, para discernir las noticias entre las que son anómalas y las que no.
4. Eliminar aquellas noticias que tienen valor True en el nuevo campo creado.
5. Eliminar el nuevo campo del archivo filtrado
6. Asignar este nuevo dataframe sin noticias anómalas al adataframe de la clase.

Como resultado se obtiene un archivo filtrado con las noticias que realmente interesan.

- **convertir_fechas:** Se expone el código en el siguiente ***enlace***. En la interfaz de usuario, una característica indispensable que debe tener es poder filtrar por fechas.

  Para poder llevar esto a cabo es necesario que todas las noticias tengan un formato de fecha estándar. Hasta ahora simplemente se tenían strings en el campo 'Fechas', las cuales todas no estaban en el mismo formato y mucho menos en el formato estándar.

  Por lo que este script se encarga de transformar el formato de las fechas del archivo '.csv' a un formato estándar, de forma que se pueda filtrar por ellas sin ningún inconveniente en la interfaz que se implementará.

  Para la resolución de este problema se utilizan funciones como '.split()' , 'str()' e incluso un diccionario con los meses del año. Obviamente se hace uso de la librería pandas.

  Se han tenido en cuenta todos los tipos de formatos diferentes de fechas que se han recolectado, en total se han tenido que tener en cuenta 5 tipos de formatos por lo que se ha tenido que hacer pasar a un formato estándar cada uno de estos 5 formatos diferentes.

  De esta forma se consigue el objetivo de pasar todas las fechas al mismo formato estándar.

- **procesar_texto:** Es importante preprocesar el texto ya que:
  - Reduce el ruido para no tener en cuenta caracteres que son irrelevantes para la clasificación.
  - Normalización del texto para convertir todas las palabras a una forma común y facilitar la tarea del clasificador para que se produzcan tópicos mas claros.
  - Mejora la coherencia temática ya que se evita que las variaciones ortográficas o gramaticales distorsionen los temas.

  El preprocesamiento que se ha hecho es el siguiente:

  - **Dividir el texto en tokens:** Para poder manejar las palabras (que son tokens) más fácilmente.
  - **Eliminar palabras vacías:** Se hace con la librería NLTK con los módulos 'punkt' y 'stopwords'.
  - **Aplicar un lemmatizer:** Concretamente se aplica el WordNetLemmatizer, también de NLTK, para poder reducir las palabras a una forma común. Se utilizó en un principio un stemmer, para ser exactos el SnowballStemmer, pero se llegó a la conclusión de que el algoritmo funcionaba mucho mejor con el lemmatizer ya que los tópicos eran mucho más coherentes.

  De esta forma se queda el texto limpio para poder aplicar de una manera mucho más eficiente la clasificación a las noticias.

  En esté método se utilizan técnicas de PLN para poder procesar los textos, por lo que se hace referencia al requisito funcional número 4, el cual especifica esta función en este ***enlace***.

6.9. Implementación Clasificación

Se procede con la explicación de los scripts cuyo objetivo es clasificar las noticias. Ya en un capítulo anterior se ha hablado y mencionado sobre el tipo de clasificador que se ha usado para poder clasificar las noticias en varios temas, el algortimo LDA.

En esta sección se hablará de como se ha realizado la implementación, así de los métodos usados y de cada uno de los dos scripts implementados.

El script que se ha programado es **Clasificacion.py**

Se detallará a continuación el propósito y la explicación del código del script:

Clasificacion.py: Este script es importante, y es el que se encarga específicamente en la clasificación de las noticias.

Las librerías usadas están en el siguiente ***enlace***. Se utilizan librerías para clasificar y evaluar el modelo como son la librería 'gensim' y otras librerías encargadas de procesar el texto de las noticias para que sea más facil al clasificador hacer su función.

Tiene un único método el cual es el main() y consta de los siguientes pasos:

1. Cargar el archivo CSV con todas las noticias recopiladas.
2. Crear un diccionario y un corpus con esta nueva columna.
3. Se aplica el modelo LDA.
4. Mostrar los tópicos que se han descubierto.
5. Según los tópicos se asignan categorías a cada noticia.
6. Se borra esta nueva columna ya que no es necesaria.
7. Almacenar el nuevo CSV en memoria. |
8. Evaluación del modelo.

Sobre estos pasos se detallarán algunas cosas que son importantes de mencionar sobre el último apartado el cual es el de evaluación:

Evaluación del modelo: Esta parte se encarga de según las noticias clasificadas evaluarlas con unas métricas implementadas, estas métricas son:

- **Coherencia:** Con la librería 'gensim' la cual es posible importar el módulo CoherenceModel para utilizar esta métrica.
- **Perplejidad:** La cual es proporcionada por el propio algoritmo LDA con la función '.log_perplexity()' y con el corpus almacenado.
- **F1-Score:** Esta métrica se ha implementado manualmente, haciendo los cálculos de las 50 noticias que se han cogido y clasificado manualmente, por lo que carece de código asociado.

Asignación automática de categoría: Cabe destacar que este algoritmo lo que hace es asignar un nº a cada noticia según la clasificación que se le haya asignado. Este número irá desde 0 hasta el número de categorías que se desean obtener, en este caso 9 categorías, menos uno.

Al usuario, no se le puede mostrar como categorías números, por lo que se ha creado este método que automáticamente es capaz de, según los tópicos de cada tema, asigna un título a cada categoría. De forma que, por ejemplo, si el programa detecta que la categoría número

3 los tópicos son 'aceite' y 'olivar' se le asignará directamente la categoría 'Aceite de Oliva y Salud'

Se ha hecho mención de esto más arriba, justo ***aquí***.

Esto se ha implementado mediante un diccionario de categorías y la función 'get_category_by_keywords'.Esta función busca las mejores coincidencias entre las palabras clave generadas por el modelo LDA y las palabras clave asociadas a cada categoría. Por lo que de esta forma la categoría con más coincidencias es seleccionada.

Este script completa el requisito funcional número 4, el cual detalla la existencia de la clasificación en el sistema, se encuentra en el siguiente ***enlace***.

Resultados obtenidos de la clasificación

Las noticias clasificadas se pueden visualizar en el siguiente gráfico:

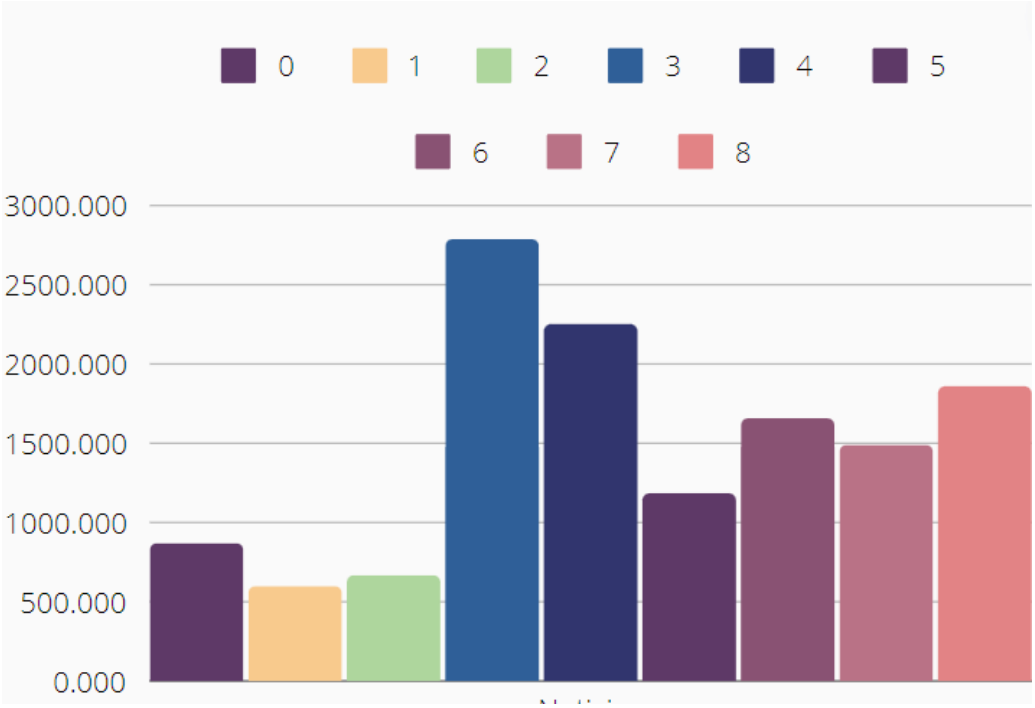

Figura 6.8: *Gráfico Resultados Clasificación*

Cada número se asigna a un tema, para visualizarlo mejor se deja el número y el tema que se asocian:

```
0: "Patrimonio y Cultura",
1: "Investigación y Proyectos Universitarios",
2: "Aceite de Oliva y Salud",
```

```
3: "Cultura y Premios",
4: "Provincia y Desarrollo Internacional",
5: "Estudiantes y Educación",
6: "Otras categorías",
7: "Financiamiento y Proyectos Regionales",
8: "Programas y Actividades Sociales"
```

Como se puede observar todas los temas tienen un gran número de noticias clasificadas, el tema que tiene más noticias es el de 'Cultura y premios' que tiene bastante sentido ya que la mayoría de noticias que se publican acerca de la universidad es sobre eventos que se hacen y cultura en general.

De media, cada categoría tiene unas 1500 noticias clasificadas, lo cual está bastante bien teniendo en cuenta el gran volumen de datos que se está manejando. Sí que es verdad, que hay algunas categorías que no pasan de las mil noticias clasificadas, esto sugiere que los periódicos se centran más en publicar noticias de otras categorías ya que son de mayor interés para los lectores, y estas categorías las 'olvidan' un poco más.

Una vez se obtiene el archivo final, con todas las noticias recolectadas, limpiadas y clasificadas se procede con la conexión a la base de datos

6.10. Conexión MongoDB

Como se ha mencionado en anteriores apartados, se ha optado por hacer uso de MongoDB por los motivos expuestos anteriormente.

Por lo que en este apartado se mostrará como se ha llevado a cabo la implementación del script encargado de conectar el archivo 'CSV' con la base de datos.

Se creado la respectiva base de datos llamada 'Sistema' y una colección en su interior con nombre 'noticias', en esta colección se almacenarán todas las noticias y sus campos.

El script programado llamado 'ConectarBBDD.py' se encuentra en el siguiente ***enlace***.

Como se puede observar, lo primero que se hace es cargar el archivo csv en memoria, una vez cargado se procede a pasar todas las noticias al formato que mongo admite. Esto se hace de esta forma y no se ha hecho en un script anterior como 'FechasAFormato.py' ya que si se almacena en memoria MongoDB no lo detectará correctamente y se introducirá la fecha como string en vez de como datetime.

Una vez pasadas las fechas al formato que reconoce MongoDB se procede con la conexión a la base de datos, simplemente se inicializa la ruta donde está alojada dicha base y la emparejamos con la bbdd y con la colección antiormente mencionadas.

Una vez emparejada la base de datos se pasa a un diccionario el archivo 'CSV' y con la función 'insert_many' se insertan todas las noticias automáticamente en la base de datos.

La forma usada para comprobar que todo esté correcto y en funcionamiento ha sido con el software de **MongoDBCompass**, a continuación se muetra una imagen de como se ve la base de datos en dicho software:

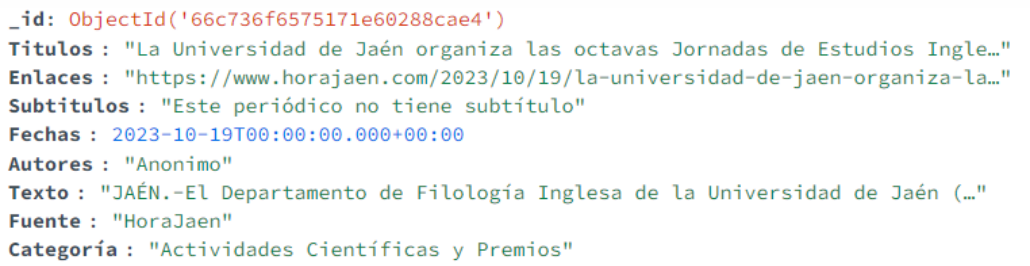

Figura 6.9: *Imagen ejemplo MongoDBCompass*

De esta forma se almacenan todas las noticias recopiladas en una base de datos en MongoDB.

6.11. Interfaz web

Como se ha comentado con anterioridad en el segundo capítulo, la interfaz web se ha implementado visualmente mediante HTML y CSS.

Para que funcione la interfaz a la hora de filtrar las noticias, cargarlas y mostrarlas se ha utilizado javascript por su facilidad para manejar los valores que introduce el usuario en el HTML, procesarlos y modificar el código HTML añadiendo lo que se desee, en este caso, lo que se añaden son las noticias con su infotrmación respectiva.

En Javascript se han desarrollado dos script:

- **script.js**
- **server.js**

No se adentrará mucho en la implementación de cada script por lo que se explicará que hace cada uno a rasgos generales:

El primer script llamado **script.js** se encarga de lo que se acaba de comentar, recoger lo que el usuario introduce en el formulario y modificar el HTML para mostrar las noticias que se han recuperado. Este script es el que se comunica con la resAPI, enviandole toda la información que genera el usuario en la interfaz web.

El segundo script llamado **server.js** se encarga de lanzar el servidor por lo que es capaz de conectarse con MongoDB, se conecta a la base de datos donde se tienen almacenadas las noticias, y según los parámetros que ha introducido el usuario en el formulario y que le ha

pasado el anterior script filtra en la base de datos de Mongo, recupera las noticias que son de interés y se las pasa de nuevo al primer script que modifica el HTML. Este archivo se encarga de alojar la resAPI y se conecta con la base de datos y con la interfaz como se acaba de explicar.

La resAPI se ha implementado con Express, Nodejs y core, por lo que está totalmente desacoplada de la interfaz que es justo lo que se quiere y necesita en este proyecto.

El conjunto de los dos scripts permite poder filtrar, recuperar y mostrar la información de las noticias de interés por lo que se referencia a los dos requisitos funcionales que lo especifican, requisito 2 y 3, justo ***aquí***.

Si es de interés saber el código de ambos script se puede encontrar en este ***enlace*** y en ***este*** respectivamente.

De esta forma, queda totalmente desacoplada la interfaz del backend ya que es posible modificar el código HTML y CSS sin necesidad de tocar nada referente al servidor. Si se necesita modificar en un futuro por cualquier causa se convierte en una tarea fácil de realizar de esta forma.

6.12. Requisitos hardware para ejecutar el programa

El programa ha sido diseñado para ser altamente compatible y eficiente, lo que permite su ejecución en prácticamente cualquier ordenador. No se requieren características específicas de hardware, por lo que podrá funcionar sin problemas en sistemas con las siguientes especificaciones mínimas:

- **Procesador:** Cualquier procesador moderno de 1 GHz o superior.
- **Memoria RAM:** Un mínimo de 2 GB de RAM, aunque se recomienda 4 GB para un rendimiento óptimo.
- **Espacio en disco:** El programa ocupa menos de 100 MB de espacio en disco, por lo que cualquier unidad de almacenamiento con suficiente espacio disponible será suficiente.
- **Tarjeta gráfica:** No es necesario contar con una tarjeta gráfica dedicada; el programa puede ejecutarse utilizando gráficos integrados.
- **Sistema operativo:** Debido a que usa chromeDriver para windows es recomendable usar windows 64, si se desea usar en otra tipo de sistema simeplemnte con cambiar y adaptar el chromedriver sería suficiente.

Gracias a su diseño optimizado, el programa puede ejecutarse de manera eficiente incluso en ordenadores de gama baja o antiguos.

6.13. Guía de Usuario

6.13.1. Introducción

Se procede con una breve explicación de la interfaz web de noticias, diseñada para ofrecer una experiencia personalizada en la búsqueda y visualización de noticias.

6.13.2. Página Principal: Formulario de Filtrado

Filtrado de Noticias

- **Filtros Disponibles:**
  - **Fuente:** Selecciona la fuente de la que deseas obtener noticias Ideal Jaén, Hora Jaén o Diario Jaén.
  - **Autor:** Filtra las noticias por autor específico.
  - **Fecha:** Define un rango de fechas para ver noticias publicadas en ese periodo.
  - **Categoría/Palabras clave:** Filtra por categorías según las noticias de su interés.

  No es necesario rellenar todos los filtros, solo los filtros que se desee.
- **Aplicar Filtros:** Una vez configurados los filtros según tus preferencias, haz clic en el botón **"Buscar"**. Esto actualizará la lista de noticias mostrada en la página.

6.13.3. Lista de Noticias Filtradas

Las noticias que coinciden con tus filtros se mostrarán en una lista. Cada entrada de noticia incluye un resumen breve para ayudarte a identificar rápidamente las noticias de interés.

6.13.4. Visualización de Detalles de una Noticia

- **Seleccionar Noticia:** Haz clic en el título de cualquier noticia para ver los detalles completos.
- **Página de Detalles:** En esta página, podrás leer el contenido completo de la noticia y detalles relevantes.

6.13.5. Navegación

- **Volver a la Página Principal:** En la página de detalles de la noticia, encontrarás un botón **"Volver a los resultados"** que permitirá regresar a la página principal donde se encuentra el formulario de filtrado y la lista de noticias.

6.13.6. Consejos y Sugerencias

- **Optimización de Filtros:** Para una búsqueda más efectiva, utiliza varios filtros combinados para afinar los resultados, es decir, filtra por 'Fecha' y por 'Categoría' a la vez.
- **Explora Diferentes Fuentes:** Si no encuentras lo que buscas en una fuente, prueba con otras disponibles en el filtro.

Capítulo 7

Conclusiones

En este proyecto, se ha desarrollado un sistema integral para la recopilación, clasificación y visualización de noticias relacionadas con la Universidad de Jaén. A lo largo de las distintas fases del proyecto, se han abordado múltiples desafíos técnicos y conceptuales, logrando implementar una solución robusta y eficiente.

1. Recopilación de Noticias: El sistema de scraping desarrollado ha demostrado ser eficaz en la recolección de noticias desde diversas fuentes. Se han implementado técnicas avanzadas para manejar grandes volúmenes de datos y adaptarse a cambios en la estructura de los sitios web, garantizando así la continuidad y fiabilidad del proceso de recolección.

2. Clasificación Automática: La utilización de técnicas de Procesamiento de Lenguaje Natural (PLN) y modelos de aprendizaje automático ha permitido clasificar las noticias de manera precisa y eficiente. El modelo LDA ha sido fundamental para identificar y agrupar temas relevantes, mejorando la organización y accesibilidad de la información.

3. Evaluación del Sistema: Se ha establecido un marco de evaluación riguroso para medir la efectividad del sistema de clasificación. Las métricas de coherencia de tópicos y perplejidad han proporcionado una visión clara de la calidad del modelo, permitiendo ajustes y mejoras continuas.

4. Interfaz de Usuario: El diseño de la interfaz de usuario ha sido cuidadosamente alineado con la identidad visual de la Universidad de Jaén3. La implementación de una interfaz intuitiva y fácil de usar ha mejorado significativamente la experiencia del usuario, facilitando la navegación y el acceso a las noticias.

5. Desafíos y Soluciones: Durante el desarrollo del proyecto, se han enfrentado varios desafíos, como la variabilidad en la estructura de los sitios web y la necesidad de manejar grandes volúmenes de datos en tiempo real. A través de un enfoque iterativo y la implementación de soluciones técnicas innovadoras, se han superado estos obstáculos, logrando un sistema robusto y escalable.

6. Impacto y Futuras Mejoras: El sistema desarrollado no solo facilita el acceso a noticias

relevantes para la comunidad universitaria, sino que también sienta las bases para futuras mejoras. Entre las posibles líneas de trabajo futuro, se incluyen la integración de nuevas fuentes de noticias, la mejora de los algoritmos de clasificación y la implementación de funcionalidades adicionales, como la personalización de noticias según las preferencias del usuario.

En resumen, este proyecto ha logrado cumplir con los objetivos planteados, proporcionando una herramienta valiosa para la Universidad de Jaén. La combinación de técnicas avanzadas de PLN, un diseño de interfaz centrado en el usuario y un enfoque robusto en la recolección y clasificación de noticias ha resultado en un sistema eficiente y efectivo. A medida que se continua explorando nuevas posibilidades y mejoras, este proyecto representa un paso significativo hacia la automatización y optimización de la gestión de información en entornos académicos.

Capítulo 8

Anexos

A continuación se muestra el código de los scripts que se han implementado y que se ha hecho referencia de ellos en otros apartados.

8.1. abstract_scraper.py

8.1.1. Librerías

```
from abc import ABC, abstractmethod
import os
import pandas as pd
```

8.1.2. Método: __init__(self, base_dir)

```
def __init__(self, base_dir):
    self.base_dir = base_dir
    self.titulos = []
    self.enlaces = []
    self.subtitulos = []
    self.fechas = []
    self.autores = []
    self.textosNoticias = []
```

```
        self.fuente = []
```

8.1.3. Método: extraer_titulosYenlaces(self, soup)

```
@abstractmethod
def extraer_titulosYenlaces(self, soup):
    #Lo implementamos en las clases de cada
    periódico
    pass
```

8.1.4. Método: extraer_noticia(self, enlace)

```
@abstractmethod
def extraer_noticia(self, enlace):
    #Lo implementamos en las clases de cada periódico
    pass
```

8.1.5. Método: guardar_csv(self, carpeta, nombre)

```
def guardar_csv(self, carpeta, nombre):
    #Creamos la carpeta si no existe
    if not os.path.exists(carpeta):
        os.makedirs(carpeta)

    #Creamos el dataframe
    destino = os.path.join(carpeta, nombre)
    df = pd.DataFrame({
        'Titulos': self.titulos,
        'Enlaces': self.enlaces,
        'Subtitulos': self.subtitulos,
        'Fechas': self.fechas,
        'Autores': self.autores,
        'Texto': self.textosNoticias,
        'Fuente': self.fuente
    })
    #Lo importamos en la carpeta de destino
    df.to_csv(destino, index=False, encoding='utf-8', sep=';')
```

8.1.6. Método: main(self)

```
def main(self):
    raise NotImplementedError(" Esto se debe
    implementar en las subclases!")
```

8.2. diarioJaen_scraper.py

8.2.1. Librerías

```
import os
import requests
from bs4 import BeautifulSoup
import datetime
from abstract_scraper import AbstractScraper
```

8.2.2. Método: extraer_titulosYenlaces(self, soup)

```
def extraer_titulosYenlaces(self, soup):
    articulos = soup.find_all('h2', class_='titulo')
    titulos_texto = []
    enlaces = []
    sitio_base = "https://www.diariojaen.es"
    for articulo in articulos:
        span = articulo.find('span', class_='priority-content')
        if span is not None:
            texto = span.get_text(strip=True)
            titulos_texto.append(texto)
        else:
            texto = "ERROR"
            titulos_texto.append(texto)
```

```
            span = articulo.find('a')
            if span is not None:
                noticia = span['href']
                enlace = sitio_base + noticia
                enlaces.append(enlace)
            else:
                enlace = "ERROR"
                enlaces.append(texto)
```

8.2.3. Método: extraer_noticia(self, enlace)

```
    def extraer_noticia(self, enlace):
        try:
            resultado = requests.get(enlace)
            resultado.encoding = 'utf-8'

            #Ya que el servidor va regular, vamos a chequear que el
            contenido que devuelve es HTML
            content_type = resultado.headers.get('Content-Type', '')
            if 'text/html' not in content_type:
                print(f"El servidor está haciendo de las suyas: {content_type}")
                return 'ERROR', 'ERROR', 'ERROR', 'ERROR'



            contenido = resultado.text
            soup = BeautifulSoup(contenido, 'html.parser')
            #print(soup)

            #Extraemos subtítulo
            noticia = soup.find('h3', class_= 'subheadline font-1 small bold')
            if noticia is not None:
                subtitulo = noticia.find('span', class_='priority-
                content').get_text(strip=True)
            else:
                subtitulo = "Sin subtitulo"
            print(subtitulo)

            #Extraemos fecha
```

```
        fecha = soup.find('div', class_='datefrom small')
        if fecha is not None:
            fecha = fecha.get_text(strip=True)
        else:
            fecha = "Error con la fecha"
        print(fecha)

        #Extraemos autor
        noticia = soup.find('div', class_='autor')
        if noticia is not None:
            autor = noticia.find('a').get_text(strip=True)
        else:
            autor = "Anonimo"
        print(autor)


        parrafos = soup.find_all('div', class_='paragraph')
        apartados = []
        for parrafo in parrafos:
            noticia = parrafo.find('p')
            if noticia is not None:
                texto = noticia.get_text(strip=True)
                apartados.append(texto)
            else:
                texto = "ERROR"
                apartados.append(texto)


        noticia_completa = "\n\n".join(apartados)
        print(noticia_completa)

        return subtitulo, fecha, autor, noticia_completa

    except requests.exceptions.TooManyRedirects:
        print(f"Demasiadas redirecciones: {enlace}")
        return 'ERROR', 'ERROR', 'ERROR', 'ERROR'

    except requests.exceptions.RequestException as e:
```

```
            print(f"Error al acceder a la página: {e}")
            return 'ERROR', 'ERROR', 'ERROR', 'ERROR'
```

8.2.4. Método: main(self)

```
    def main(self):
        #Esto lo hacemos ya que el enlace se va ajustando al dia actual,
        por lo que hay que actualizarlo al dia de hoy.
        fecha_actual = datetime.datetime.now()
        fecha_Anio = fecha_actual.strftime("%Y%m%d")
        fechaSinAnio = fecha_actual.strftime("%m%d")
        print(fecha_Anio)
        print(fechaSinAnio)
        sitio_base = f'https://www.diariojaen.es/busqueda
        /-/search/"Universidad%20de%20Jaen"
        /false/false/1982{fechaSinAnio}/{fecha_Anio}
        /date/true/true/0/0/meta/0/0/0/{{}}'

        pagina = 1

        while True:
            print(f"Página: {pagina}")
            sitio_web = sitio_base.format(pagina)
            print(sitio_web)
            resultado = requests.get(sitio_web)

            #Configuramos la codificación de los carácteres correcta
            resultado.encoding = 'utf-8'

            #Parseamos el html
            contenido = resultado.text
            soup = BeautifulSoup(contenido, 'html.parser')

            #Extraemos datos y lo metemos en la lista global
            titulos, enlaces = self.extraer_titulosYenlaces(soup)
            self.titulos.extend(titulos)
            self.enlaces.extend(enlaces)
```

```
            for enlace in enlaces:
                print(enlace)
                self.fuente.append("DiarioJaen")
                subtitulo, fecha, autor, noticia =
                self.extraer_noticia(enlace)
                self.subtitulos.append(subtitulo)
                self.fechas.append(fecha)
                self.autores.append(autor)
                self.textosNoticias.append(noticia)

            if not titulos:
                print("No hay más títulos")
                break

            #Incrementamos el número de la página
            pagina += 1

        print(f"Longitudes finales: titulos={len(self.titulos)},
        enlaces={len(self.enlaces)}, subtitulos={len(self.subtitulos)},
        fechas={len(self.fechas)}, autores={len(self.autores)},
        textosNoticias={len(self.textosNoticias)},
        fuente={len(self.fuente)}")
```

8.3. extraerHTML_IdealJaen.py

8.3.1. Librerías

```
import os
import pickle
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By
```

```
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.action_chains import ActionChains
```

8.3.2. Método: cargarEstructurasPickle(elemento, carpeta, nombre)

```
def cargarEstructurasPickle(elemento, carpeta, nombre):
    #Creamos la carpeta en el caso de que no exista
    if not os.path.exists(carpeta):
        os.makedirs(carpeta)

    destino = os.path.join(carpeta, nombre)

    with open(destino, 'wb') as pickle_file:
        pickle.dump(elemento, pickle_file)
```

8.3.3. Método: main()

```
@staticmethod
def main():
    dir_base = os.getcwd()
    IdealJaen = os.path.join(dir_base, "IdealJaen")

    #Inicializamos la ruta donde está el webDriver
    prueba = os.path.join(dir_base, "config", "chromedriver.exe")

    #Configuramos el servicio de Chrome con la ruta
    service = Service(prueba)

    #Inicializamos el controlador de Chrome
    driver = webdriver.Chrome(service=service)

    html = 0

    try:
        #Abrimos la página web, es el ideal con la búsqueda "UJA"
        driver.get('https://www.ideal.es/temas/
```

```
        entidades/universidad-de-jaen.html')

        #Esperamos para asegurar que el botón sea clicable
        botonCook = WebDriverWait(driver, 10).until(
            EC.element_to_be_clickable((By.ID, 'didomi-notice-agree-button'))
        )

        #Hacemos clic en el botón de cookies
        botonCook.click()

        #Hacemos clic en el botón "ver más noticias" múltiples veces,
        hasta que no haya más
        for i in range(0, 1000):
            #time.sleep(2)

            #Esperamos para asegurar que el botón sea clicable
            botonMas = WebDriverWait(driver, 10).until(
                EC.element_to_be_clickable((By.ID, 'btn-crono'))
            )

            #Desplazamos la vista al botón "Más Noticias" usando ActionChains,
            asi evitamos el error de elemento intercepado
            actions = ActionChains(driver)
            actions.move_to_element(botonMas).perform()

            #time.sleep(1)

            #Hacemos click en el botón
            driver.execute_script("arguments[0].click();", botonMas)

    except Exception as e:

        #Cuando se lanze la excepción porque no se encuentra el botón
        "Mas Noticias" es porque no hay más noticias
        #Guardamos el html
        html = driver.page_source
        extraerHTML_IdealJaen.cargarEstructurasPickle(html, IdealJaen,
        "html.pickle")
```

```
        #Imprimimos el error para saber que ha ocurrio
        print(f"Se ha terminado de pulsar el Boton: {e}")

        #Esta funcion para dejar abierto el navegador
        cuando la excepción se lance
        #input("Presionar Enter para cerrar chrome")

    finally:
        #Cerramos si no hubo errores
        driver.quit()
```

8.4. idealJaen_scraper.py

8.4.1. Librerías

```
import os
import pickle
import requests
from bs4 import BeautifulSoup
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from abstract_scraper import AbstractScraper
```

8.4.2. Método: extraer_titulosYenlaces(self, soup)

```
def extraer_titulosYenlaces(self, soup):
    titulos_texto = []
    enlaces = []
    articulos = soup.find_all('h2', class_='v-a-t')
    for articulo in articulos:
```

```
            a_tag = articulo.find('a')
            if a_tag is not None:
                texto = a_tag.get_text(strip=True)
                enlace = a_tag['href']

                enlaces.append(enlace)
                titulos_texto.append(texto)
            else:
                texto = "ERROR"
                enlace = "ERROR"

                titulos_texto.append(texto)
                enlaces.append(enlace)

        return titulos_texto, enlaces
```

8.4.3. Método: extraer_noticia(self, html)

```
    def extraer_noticia(self, html):
        try:
            soup = BeautifulSoup(html, 'html.parser')
            #print(soup)

            #Extraemos subtítulo
            noticia = soup.find('h2', class_='v-a-sub-t')
            if noticia is not None:
                subtitulo = noticia.get_text(strip=True)
            else:
                subtitulo = "Sin subtitulo"
            print(subtitulo)

            #Extraemos fecha
            fecha = soup.find('p', class_='v-mdl-ath__p v-mdl-ath__p--6')
            if fecha is not None:
                fecha = fecha.get_text(strip=True)
            else:
                fecha = "Error con la fecha"
            print(fecha)
```

```
        #Extraemos autor
        noticia = soup.find('div', class_='v-mdl-ath__inf')
        if noticia is not None:
            autor = noticia.find('p', class_='v-mdl-
            ath__p v-mdl-ath__p--2').get_text(strip=True)
        else:
            autor = "Anonimo"
        print(autor)


        #parrafos = soup.find('div', class_='v-d v-d--ab-c v-d--bs')
        apartados = []
        #Coge cosas como telegram, whatsapp etc..
        parrafos = soup.find_all('p', class_='v-p')
        if parrafos is not None:
            for noticia in parrafos:
                if noticia is not None:
                    texto = noticia.get_text(strip=True)
                    if len(texto) >= 20:
                        apartados.append(texto)
        else:
            texto = "ERROR"
            apartados.append(texto)



        noticia_completa = "\n\n".join(apartados)

        print(noticia_completa)


        return subtitulo, fecha, autor, noticia_completa

    except requests.exceptions.RequestException as e:
        print(f"Error al acceder a la página: {e}")
        return 'ERROR', 'ERROR', 'ERROR', 'ERROR'
```

8.4.4. Método: main(self))

```
def main(self):
    try:
        destino = os.path.join(self.base_dir,"IdealJaen", "html.pickle")
        with open(destino, 'rb') as file:
            contenido = pickle.load(file)
        soup = BeautifulSoup(contenido, 'html.parser')

        #Extraemos datos y lo metemos en la lista global
        titulos, enlaces = self.extraer_titulosYenlaces(soup)

        self.titulos.extend(titulos)
        self.enlaces.extend(enlaces)

        #Inicializamos la ruta donde está el webDriver
        prueba = os.path.join(self.base_dir, "config", "chromedriver.exe")

        #Configuramos el servicio de Chrome con la ruta
        service = Service(prueba)

        driver = webdriver.Chrome(service=service)

        for enlace in enlaces:
            print(enlace)
            self.fuente.append("IdealJaen")
            driver.get(enlace)
            html = driver.page_source
            subtitulo, fecha, autor, noticia = self.extraer_noticia(html)
            self.subtitulos.append(subtitulo)
            self.fechas.append(fecha)
            self.autores.append(autor)
            self.textosNoticias.append(noticia)


    except Exception as e:
        print(f"Ocurrió un error: {e}")
```

```
        finally:

            #Guardamos todo lo recopilado en formato .csv
            self.guardar_csv(os.path.join(self.base_dir, "IdealJaen"), "IdealJaen.csv"

            #Quitamos chrome para que se pueda ejecutar el siguiente scraper
            driver.quit()
```

8.5. horaJaen_scraper.py

8.5.1. Librerías

```
import os
import requests
from bs4 import BeautifulSoup
from abstract_scraper import AbstractScraper
```

8.5.2. Método: extraer_titulosYenlaces(self, soup)

```
def extraer_titulosYenlaces(self, soup):
    titulos_texto = []
    enlaces = []

    noticias = soup.find('div', class_='td-ss-main-content')

    #Se verifica que el contenedor fue encontrado
    if not noticias:
        print("No se han encontrado más títulos de noticias")
        return titulos_texto, enlaces  #Salir de la
        función si no se encuentra el contenedor

    articulos = noticias.find_all('h3', class_='entry-title td-module-title')
```

```
        for articulo in articulos:
            a_tag = articulo.find('a')
            if a_tag is not None:
                texto = a_tag.get_text(strip=True)
                enlace = a_tag['href']
                enlaces.append(enlace)
                titulos_texto.append(texto)
            else:
                texto = "ERROR"
                enlace = "ERROR"
                titulos_texto.append(texto)
                enlaces.append(enlace)
```

8.5.3. Método: extraer_noticia(self, enlace)

```
    def extraer_noticia(self, enlace):
        try:
            resultado = requests.get(enlace)
            resultado.encoding = 'utf-8'

            #Ya que el servidor va regular, vamos a chequear
            que el contenido que devuelve es HTML
            content_type = resultado.headers.get('Content-Type', '')
            if 'text/html' not in content_type:
                print(f"El servidor está haciendo de las suyas: {content_type}")
                return 'ERROR', 'ERROR', 'ERROR', 'ERROR'


            contenido = resultado.text
            soup = BeautifulSoup(contenido, 'html.parser')


            #Extraemos subtítulo
            subtitulo = "Este periódico no tiene subtítulo"
            print(subtitulo)

            #Extraemos fecha
            fecha = soup.find('time', class_='entry-date updated td-module-date')
```

```
        if fecha is not None:
            fecha = fecha.get_text(strip=True)
        else:
            fecha = "Error con la fecha"
        print(fecha)

        #Extraemos autor
        autor = "Anonimo"
        print(autor)


        parrafos = soup.find('div', class_='td-post-content')
        apartados = []
        parrafos = parrafos.find_all('p')
        for noticia in parrafos:
            #noticia = parrafo.find('p')
            if noticia is not None:
                texto = noticia.get_text(strip=True)
                apartados.append(texto)
            else:
                texto = "ERROR"
                apartados.append(texto)


        noticia_completa = "\n\n".join(apartados)
        print(noticia_completa)

        return subtitulo, fecha, autor, noticia_completa

    except requests.exceptions.TooManyRedirects:
        print(f"Demasiadas redirecciones: {enlace}")
        return 'ERROR', 'ERROR', 'ERROR', 'ERROR'

    except requests.exceptions.RequestException as e:
        print(f"Error al acceder a la página: {e}")
        return 'ERROR', 'ERROR', 'ERROR', 'ERROR'
```

8.5.4. Método: main(self)

```
def main(self):
    sitio_base = 'https://www.horajaen.com/page/{}/?s="universidad+de+Jaén"'
    sitio_base2 = 'https://www.horajaen.com/page/{}/?s="+UJA+"'

    pagina = 1
    while True:
        print(f"Página: {pagina}")
        print("Universidad de Jaén")
        sitio_web = sitio_base.format(pagina)
        resultado = requests.get(sitio_web)

        #Configuramos la codificación de los carácteres correcta
        resultado.encoding = 'utf-8'

        #Parseamos el html
        contenido = resultado.text
        soup = BeautifulSoup(contenido, 'html.parser')

        #Extraemos datos y lo metemos en la lista global
        titulos, enlaces = self.extraer_titulosYenlaces(soup)
        self.titulos.extend(titulos)
        self.enlaces.extend(enlaces)

        for enlace in enlaces:
            print(enlace)
            self.fuente.append("HoraJaen")
            subtitulo, fecha, autor, noticia = self.extraer_noticia(enlace)
            self.subtitulos.append(subtitulo)
            self.fechas.append(fecha)
            self.autores.append(autor)
            self.textosNoticias.append(noticia)

        if not titulos:
            print("No hay más títulos")
            break
```

```
        #Incrementamos el número de la página
        pagina += 1

    pagina = 1
    while True:
        print(f"Página: {pagina}")
        print("UJA")
        sitio_web = sitio_base2.format(pagina)
        resultado = requests.get(sitio_web)

        #Configuramos la codificación de los carácteres correcta
        resultado.encoding = 'utf-8'

        #Parseamos el html
        contenido = resultado.text
        soup = BeautifulSoup(contenido, 'html.parser')

        #Extraemos datos y lo metemos en la lista global
        titulos, enlaces = self.extraer_titulosYenlaces(soup)
        self.titulos.extend(titulos)
        self.enlaces.extend(enlaces)

        for enlace in enlaces:
            print(enlace)
            self.fuente.append("HoraJaen")
            subtitulo, fecha, autor, noticia = self.extraer_noticia(enlace)
            self.subtitulos.append(subtitulo)
            self.fechas.append(fecha)
            self.autores.append(autor)
            self.textosNoticias.append(noticia)

        if not titulos:
            print("No hay más títulos")
            break

        #Incrementamos el número de la página
        pagina += 1
```

```
        self.guardar_csv(os.path.join(self.base_dir, "HoraJaen"), "HoraJaen.csv")
```

8.6. combinar_csv

```
def combinar_csv(self):
    dir_base = os.getcwd()
    ruta_Diario = os.path.join(dir_base, "DiarioJaen", "DiarioJaen.csv")
    ruta_Ideal = os.path.join(dir_base, "IdealJaen", "IdealJaen.csv")
    ruta_Hora = os.path.join(dir_base, "HoraJaen", "HoraJaen.csv")

    noticias = []

    Diario = pd.read_csv(ruta_Diario, delimiter=';', encoding='utf-8')
    Ideal = pd.read_csv(ruta_Ideal, delimiter=';', encoding='utf-8')
    Hora = pd.read_csv(ruta_Hora, delimiter=';', encoding='utf-8')

    noticias.append(Diario)
    noticias.append(Ideal)
    noticias.append(Hora)

    #Se concatena todo en un solo DataFrame
    self.df_procesado = pd.concat(noticias, ignore_index=True)
```

8.7. filtrar_noticias_anomalas

```
def filtrar_noticias_anomalas(self):
    df = self.df_procesado

    #Filtrar las noticias de "HoraJaen"
    df_horajaen = df[df['Fuente'] == 'HoraJaen']

    #Eliminar duplicados dentro de "HoraJaen"
    df_horajaen_sin_duplicados = df_horajaen.drop_duplicates(subset=['Titulos'
    , "Texto"], keep='first')

    #Filtrar el resto de las noticias
    df_otros = df[df['Fuente'] != 'HoraJaen']

    #Combinar las noticias sin duplicados de "HoraJaen"
    con el resto del conjunto de datos
    df_sin_duplicados = pd.concat([df_horajaen_sin_duplicados, df_otros])

    #Definir función para identificar noticias anómalas
    def es_anomala(fila):
        for valor in fila:
            if pd.isna(valor) or (isinstance(valor, str) and 'ERROR' in valor):
                return True

        if len(fila['Texto']) < 100:
            return True

        palabras = fila['Texto'].split()
        num_palabras = len(palabras)
        if num_palabras > 0:
            palabras_unicas = set(palabras)
            if len(palabras_unicas) / num_palabras < 0.2:
                return True

        return False
```

```
    #Aplicar la función es_anomala a cada fila del DataFrame sin duplicados
    df_sin_duplicados['Anomala'] = df_sin_duplicados.apply(es_anomala, axis=1)

    #Filtrar las noticias anómalas
    df_anomalas = df_sin_duplicados[df_sin_duplicados['Anomala']]
    self.df_procesado = df_sin_duplicados[~df_sin_duplicados['Anomala']]
```

8.8. convertir_fechas

```
def convertir_fechas(self):
    df = self.df_procesado

    # Diccionario de meses
    meses = {
        'enero': '01', 'febrero': '02', 'marzo': '03', 'abril': '04', 'mayo': '05',
        'junio': '06', 'julio': '07', 'agosto': '08', 'septiembre': '09',
        'octubre': '10', 'noviembre': '11', 'diciembre': '12',
        'ene': '01', 'feb': '02', 'mar': '03', 'abr': '04', 'may': '05',
        'jun': '06', 'jul': '07', 'ago': '08', 'sep': '09', 'oct': '10',
        'nov': '11', 'dic': '12'
    }

    def convertir_fecha_general(fecha_str):
        try:
            if ' / ' in fecha_str:
                fecha_partes, hora_partes = fecha_str.split(' / ')
                dia, mes, año = fecha_partes.split()
                mes_num = meses[mes.lower()]
                fecha_formateada = f"{int(dia):02d}/{mes_num}/{str(año)}"
                return datetime.strptime(fecha_formateada, '%d/%m/%Y')
```

```
            elif '/' in fecha_str and ',' in fecha_str and
            ':' in fecha_str and "| " in fecha_str:
                antiguo, nuevo = fecha_str.split("| ")
                actualizado, fecha_formateada, hora = nuevo.split(" ")
                return datetime.strptime(fecha_formateada, '%d/%m/%Y')
            elif ',' in fecha_str and ':' in fecha_str and "| " in fecha_str:
                fecha, actualizado = fecha_str.split('|')
                dia_semana, resto = fecha.split(', ', 1)
                dia, mes_año_hora = resto.split(' de ')
                mes_año, hora = mes_año_hora.split(', ')
                mes, año = mes_año.split(' ')
                mes_num = meses[mes.lower()]
                fecha_formateada = f"{int(dia):02d}/{mes_num}/{str(año)}"
                return datetime.strptime(fecha_formateada, '%d/%m/%Y')
            elif ',' in fecha_str and ':' in fecha_str:
                dia_semana, resto = fecha_str.split(', ', 1)
                dia, mes_año_hora = resto.split(' de ')
                mes_año, hora = mes_año_hora.split(', ')
                mes, año = mes_año.split(' ')
                mes_num = meses[mes.lower()]
                fecha_formateada = f"{int(dia):02d}/{mes_num}/{str(año)}"
                return datetime.strptime(fecha_formateada, '%d/%m/%Y')
            else:
                if ',' in fecha_str:
                    fecha_partes, año = fecha_str.split(', ')
                    dia, mes = fecha_partes.split()
                    mes_num = meses[mes.lower()]
                    fecha_formateada = f"{int(dia):02d}/{mes_num}/{str(año)}"
                    return datetime.strptime(fecha_formateada, '%d/%m/%Y')
                else:
                    raise ValueError("Formato de fecha no reconocido")
        except Exception as e:
            print(f"Error al convertir la fecha '{fecha_str}': {e}")
            return None

    df['Fechas'] = df['Fechas'].apply(convertir_fecha_general)
    df['Fechas'] = pd.to_datetime(df['Fechas'], errors='coerce')
```

```
    self.df_procesado=df
```

8.9. procesar_texto

```
def procesar_texto(self):
        df = self.df_procesado

        nltk.download('punkt')
        nltk.download('stopwords')

        #Stopwords
        stop_words = set(stopwords.words('spanish'))
        domain_specific_stopwords = {'jaén', 'universidad', 'uja',
        'persona', 'si', 'hace', 'ser'}
        stop_words = stop_words.union(domain_specific_stopwords)

        #Para eliminar puntuación
        translator = str.maketrans('', '', string.punctuation)

        def preprocess(text):
            tokens = word_tokenize(text.lower().translate(translator))
            tokens = [word for word in tokens if word.isalpha() and word
            not in stop_words]
            lemmatizer = WordNetLemmatizer()
            tokens = [lemmatizer.lemmatize(word) for word in tokens]
            return tokens

        #Se crea una nueva columna con los textos procesados
        df['processed_text'] = df['Texto'].apply(preprocess)

        self.df_procesado = df
```

8.10. Clasificación.py

```
import pandas as pd
import nltk
import ast
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from gensim import corpora
from gensim.models.ldamodel import LdaModel
from gensim.models.coherencemodel import CoherenceModel
import string

class Clasificacion:
    def __init__(self, csv_input_path, csv_output_path,
    num_topics=9, passes=20, alpha='auto', eta='auto'):
        self.csv_input_path = csv_input_path
        self.csv_output_path = csv_output_path
        self.num_topics = num_topics
        self.passes = passes
        self.alpha = alpha
        self.eta = eta
        self.df = None
        self.dictionary = None
        self.corpus = None
        self.lda_model = None

    def cargar_datos(self):
        # Cargar el archivo CSV
        self.df = pd.read_csv(self.csv_input_path, encoding='utf-8', delimiter=';')
        print(f"Datos cargados desde {self.csv_input_path}")

        # Convertir las cadenas de texto de 'processed_text' en listas
        self.df['processed_text'] = self.df['processed_text'].
        apply(ast.literal_eval)
        print("Datos convertidos a listas.")
```

```
    def crear_diccionario_y_corpus(self):
        # Crear diccionario y corpus

        print(self.df['processed_text'].head())

        self.dictionary = corpora.Dictionary(self.df['processed_text'])
        self.corpus = [self.dictionary.doc2bow(text) for text
        in self.df['processed_text']]
        print("Diccionario y corpus creados.")

    def aplicar_lda(self):
        # Aplicar LDA
        self.lda_model = LdaModel(
            self.corpus,
            num_topics=self.num_topics,
            id2word=self.dictionary,
            passes=self.passes,
            alpha=self.alpha,
            eta=self.eta
        )
        print("Modelo LDA aplicado.")

    def mostrar_topicos(self):
        # Mostrar los tópicos descubiertos
        topics = self.lda_model.print_topics(num_words=5)
        for topic in topics:
            print(topic)

    def asignar_categorias(self):
        # Asignar categorías basadas en los tópicos descubiertos
        def get_category(text):
            bow = self.dictionary.doc2bow(text)
            topic_distribution = self.lda_model.get_document_topics(bow)
            dominant_topic = max(topic_distribution, key=lambda x: x[1])[0]
            return dominant_topic

        self.df['Categoría'] = self.df['processed_text'].apply(get_category)
        print("Categorías asignadas.")
```

```
    def guardar_csv(self):
        # Guardar el CSV con la nueva columna
        self.df.to_csv(self.csv_output_path, index=False, encoding='utf-8', sep=';')
        print(f"Datos guardados en {self.csv_output_path}")

    def evaluar_modelo(self):
        """----------------Evaluación----------------"""
        # Calcular coherencia de tópicos
        coherence_model_lda = CoherenceModel(model=self.lda_model,
        texts=self.df['processed_text'], dictionary=self.dictionary
        , coherence='c_v')
        coherence_lda = coherence_model_lda.get_coherence()
        print(f'Coherencia de tópicos: {coherence_lda}')

        # Calcular perplejidad
        perplexity = self.lda_model.log_perplexity(self.corpus)
        print(f'Perplejidad: {perplexity}')

    def main(self):
        # Ejecutar todos los pasos
        self.cargar_datos()
        self.crear_diccionario_y_corpus()
        self.aplicar_lda()
        self.mostrar_topicos()
        self.asignar_categorias()
        self.guardar_csv()
        self.evaluar_modelo()

if __name__ == '__main__':
    # Uso de la clase
    clasificacion = Clasificacion('NoticiasProcesadas.csv',
    'NoticiasClasificadas.csv')
    clasificacion.main()
```

8.11. ConectarBBDD.py

```
import pandas as pd
from pymongo import MongoClient
from datetime import datetime

class Almacenamiento:
    def __init__(self, csv_file_path, mongo_uri='mongodb://localhost:27017/',
    db_name='Sistema', collection_name='noticias'):
        self.csv_file_path = csv_file_path
        self.mongo_uri = mongo_uri
        self.db_name = db_name
        self.collection_name = collection_name
        self.df = None

    def leer_csv(self, sep=';'):
        # Leer el archivo CSV
        self.df = pd.read_csv(self.csv_file_path, sep=sep)
        print(f"CSV leído correctamente desde {self.csv_file_path}")

    def convertir_fechas(self):
        # Convertir las fechas en el DataFrame
        for i, row in self.df.iterrows():
            fecha_str = row['Fechas']
            try:
                # Convertir la cadena a un objeto datetime
                fecha_datetime = datetime.strptime(str(fecha_str), '%Y-%m-%d')
                # Actualizar la fecha en el DataFrame
                self.df.at[i, 'Fechas'] = fecha_datetime
            except ValueError:
                print(f"Error al convertir la fecha en la fila {i}: {fecha_str}")
        print("Fechas convertidas correctamente.")

    def insertar_en_mongodb(self):
        # Conectar a MongoDB
        client = MongoClient(self.mongo_uri)
        db = client[self.db_name]
        collection = db[self.collection_name]
```

```
        # Convertir el DataFrame a diccionarios y insertar en MongoDB
        data_dict = self.df.to_dict(orient='records')
        collection.insert_many(data_dict)
        print("Datos insertados en MongoDB correctamente.")

    def main(self):
        # Ejecutar todos los pasos
        self.leer_csv()
        self.convertir_fechas()
        self.insertar_en_mongodb()

# Uso de la clase
almacenamiento = Almacenamiento('NoticiasClasificadas.csv')
almacenamiento.main()
```

8.12. script.js

```
document.addEventListener('DOMContentLoaded', () => {
    if (window.location.pathname.endsWith('noticia.html')) {
        cargarNoticia();

        // Añadir event listener al botón de "Volver a los resultados"
        const volverBtn = document.getElementById('volverBtn');
        volverBtn.addEventListener('click', () => {
            // Restaurar la página de filtros y resultados aplicados previamente
            const filters = JSON.parse(localStorage.getItem('filters'));
            if (filters) {
```

```
                const params = new URLSearchParams(filters).toString();
                window.location.href = `index.html?${params}`;
            } else {
                window.location.href = 'index.html';
            }
        });
    } else {
        cargarFiltrosGuardados();
        window.addEventListener('popstate', cargarFiltrosGuardados);
    }
});

// Función para filtrar noticias
function filtrarNoticias() {
    const fechaInicio = document.getElementById('fechaInicio').value;
    const fechaFin = document.getElementById('fechaFin').value;
    const categoria = document.getElementById('categoria').value;
    const autor = document.getElementById('autor').value;
    const fuente = document.getElementById('fuente').value;

    const params = new URLSearchParams();
    if (fechaInicio) params.append('fechaInicio', fechaInicio);
    if (fechaFin) params.append('fechaFin', fechaFin);
    if (categoria) params.append('categoria', categoria);
    if (autor) params.append('autor', autor);
    if (fuente) params.append('fuente', fuente);

    // Guardar filtros en localStorage
    const filters = {
        fechaInicio, fechaFin, categoria, autor, fuente
    };
    localStorage.setItem('filters', JSON.stringify(filters));

    // Guardar estado en el historial del navegador
    history.pushState(filters, '', `?${params.toString()}`);

    // Consultar la API con o sin parámetros
    const url = params.toString() ? `http://localhost:3000/api/noticias?
```

```
${params.toString()}‘ : ’http://localhost:3000/api/noticias’;

fetch(url)
    .then(response => response.json())
    .then(data => {
        const resultsDiv = document.getElementById(’results’);
        resultsDiv.innerHTML = ’’;

        // Ordenar las noticias por fecha de más nuevas a más antiguas
        data.sort((a, b) => new Date(b.Fechas) - new Date(a.Fechas));

        // Mostrar el número de noticias recuperadas
        const countMessage = document.createElement(’p’);
        countMessage.className = ’results-count’;
        if (data.length > 0) {
            countMessage.innerText = ‘Se han recuperado
            ${data.length} noticias.‘;
        } else {
            countMessage.innerText = ’No se encontraron noticias con
            los filtros seleccionados.’;
        }
        resultsDiv.appendChild(countMessage);

        if (data.length > 0) {
            data.forEach(noticia => {
                const newsItem = document.createElement(’div’);
                newsItem.className = ’noticia’;

                let subtituloHTML = ’’;
                if (noticia.Subtitulos && noticia.Subtitulos !== "Sin subtitulo")
                    subtituloHTML = ‘<p>${noticia.Subtitulos}.</p>‘;
                }

                newsItem.innerHTML = ‘
                    <h3>${noticia.Titulos}</h3>
                    <p><strong>Fuente:</strong> ${noticia.Fuente}</p>
                    <p><strong>Fecha:</strong> ${new Date(noticia.Fechas).
                    toLocaleDateString()}</p>
```

```
                        <p><strong>Categoría:</strong> ${noticia.Categoría}</p>
                        ${subtituloHTML}
                    `;

                    newsItem.onclick = () => {
                        window.location.href = `noticia.html?id=${noticia._id}`;
                    };

                    resultsDiv.appendChild(newsItem);
                });
            }
        })
        .catch(error => {
            console.error('Error al filtrar noticias:', error);
        });
}


// Función para cargar filtros guardados y resultados en el caso de que haya
function cargarFiltrosGuardados() {
    const savedFilters = JSON.parse(localStorage.getItem('filters'));
    if (savedFilters) {
        document.getElementById('fechaInicio').value = savedFilters.fechaInicio;
        document.getElementById('fechaFin').value = savedFilters.fechaFin;
        document.getElementById('categoria').value = savedFilters.categoria;
        document.getElementById('autor').value = savedFilters.autor;
        document.getElementById('fuente').value = savedFilters.fuente;

        // Simular el filtro cuando la página carga
        filtrarNoticias();
    } else {
        // Si no hay filtros guardados, simplemente mostrar todas las noticias
        fetch('http://localhost:3000/api/noticias')
            .then(response => response.json())
            .then(data => {
                const resultsDiv = document.getElementById('results');
                resultsDiv.innerHTML = '';
```

```
                if (data.length > 0) {
                    data.forEach(noticia => {
                        const newsItem = document.createElement('div');
                        newsItem.className = 'noticia';

                        let subtituloHTML = '';
                        if (noticia.Subtitulos && noticia.Subtitulos
                        !== "Sin subtitulo") {
                            subtituloHTML = `<p>${noticia.Subtitulos}</p>`;
                        }

                        newsItem.innerHTML = `
                            <h3>${noticia.Titulos}</h3>
                            <p><strong>Fuente:</strong> ${noticia.Fuente}</p>
                            <p><strong>Fecha:</strong> ${new Date(noticia.Fechas).
                            toLocaleDateString()}</p>
                            <p><strong>Categoría:</strong> ${noticia.Categoría}</p>
                            ${subtituloHTML}
                        `;

                        newsItem.onclick = () => {
                            window.location.href = `noticia.html?id=${noticia._id}`;
                        };

                        resultsDiv.appendChild(newsItem);
                    });
                } else {
                    resultsDiv.innerHTML = '<p>No hay noticias disponibles.</p>';
                }
            })
            .catch(error => {
                console.error('Error al cargar todas las noticias:', error);
            });
    }
}

// Función para cargar la noticia específica
function cargarNoticia() {
```

```
    const id = getParameterByName('id');
    if (!id) {
        alert('No se ha proporcionado un ID de noticia válido.');
        return;
    }

    fetch(`http://localhost:3000/api/noticias/${id}`)
        .then(response => response.json())
        .then(noticia => {
            if (noticia) {
                document.getElementById('titulo').innerText = noticia.Titulos;

                // Verificar el subtítulo
                const subtitulo = noticia.Subtitulos;
                if (subtitulo && subtitulo !== "Sin subtitulo") {
                    document.getElementById('subtitulo').innerText = subtitulo;
                } else {
                    document.getElementById('subtitulo').style.display = 'none';
                }

                document.getElementById('autor').innerText =
                noticia.Autores;
                document.getElementById('fecha').innerText =
                new Date(noticia.Fechas).toLocaleDateString();
                document.getElementById('categoria').innerText =
                noticia.Categoría;
                document.getElementById('fuente').innerText =
                noticia.Fuente;
                document.getElementById('texto').innerHTML =
                noticia.Texto.replace(/\n/g, '<br>');
            } else {
                alert('No se encontró la noticia.');
            }
        })
        .catch(error => {
            console.error('Error al cargar la noticia:', error);
        });
}
```

```
// Función para obtener parámetros de la URL
function getParameterByName(name) {
    const url = window.location.href;
    name = name.replace(/[\[\]]/g, '\\$&');
    const regex = new RegExp('[?&]' + name + '(=([^&#]*)|&|#|$)');
    const results = regex.exec(url);
    if (!results) return null;
    if (!results[2]) return '';
    return decodeURIComponent(results[2].replace(/\+/g, ' '));
}
```

8.13. server.js

```
const express = require('express');
const { MongoClient } = require('mongodb');
const cors = require('cors');

const app = express();
app.use(cors());

const uri = 'mongodb://localhost:27017';
const client = new MongoClient(uri, { useNewUrlParser: true,
useUnifiedTopology: true });

let db;

async function connectToDatabase() {
    try {
        await client.connect();
```

```
        db = client.db('Sistema');
        console.log('Conectado a la base de datos');
    } catch (error) {
        console.error('Error conectando a la base de datos:', error);
    }
}

connectToDatabase();

app.get('/api/noticias', async (req, res) => {
    const { fechaInicio, fechaFin, categoria, autor, fuente } = req.query;

    const filtro = {};

    if (fechaInicio) {
        const fechaInicioDate = new Date(fechaInicio);
        let fechaFinDate = new Date(fechaFin || fechaInicio);


        fechaFinDate.setDate(fechaFinDate.getDate() + 1);

        filtro.Fechas = {
            $gte: fechaInicioDate,
            $lt: fechaFinDate
        };
    }

    if (categoria) filtro.Categoría = categoria;
    if (autor) filtro.Autores = new RegExp(autor, 'i');
    if (fuente) filtro.Fuente = new RegExp(fuente, 'i');

    try {
        if (!db) {
            return res.status(500).json({ error: 'No conectado a la base de datos' })
        }
        const noticias = await db.collection('noticias').find(filtro).toArray();
        res.json(noticias);
    } catch (error) {
```

```
        console.error('Error al consultar las noticias:', error);
        res.status(500).json({ error: 'Ocurrió un error al filtrar las noticias' });
    }
});

const port = 3000;
app.listen(port, () => {
    console.log(`Servidor escuchando en http://localhost:${port}`);
});


const { ObjectId } = require('mongodb');

app.get('/api/noticias/:id', async (req, res) => {
    const { id } = req.params;

    try {
        if (!db) {
            return res.status(500).json({ error: 'No conectado a la base de datos' })
        }

        const noticia = await db.collection('noticias').findOne
        ({ _id: new ObjectId(id) });

        if (!noticia) {
            return res.status(404).json({ error: 'Noticia no encontrada' });
        }

        res.json(noticia);
    } catch (error) {
        console.error('Error al consultar la noticia:', error);
        res.status(500).json({ error: 'Ocurrió un error al consultar la noticia' });
    }
});
```

8.14. Pipeline.py

```
from Preprocesamiento import ProcesadorNoticias
from Clasificación import Clasificacion
from ConectarBBDD import Almacenamiento
from Periodicos import Periodicos

class Pipeline:
    def __init__(self, csv_input_path, csv_output_path):
        self.csv_input_path = csv_input_path
        self.csv_output_path = csv_output_path

    def ejecutar_scraper(self):
        print("Iniciando scraping de periódicos...")
        Periodicos.run_main()

    def ejecutar_preprocesamiento(self):
        print("Iniciando preprocesamiento de datos...")
        procesador = ProcesadorNoticias()
        procesador.main()

    def ejecutar_clasificacion(self):
        print("Iniciando clasificación de tópicos...")
        clasificacion = Clasificacion()
        clasificacion.main(self.csv_input_path, self.csv_output_path)

    def conectar_bd(self):
        print("Conectando a la base de datos y almacenando datos...")
        almacenamiento = Almacenamiento()
        almacenamiento.main(self.csv_output_path)

    def ejecutar_pipeline(self):
        self.ejecutar_scraper()
        self.ejecutar_preprocesamiento()
        self.ejecutar_clasificacion()
        self.conectar_bd()
```

```
if __name__ == '__main__':
    # Definir rutas de los CSV
    csv_input_path = 'NoticiasProcesadas.csv'
    csv_output_path = 'NoticiasClasificadas.csv'

    pipeline = Pipeline(csv_input_path, csv_output_path)
    pipeline.ejecutar_pipeline()
```

Capítulo 9

Bibliografía

[1] Analytics Vidhya. Topic modeling and latent dirichlet allocation (lda) using gensim and scikit-learn. https://www.analyticsvidhya.com/blog/2021/06/part-3-topic-modeling-and-latent-dirichlet-allocation-lda-using-gensim-and-sklear 2021. Blog post.

[2] Astera Software. Sql vs nosql: Understanding the difference. https://www.astera.com/es/knowledge-center/sql-vs-nosql/. Página web.

[3] Canva. Create beautiful designs. https://www.canva.com/. Página web.

[4] Cognizant. Natural language processing (nlp). https://www.cognizant.com/es/es/glossary/natural-language-processing. Página web.

[5] Diario Jaén. https://www.diariojaen.es/. Página web.

[6] Google Chrome Developers. Chromedriver - webdriver for chrome. https://developer.chrome.com/docs/chromedriver/downloads?hl=es-419. Página web.

[7] IBM. Natural language processing (nlp). https://www.ibm.com/es-es/topics/natural-language-processing. Página web.

[8] Ideal. https://www.ideal.es/. Página web.

[9] LinkedIn. Aprovechando al máximo mongodb: Una guía completa. https://www.linkedin.com/pulse/aprovechando-al-m%C3%A1ximo-mongodb-una-gu%C3%ADa-completa-de-juan-francisco-mwfsf/?originalSubdomain=es. Artículo en LinkedIn.

[10] MongoDB. Mongodb database tools. https://www.mongodb.com/try/download/database-tools. Página web.

[11] Pandas. Python data analysis library (pandas). https://pandas.pydata.org/. Página web.

[12] Radim Rehurek. Gensim: Topic modeling for humans. https://radimrehurek.com/gensim/models/ldamodel.html. Página web.

[13] Selenium. Selenium webdriver. https://www.selenium.dev/. Página web.

[14] Hora Jaén. https://www.horajaen.com/. Página web.

[15] Wikipedia. Latent dirichlet allocation. https://es.wikipedia.org/wiki/Latent_Dirichlet_Allocation, 2023. Entrada en Wikipedia.

[16] Creately. Tutorial del diagrama de secuencia. https://creately.com/blog/es/diagramas/tutorial-del-diagrama-de-secuencia/. Página web.

[17] Ibiblio. El diagrama de clase en uml. https://www.ibiblio.org/pub/linux/docs/LuCaS/Tutoriales/doc-modelado-sistemas-UML/multiple-html/x219.html. Tutorial en línea.

[18] YouTube. Cómo realizar web scraping con selenium. https://www.youtube.com/watch?v=gyai5HaqPcI, 2021. Video.

[19] YouTube. Introducción a lda: Latent dirichlet allocation. https://www.youtube.com/watch?v=LV0GRk-Nt6w, 2021. Video.

[20] YouTube. Natural language processing in 5 minutes. https://www.youtube.com/watch?v=S0MW45jXjVY, 2021. Video.

[21] José R. Zapata. Python para la ciencia de datos: Uso de pandas. https://joserzapata.github.io/courses/python-ciencia-datos/pandas/. Curso en línea.

[22] Universidad de Jaén. Manual de identidad corporativa universidad de jaén. https://www.ujaen.es/gobierno/viccom/sites/gobierno_viccom/files/uploads/node_book/2023-04/Manual%20de%20Identidad%20Corporativa%20Universidad%20de%20Ja%C3%A9n%20%2827032023%29%20version_resumida.pdf, 2023. Documento PDF.