UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Adquisición de documentos para procesamiento del lenguaje natural

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Javier García Rojas

Tutor: Lluís Felip Hurtado Oliver

Ferran Pla Santamaría

2019

Resumen

Los medios de comunicación son una gran fuente de información. Gracias al creciente uso de las redes, y con un correcto tratamiento del contenido generado por los usuarios, es posible hacer uso de este contenido en el procesamiento del lenguaje natural. Se ha propuesto realizar como objetivo principal, una herramienta que sea capaz de adquirir contenido de periódicos digitales y una aplicación web capaz de procesar y analizar los datos adquiridos. Para satisfacer el proyecto, ha sido necesario un análisis de requisitos y de datos, un diseño de la solución con diagramas, un estudio de las tecnologías vigentes y un desarrollo de la solución empleando tecnologías como Django, MongoDB y BeautifulSoup4. Tras la medición de tiempos y las pruebas pertinentes, las herramientas permiten extraer, procesar y analizar el contenido con los parámetros definidos por el usuario de una forma adecuada.

Palabras clave: lenguaje natural, adquisición, periódicos, web, análisis.

Resum

Els mitjans de comunicació són una gran font d'informació. Amb el creixent ús de les xarxes, i amb un correcte tractament del contingut generat pels usuaris, és possible fer ús d'aquest contingut en el processament del llenguatge natural. Com a objectiu principal, s'ha proposat el desenvolupament d'una ferramenta que siga capaç d'adquirir continguts de periòdics digitals i una aplicació web capaç de processar i analitzar les dades adquirides. Per a satisfer el projecte, ha sigut necessari una anàlisi de requisits i de dades, un disseny de la solució amb diagrames, un estudi de les tecnologies vigents i un desenvolupament de la solució emprant tecnologies com Django, MongoDB i BeautifulSoup4. Després de mesurar els temps i fer les proves pertinents, les ferramentes permeten extraure, processar i analitzar el contingut amb els paràmetres definits per l'usuari d'una forma adequada.

Paraules clau: llenguatge natural, adquisició, diaris, web, anàlisi.

Abstract

The mass media are a great source of information. With the ever-increasing use of networks, and with a proper treatment of user-generated content, it is possible to make use of this content in natural language processing. It has been suggested as a main objective, a tool capable of acquiring digital newspaper content and a web application capable of processing and analyzing the acquired data. To satisfy the project, it has been necessary an analysis of requirements and data, a design of the solution with diagrams, a study of current technologies and a development of the solution, using technologies such as Django, MongoDB and BeautifulSoup4. After having measured the times and carried out the relevant tests, the tools allow the content to be extracted, processed and analysed with the parameters defined by the user in an appropriate way.

Keywords: natural language, acquisition, newspapers, web, analysis.

Tabla de contenidos

Índice de tablas

Índice de figuras

1. Introducción

Las nuevas tecnologías han hecho que en los últimos años la sociedad utilice nuevas herramientas para publicar información de forma sencilla. Esto ha desembocado en una creación masiva de contenido en internet, ya sea un simple comentario en una red social, un artículo en un foro o una noticia en un periódico digital. Esta información es muy valiosa y se puede interpretar y analizar más allá de lo que la propia información transmite.

El uso que puede dar un usuario a estos contenidos es limitado, pero desde un punto de vista tecnológico es una gran fuente de información. Esta nos lleva al concepto BigData, donde se extraen grandes cantidades de información para almacenarla y posteriormente hacer un análisis de datos mediante diferentes técnicas para interpretar y analizar esta información.

1.1. Motivación

Es mucho el contenido que encontramos en los medios digitales, y la forma de acceder a ellos está al alcance de cualquier usuario. Sin embargo, el enfoque a nivel usuario que se le da a este contenido es limitado. Aquí es cuando entra en juego una visión más tecnológica, haciendo uso del tratamiento de datos a través del cual podemos analizar este contenido y extraer una información valiosa.

El análisis de datos nos proporciona una visión de los datos más enriquecido, esto es, podemos saber cómo evoluciona una noticia, un personaje, el lenguaje empleado por un periódico a lo largo de los años.

Por todo ello, en este proyecto se realiza a pequeña escala una adquisición de datos directamente de portales web enfocados a la prensa digital que posteriormente serán almacenado y tratados con varios algoritmos para ser representados de forma gráfica al usuario.

Dada la cantidad de aplicaciones que se pueden beneficiar de la información obtenida, en el punto **8** se mostrarán diversos ejemplos de uso real.

1.2. Objetivos

El objetivo principal de este proyecto es el desarrollo de una herramienta software que obtenga la mayor cantidad de información posible publicada en diversos sitios webs de prensa digital y la visualización de estos datos para inferir más información útil.

Para poder hacer un buen desarrollo se han fijado los siguientes objetivos:

- Obtención de datos de los siguientes periódicos digitales:
    - El País.
    - El Levante-El Mercantil Valenciano.
    - Daily Mail.
- Almacenamiento de todos los datos obtenidos en una base de datos.
- Limpieza y tratamientos de los datos almacenados.
- Desarrollo de una plataforma web para la visualización de estadísticas obtenidas en base a los datos almacenados.
- Casos de estudio descritos en el punto **8** dónde podremos ver el funcionamiento aplicado en diversos ámbitos.

1.3. Metodología

Dado que el proyecto se basa en el desarrollo de un software para la extracción de datos, se va a seguir una fusión entre las siguientes dos metodologías.

Ciclo de vida de desarrollo de software (SDLC) (1)

1. Planificar: Fase de identificación y análisis de los requisitos del proyecto.
2. Diseño: A partir de los requisitos obtenidos en la fase de análisis, se comienza con el proceso de diseño en el cual se plantean los diagramas de secuencia y los diseños de interfaz.
3. Desarrollo: Implementación de los componentes necesarios para cumplir los requisitos previos.
4. Pruebas: En esta fase se comprueba que los resultados obtenidos sean los correctos según los requisitos.

Ciclo de vida de un proyecto de extracción de datos (CRISP-DM) (2)

1. Comprensión del negocio: Primera fase para comprender el problema y como enfocarlo a la minería de datos.
2. Estudio de los datos: Recolectar unos primeros datos para familiarizarse y comprenderlos. Se identifican problemas que puedan generar, así como beneficios que nos puedan aportar.

3. Preparación de los datos: Se realiza una limpieza de los datos obtenidos para obtener un conjunto final con información útil que facilitan el estudio de estos datos.

4. Modelado: Se aplican técnicas de modelado para calibrar y mejorar estos datos a valores óptimos.

5. Evaluación: Se comparan los objetivos de negocio con los resultados obtenidos.

6. Implantación: Se plasmas los datos obtenidos para el usuario final.

Metodología empleada en el proyecto

1. Planificar: Se ha identificado el problema y los requisitos necesarios para satisfacer el objetivo de este proyecto.

2. Estudio y comprensión de los datos: Estudio de las herramientas disponibles para la obtención de datos y extracción de una colección pequeña (1000 noticias) para construir una estructura común en la base de datos.

3. Diseño: Esquema del desarrollo software y de las diferentes interfaces para la visualización de los datos.

4. Desarrollo: Implementación de un sistema de obtención de datos.

5. Preparación de los datos: Limpieza y verificación de los datos.

6. Pruebas: Realizar diversos casos de prueba para comprobar el correcto funcionamiento de la aplicación.

7. Implantación: Pasos necesarios para poner en funcionamiento el sistema.

1.4. Estructura

Comenzaremos con la introducción, la motivación y los objetivos que nos han llevado a realizar este proyecto. Continuemos con el estado del arte donde hablaremos de proyectos o estudios ya realizados y que mejoras hemos abordado en este proyecto respecto al resto de proyectos existentes. En el capítulo 3 abordaremos en profundidad la solución propuesta para realizar el proyecto. En el capítulo 4 hablaremos del diseño de la solución donde expondremos las tecnologías utilizadas, la arquitectura del sistema y un análisis más detallado de las herramientas desarrolladas. En el capítulo 5 extenderemos lo explicado en el capítulo 4 pero profundizando más en la parte de desarrollo e implementación. En el capítulo 6 presentaremos la información necesaria para la puesta en marcha de las herramientas. Todo proyecto software necesita el desarrollo de pruebas manuales o automáticas por lo que en el capítulo 7 hablaremos de las pruebas realizadas para comprobar el correcto funcionamiento de las herramientas. Para dar un punto de vista más real, hemos realizado diversos casos de estudio que demuestran la utilidad del proyecto. Finalizaremos con unas conclusiones y trabajos futuros para mejorar el proyecto.

2. Estado del arte

En las dos últimas décadas, con el creciente uso de internet y las nuevas tecnologías, muchos periódicos, revistas y otras fuentes de información que solo informaban en formato físico, se han ido adaptando y han creado un nuevo termino llamado “prensa digital”. Esto ha hecho que la información sea más accesible, dinámica y visual, consiguiendo un mayor alcance e impacto en la sociedad. Por ello, han surgido nuevas empresas de periodismo que han dado el salto directamente a publicar artículos en la red sin formato físico. Entre las distintas fuentes de información existentes, este proyecto se va a basar en los artículos extraídos de periódicos digitales.

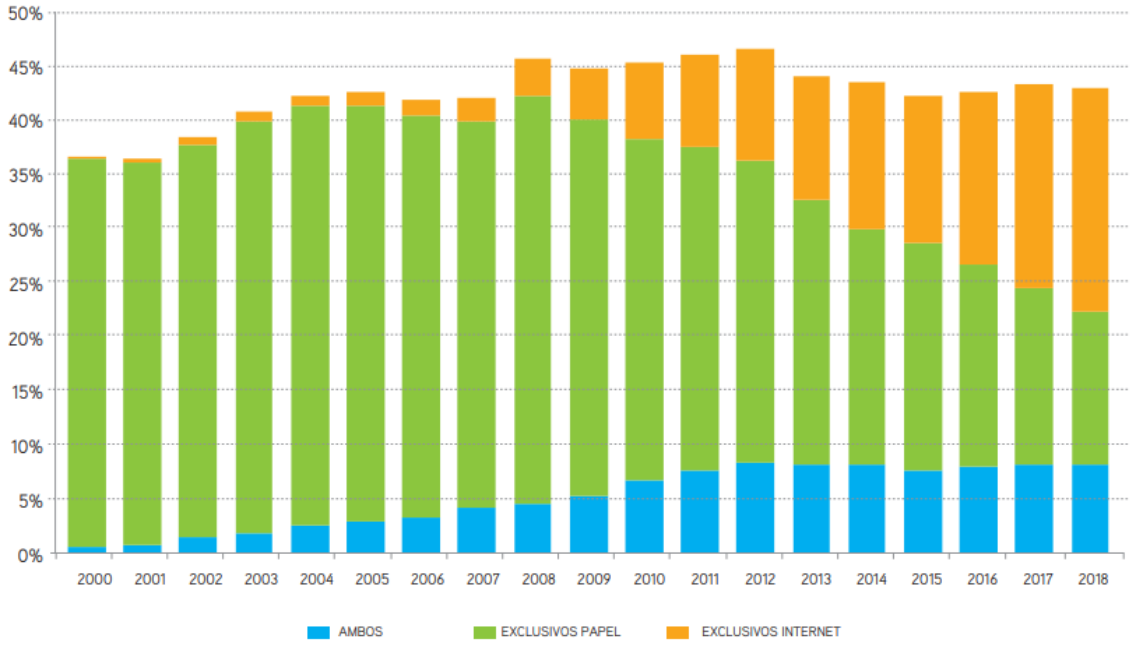

Figura 1. Evolución de la audiencia de diarios en España **(3)**

En la **Figura 1** se muestra la evolución de la audiencia de diario en España entre los años 2000 y el 2018. Como se observa, el porcentaje de audiencia de internet para la lectura de los diarios cada año aumenta proporcionalmente con la disminución de la audiencia del uso de diarios en papel.

Los diarios digitales, además de publicar artículos, también publican gráficas, imágenes y videos dónde la información que se quiere transmitir a veces cobra de más interés para el usuario. Por otro lado, los usuarios pueden transmitir su aprobación, opinión sobre un tema o rechazo por lo que han leído mediante comentarios en los artículos publicados. Esto nos lleva a más información añadida a parte de los propios artículos.

Toda esta gran cantidad de contenido que se genera diariamente abre otras ramas de investigación, que, gracias al uso de herramientas de explotación de datos, somos capaces de extraer y posteriormente analizar. Esta extracción de datos nos permite

generar un conjunto enorme y estructurado de textos también llamado como corpus lingüístico. Hay proyectos desarrollados e impulsados por instituciones y/o empresas que han generado corpus de una gran cantidad de texto, además, han hecho uso de herramientas que permiten el análisis y estudio del corpus. A continuación, vamos a explicar varios corpus existentes y cuales están enfocados a periódicos digitales.

2.1. Corpus y herramientas similares

Hay muchos corpus publicados en la web, la mayoría de estos corpus se encuentran en inglés, aunque también hay una importante cantidad en español y otros idiomas.

La Real Academia Española (RAE) proporciona desde su web (https://www.rae.es/recursos/banco-de-datos) en el apartado “Banco de datos” los siguientes corpus:

- Corpus del Español del Siglo XXI (CORPES XXI) (4): Es un “corpus de referencia” el cual sirve para sacar estadísticas de la lengua en un momento determinado de la historia. Está patrocinado por el Banco Santander y presenta un corpus formado por texto de diferentes ámbitos (libros, revistas, periódicos, blogs, etc.). Desde diciembre de 2019 cuenta con más de 285.000 documentos.

- CREA (5): Al igual que CORPES XXI, es un corpus de referencia. Su última versión es de junio de 2018 y alberga más de ciento sesenta millones de forma provenientes de textos escritos y orales, producidos en diferentes países de habla hispana.

- CDH (6): El corpus Nuevo diccionario histórico del español, consta de 355740238 registros formados por texto españoles y obras americanas, textos escritos entre el siglo XII y 1975 y obras datadas entre 1975 y 2000.

- CORDE (7): En un corpus textual de todas las épocas y lugares donde se habló español. Se inicio en el 1974 y su utilidad es la de estudiar las palabras y sus significados, así como la gramática y su uso a través del tiempo.

En la **Figura 2** se muestra una plataforma de la RAE para estudiar los cuatro corpus que proporcionan. Todos ellos están disponibles para estudiarlos a través de una plataforma web en la cual puedes buscar lemas, formas y clases de palabras para sacar estadísticas o concordancias.

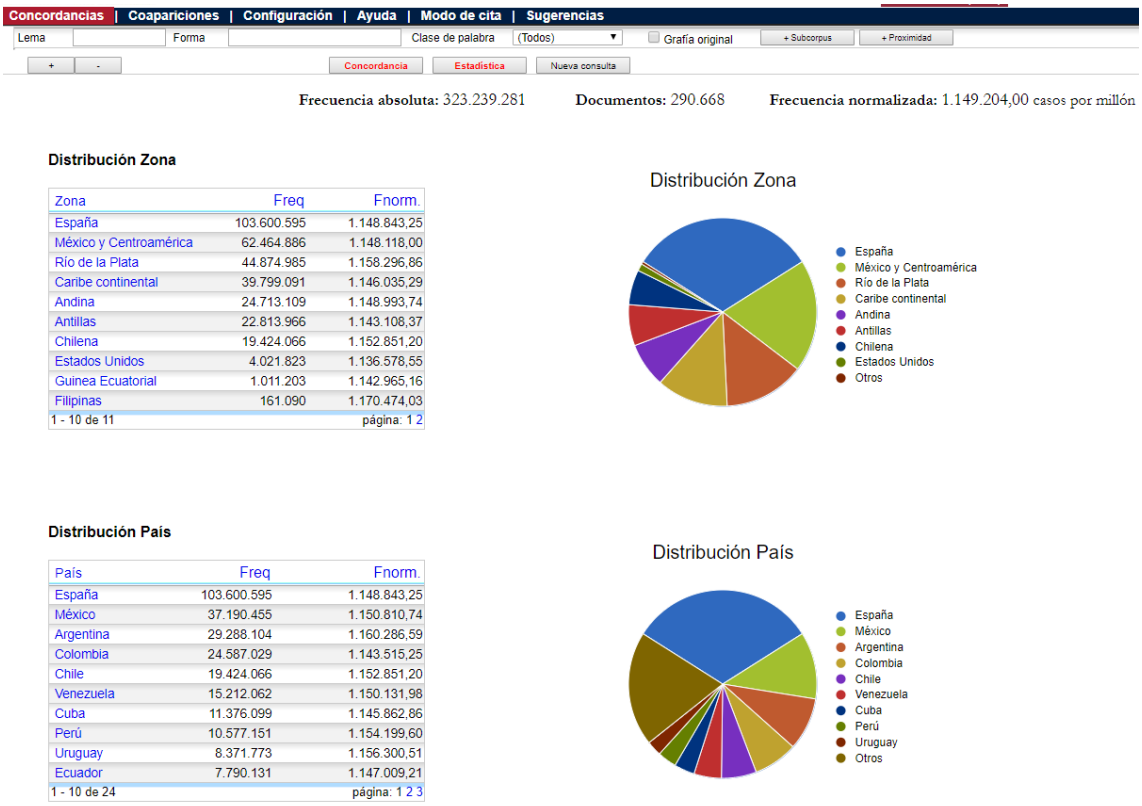

Figura 2. Plataforma web de la RAE para el estudio de los diferentes corpus que proporcionan

Hay otros corpus enfocados a extraer datos de los periódicos digitales como el proyecto Aracne (8), enfocado en el estudio de la variación de la riqueza lingüística en la prensa española desde 1914 hasta 2014. Su corpus está constituido por 5167 artículos, 84 900 oraciones y 1 921 566 palabras, se ha utilizado para estudiar la riqueza léxica y su densidad además de la complejidad semántica, sintáctica y conceptual. Este corpus no está accesible para su estudio.

Otro corpus actualmente activo es el Corpus del Español NOW (News on the Web) (9), contiene cerca de 7,2 billones de palabras de 21 países diferentes de habla hispana y está basado en el contenido de revistas y periódicos desde el 2012 hasta hoy. Cada mes crece cerca de 170-180 millones de palabras provenientes de 200.000 - 250.000 artículos nuevos. Actualmente el acceso a este corpus se rige a través de una interfaz (**Figura 3**) que proporcionan en la propia web donde se puede estudiar y hacer consultas, ofreciendo funcionalidades como sintaxis básica, concordancia, lista de frecuencias, colocaciones, comparación de palabras y comparación de dialectos.

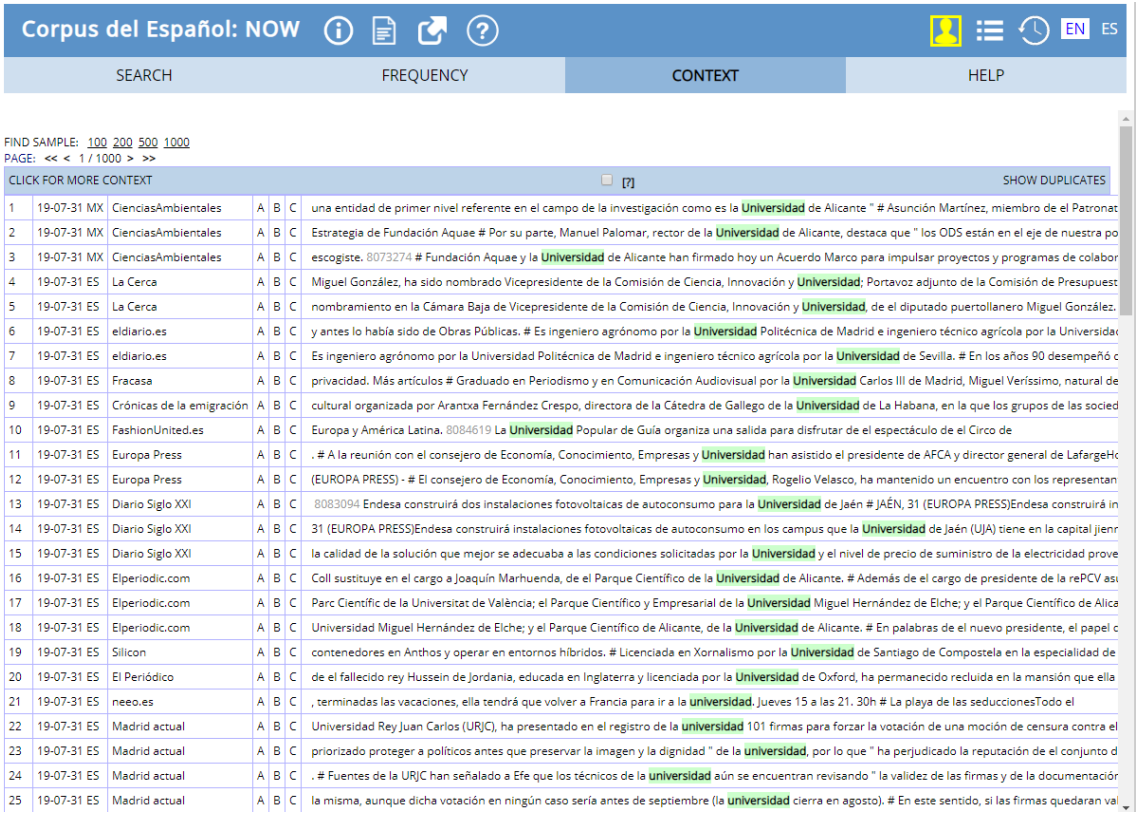

Figura 3. Interfaz para estudiar el corpus NOW

El corpus Timestamped JSI web (10) está formado por artículos de periódicos digitales extraídos de servidores RSS desde el año 2014 hasta la actualidad. Está en activo y cada día procesa 75.000 RSS de los que extrae entre 100.00 y 150.000 artículos. Su corpus en español tiene más de 8,3 billones de palabras y cada mes crece cerca de 150 millones de palabras. Su corpus desde el 2014 hasta el 2016 está disponible para usuarios de prueba y el corpus actualizado hasta la actualidad es accesible para usuario suscritos, donde solo podrán consultar el corpus a través de la propia web. Además, estos usuarios disponen de una herramienta donde se puede estudiar el corpus. En la **Figura 4** se muestra parte de la herramienta la cual proporciona una funcionalidad amplia como colocaciones categorizadas por relaciones gramaticales, sinónimos y palabras similares, extracción de terminologías, numero de palabras por categoría gramatical, concordancia, n-gramas, tendencias, etc.

Figura 4. Interfaz para el estudio del corpus creado por Sketchengine **(10)**

Otro corpus que adquiere su contenido de periódicos es el “Corpus del molinero” creado por Hemero (11). Este corpus es un corpus formado a partir de artículos de prensa de España, Argentina y México, que datan del 1997 hasta el 2009. Su tamaño es de más de 660 millones de palabras y cuenta con 1.700 millones de artículos. Su corpus no es accesible, pero proporciona una herramienta en la web para estudiarlo. En la **Figura 5** se muestra una parte de la interfaz donde se puede hacer uso de diferentes funcionalidades como buscar por texto, título o sección de un artículo a partir de una palabra o una frase.

Por último, el consorcio abierto entre universidades, empresas y laboratorios gubernamentales de investigación llamado Linguistic Data Consortium (12) se encarga de crear, recopilar y distribuir bases de datos de habla y texto, léxicos y otros recursos para fines de investigación y desarrollo lingüístico. En español hay varios corpus obtenidos de diversos ámbitos. El corpus Spanish News Text es un corpus solo en español compuesto de textos periodísticos extraídos de diferentes periódicos y agencias de noticias de hispanoamérica entre el año 1995 y 1996.

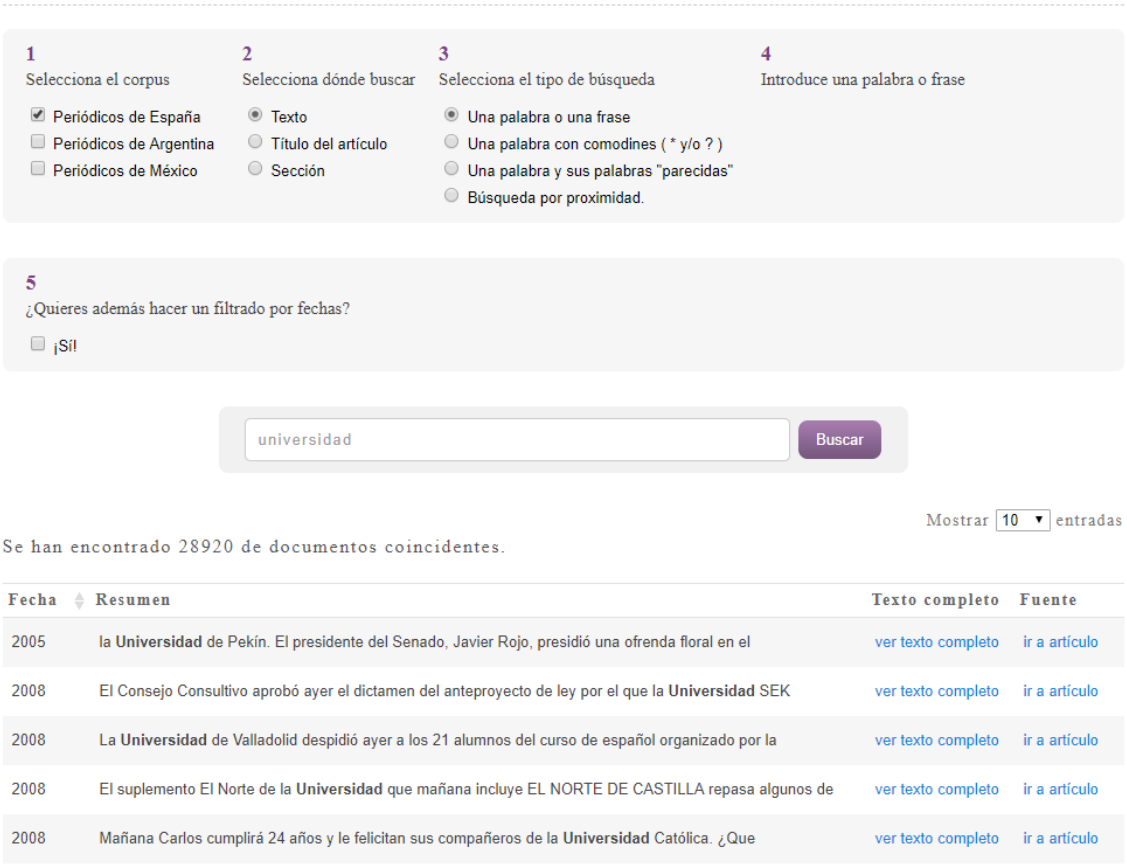

Figura 5. Interfaz para el estudio del corpus molinero **(11)**

2.2. Crítica al estado del arte

Como hemos visto, hay una gran cantidad de corpus disponibles en la red, algunos de ellos previo pago y otros accesibles a través de una interfaz con la que interactuar y hacer un estudio sobre el corpus. Muchos de ellos adquieren los datos de libros, revistas, artículos, etc., pero son pocos los que su totalidad del corpus viene extraída de periódicos digitales. Además, algunos son proyectos puntuales o estudios que no perduran en el tiempo o simplemente no proporcionan la accesibilidad al corpus de forma libre o a la herramienta con la que han generado el corpus. Nuestra herramienta, está diseñada para extraer artículos de tres periódicos digitales, siendo posible agregar más periódicos en un futuro. Cada usuario podrá extraer su corpus y analizarlo con la plataforma web desarrollada en este proyecto o hacer uso del corpus para, por ejemplo, aprender modelos de lenguaje natural.

3. Análisis del problema

El objetivo principal de este proyecto es la adquisición de datos de diferentes portales web y el posterior análisis. Se ha enfocado esta adquisición a portales web de diarios digitales.

El criterio elegido para seleccionar los distintos portales web ha sido por importancia a nivel nacional o internacional y por importancia a nivel de la Comunitat Valenciana. El criterio por importancia se ha basado en las estadísticas proporcionadas por los principales sistemas de medición de audiencia de España.

Si observamos el listado de portales más visitados según el Estudio General de Medios (EGM) a fecha de 5 de Julio del 2019 los periódicos digitales más visitados son El País, Marca y la Vanguardia (13). Como diario digital de la Comunitat Valenciana, se ha optado por el Levante-EMV (14), y como diario digital internacional el Daily Mail Online (15) ya que son de los más visitados a nivel autonómico e internacional respectivamente.

Aunque la cantidad de periódicos digitales sea pequeña respecto a la cantidad real de periódicos que existen, la cantidad de datos será muy grande, por lo que se plantea un problema propio de explotación de datos.

Para abordar el problema se hace necesaria una herramienta de adquisición de datos automática y una plataforma para visualizar y analizar estos datos adquiridos.

3.1. Análisis de datos

Para poder obtener toda la información de un artículo en un portal web es necesario entender como es la estructura de un artículo. Esta estructura se basa en etiquetas HTML. Más adelante se explicarán técnicas para analizar documentos HTML, que herramientas utilizar y las principales diferencias de las etiquetas HTML de las fuentes de datos analizadas.

HTML (HyperText Markup Language)

Es un lenguaje de marcado específico para el diseño de páginas web. La información se estructura mediante etiquetas con o sin atributos donde estas pueden aportar un diseño visual al contenido.

```
<h1>El tiempo en Valencia</h1>

<p>El tiempo en Valencia en Semana Santa será cuanto menos cambiante…</p>
```

Como vemos en el ejemplo, el uso de las etiquetas HTML hace los documentos difíciles de leer y manipular. Por ello, se debe almacenar la información necesaria en un formato que facilite la transmisión de datos e incorpore campos o etiquetas que

describan el contenido que albergan. Dos formatos con estas características son los siguientes.

JSON (JavaScript Object Notation)

Es un formato de texto para la transmisión de datos. Su notación basada en pares "atributo: valor" facilitan la conversión a un objeto JavaScript. Su utilización está muy extendida en la representación de objetos en páginas web.

```
{
  "noticias": [{
    "titulo": "El tiempo en Valencia",
    "texto": "El tiempo en Valencia en Semana Santa será cuanto menos cambiante…"
  }]
}
```

XML (eXtensible Markup Language)

Es un metalenguaje estructurado jerárquicamente utilizado para almacenar datos de forma legible. Al igual que HTML la información esta encapsulada entre etiquetas, sin embargo, estas etiquetas definen o nombran el contenido que se alberga en ellas.

```
<?xml version="1.0"?>
<noticias>
   <titulo>El tiempo en Valencia </titulo>
   <texto>El tiempo en Valencia en Semana Santa será cuanto menos cambiante…</texto>
</noticias>
```

Una vez definidos estos dos tipos de estructura de datos se va a valorar los aspectos positivos **Tabla 1** y aspectos negativos **Tabla 2** de estos lenguajes de marcas. Esto nos ayudará a justificar la elección realizada.

Aspectos positivos	
JSON	**XML**
• Estructura de fácil lectura. • Sus valores pueden representar diferentes tipos de datos tales como texto, números y listas. • Ocupa poco espacio. • La escritura y lectura es rápida.	• Estructura de fácil lectura. • Permite validación de la estructura.

Tabla 1. Aspectos positivos entre JSON y XML

Aspectos negativos	
JSON	**XML**
• Relativamente nuevo por lo que tiene menos soporte.	• Ocupa bastante espacio. • No se pueden representar listas de forma fácil. • Necesita ser procesado para ser utilizado por una API REST.

Tabla 2. Aspectos negativos entre JSON y XML

Aunque JSON sea mucho más nuevo tecnológicamente hablando, la fuerza con la que ha entrado y su utilidad es mayor a la del XML. Dado que es un proyecto que se acerca a recolección de cantidades grandes de datos (Big Data), necesitamos una estructura que ocupe el menor espacio posible, además de que sea sencilla y rápida la recuperación de datos en una base de datos no relacional. Por todo esto, el problema planteado sobre que estructura de datos elegir ha sido resuelta decantándonos por la estructura JSON.

3.2. Especificación de requisitos

En este apartado se van a especificar los requisitos de nuestro software. Aunque haya dos subsistemas en nuestra herramienta, parseadores de datos y el portal web que visualiza los datos analizados, se tratarán como un único software. Esta especificación de requisitos del software (ERS) se detallarán según el estándar IEEE830.

3.2.1. Propósito

Las ERS nos ayudará a describir el comportamiento completo del sistema y definir las interacciones que tendrá el usuario sobre este.

3.2.2. Ámbito del sistema

El sistema obtendrá todas las noticias de diversos periódicos digitales y permitirá ver un análisis de estas noticias a través de un portal web.

El sistema permitirá:

- Recolectar noticias usando diversos scripts.
- Almacenar toda la información adquirida.
- Analizar todas las noticias para sacar información útil.
- Explotar los resultados para obtener información estadística de interés.

- Realizar búsquedas simples para obtener resultados acotados.
- Guardar las estadísticas en formato imagen.

El sistema no permitirá:

- Ejecutar diariamente de forma automática los scripts. El usuario deberá ejecutar los scripts manualmente o programar su ejecución diaria.
- Realizar búsquedas complejas para filtrar los datos en el portal web.

Características detalladas

Característica	Descripción
CA01	Adquisición de datos vía scripts
CA02	Adquisición de datos entre dos fechas
CA03	Tipo de objeto a analizar: noticias
CA04	Portal web
CA05	Búsqueda por fecha y categorías
CA06	Descargar estadísticas en formato imagen

Tabla 3. Características del sistema

3.2.3. Definiciones, acrónimos

En la **Tabla 4** se muestran los términos empleados a la largo del proyecto junto a una breve descripción de estos.

Término	Descripción
Web Scraping	Técnica utilizada mediante programas de software para extraer información de sitios web.
CRUD	El acrónimo, en inglés, de "Crear, Leer, Actualizar y Borrar", usado para referirse a las funciones básicas en bases de datos o la capa de persistencia en un software.
Parsear	Es el proceso de analizar una cadena de símbolos, ya sea en lenguaje natural, lenguaje informático o estructuras de datos, de acuerdo con las reglas de una gramática formal.
Word embeddings	Es el nombre de un conjunto de lenguajes de modelado y técnicas de aprendizaje en procesamiento del lenguaje natural (PLN). Transforma el lenguaje humano en formas numéricas. Esta representación numérica tiene el objetivo de capturar significados, relaciones semánticas, similitudes entre palabras, y el contexto de los diferentes significados de una palabra.

Tabla 4. Términos utilizados en el proyecto

3.2.4. Descripción general

A continuación, se muestra un diagrama de casos de uso el cual ayuda a comprender cual es el rol del usuario frente al sistema.

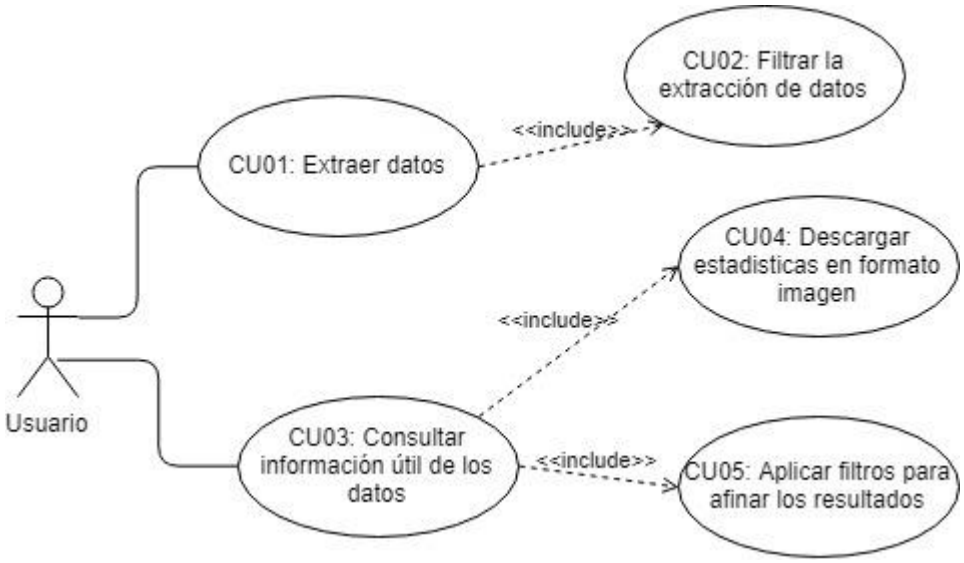

Figura 6. Diagrama de casos de uso

En la **Tabla 5** se puede observar la relación de los casos de uso representados en la **Figura 6** con las características descritas en la **Tabla 3**.

Caso de uso	Descripción	Característica
CU01	Extraer datos	CA01 y CA03
CU02	Filtrar la extracción de datos	CA02
CU03	Consultar información útil de los datos	CA04
CU04	Descargar estadísticas en formato imagen	CA06
CU05	Aplicar filtros para afinar los resultados	CA05

Tabla 5. Relación de los casos de uso con las características del sistema

3.3. Análisis de la propiedad intelectual

Como hemos comentado, la parte fundamental de este proyecto es la extracción de noticias de los periódicos El País, Levante-EMV y DailyMail. Estas noticias son escritas y publicadas por empresas o escritores, por lo que es conveniente explicar que uso se va a realizar con las noticas para no vulnerar la Ley de Propiedad Intelectual.

El artículo 10.1 de la Ley de Propiedad Intelectual (16) publicado en el BOE cita textualmente lo siguiente:

> "Son objeto de propiedad intelectual todas las creaciones originales literarias, artísticas o científicas expresadas por cualquier medio o soporte, tangible o intangible, actualmente conocido o que se invente en el futuro."

En la sección "Aviso Legal" de los periódicos El País (punto 5) (17) y Levante-EMV (punto 7) (18) expresan claramente que está prohibido el uso total o parcial del contenido publicado en la web con fines comerciales. El periódico DailyMail expresa que sólo el contenido con la "etiqueta" "Embeddable Content" es apto para ser usado con fines no comerciales y respetando una serie de términos legales (19).

Por estos motivos, no estamos autorizados a publicar ningún artículo, por ende, los corpus generados no estarán disponibles ni se publicarán.

4. Diseño de la solución

En este capítulo hablaremos de la arquitectura diseñada para nuestro sistema, así como de las tecnologías utilizadas que nos han permitido llevar a cabo el desarrollo y para finalizar profundizaremos en el diseño implementado.

4.1. Arquitectura del sistema

El sistema está dividido en tres componentes clave representados en la **Figura 7**. El primero de ellos está formado por los parseadores de datos que ejecutándose independientemente unos de otros, extraen todos los datos posibles de cada uno de los periódicos digitales elegidos. Estos datos se almacenan en el segundo componente, la base de datos no relacional MongoDB. De este componente se extraen los datos para realizar dos funciones, analizar los datos para convertirlos en información útil y representar esta información a modo de estadísticas. De estas dos funciones se encarga el tercer componente, un portal web desarrollado con el framework Django. En este proyecto se ha optado por incluir un conjunto que implemente una arquitectura o patrón de diseño. Django implementa el patrón modelo-plantilla-vista (Model-Template-View o MTV en inglés), una arquitectura similar al patrón modelo-vista-controlador (MVC). A continuación, se explica de forma general el patrón MVC, en el punto **4.3.2** se explica de forma detallada el patrón usado por Django.

Patrón MVC

MODELO

Es la capa en la que se trabaja con los datos, o capa de negocio. Se encarga de manipular, gestionar y actualizar los datos. En este portal web realizaría todas las tareas de comunicación con la base de datos.

VISTA

Se encarga de representar la interfaz con la información obtenida por el modelo. La información representada cambia a través de las acciones que realiza el usuario en la interfaz.

CONTROLADOR

Recibe eventos de entrada del usuario y a partir de ellos trata los datos a través del modelo y actualiza la vista con los nuevos datos.

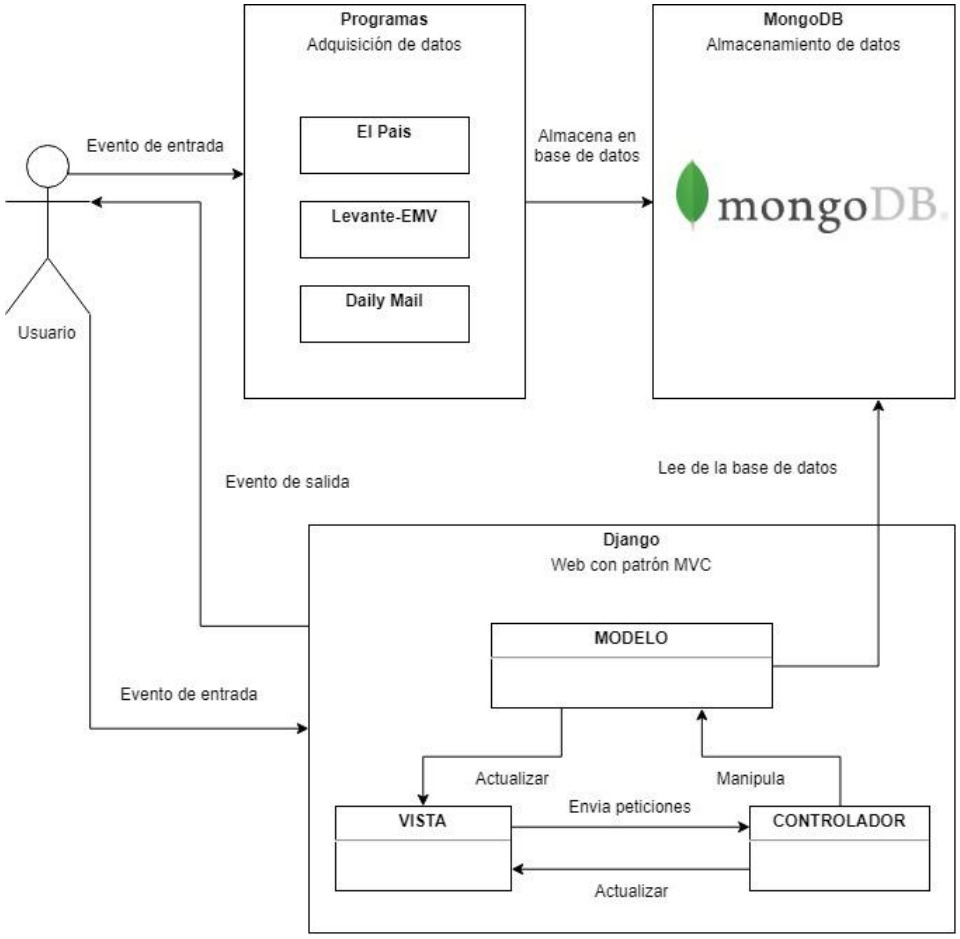

Figura 7. Diagrama UML de la arquitectura del sistema

4.2. Tecnologías utilizadas

En este punto comentaremos que librerías y paquetes desarrollados para Python nos han permitido extraer y consultar los datos necesarios para el proyecto, así como las tecnologías y frameworks necesarios para el almacenamiento de datos y la visualización de los datos analizados.

4.2.1. Request

Request es una librería HTTP de Python. Su objetivo principal es permitir hacer peticiones HTTP de una forma más fácil y amigable. La necesidad de utilizar Request se debe a que la librería descrita en el siguiente apartado (Beautiful Soup 4) necesita el contenido de la respuesta recibida por el servidor. Este contenido será el código fuente de la URL solicitada, el cual es necesario para realizar web scraping.

Python ofrece una librería llamada Urllib, que permite realizar peticiones HTTP. Sin embargo, esta librería está anticuada e implica tener que hacer peticiones manualmente

así produciendo un código de programación más complicado y extenso. Por este motivo se ha optado por utilizar Request.

La librería Request proporciona los métodos más conocidos en cuanto a peticiones HTTP (*put*, *get*, *post*, *head*, *delete*, *options*). Estos métodos se encargan de indicar que tipo de petición se quiere hacer, como, por ejemplo, obtener (Get), añadir (Post), modificar (*Put*) o borrar (*Delete*) información. Estas peticiones permiten realizar acciones sobre recursos alojados en una URL, además, permite el uso de parámetros opcionales y la capacidad de enviar datos a través del *Body* del mensaje de la petición, el cual es utilizado por los métodos *Post y Put*.

A continuación, se muestra un ejemplo de cuatro peticiones HTTP distinta.

- Modifica el título de la noticia con el identificar 10.

```
requests.put('https://ejemplo.es/noticia/10', data = {‘titulo’:’Titulo de la noticia’})
```

- Borra la noticia con el identificador 15.

```
requests.delete(‘https://ejemplo.es/noticia/15’)
```

- Añade una nueva noticia.

```
requests.post(‘https://ejemplo.es/noticia’, data = {‘titulo’:’Titulo de la noticia’})
```

- Obtiene todas las noticias.

```
requests.get(‘https://ejemplo.es/noticias’)
```

4.2.2. Beautiful Soup 4

Beautiful Soup 4 (BS4) es un paquete de Python que permite parsear documentos HTML y XML para extraer datos de ellos a partir de una estructura con etiquetas. Existen librería para parsear como Scrapy, lxml o Selenium.

Los motivos que me han hecho utilizar BS4 en este proyecto son:

- Selenium está enfocado a páginas web con JavaScript.
- BS4 permite parsear el contenido con uno de los parseadores más rápidos llamado lxml.
- La velocidad de la que destaca Scrapy se puede solventar con la ejecución por hilos de Python 3.

BS4 no es del todo un analizador de documentos, es una interfaz que nos proporciona la posibilidad de utilizar otros parseadores: lxml, html.parser, html5lib y lxml-xml.

Las principales diferencias entre los parseadores disponibles son las siguientes:

	Ventajas	Desventajas
lxml	Muy rápido	Se necesita instalar el módulo lxml
html.parser	Rápido	Baja tolerancia a fallos
html5lib	Analiza cualquier documento HTML incluso si este no está bien construido	Lento Se necesita instalar el módulo html5lib
lxml-xml o xml	Muy rápido	Solo analiza documentos XML

Tabla 6. Ventajas y desventajas de los diferentes parseadores

Dado que los documentos a analizar son de tipo HTML, no podemos utilizar la opción “lxml-xml”. Si analizamos las ventajas y desventajas, la característica más importante en herramientas de extracción de datos es la velocidad con la que los analiza. Se ha realizado un pequeño estudio para determinar que parseador es el más apropiado para nuestra aplicación.

Análisis de velocidad

Se ha realizado un programa que muestra el tiempo que tarda cada uno de los parseadores disponibles en analizar una URL n veces seguidas. El tiempo medido en segundos representa únicamente la descarga de la URL y el posterior parseo de todo el contenido de la URL realizado por BS4 con cada parseador, no se incluye el tiempo de guardado en la base de datos ni la obtención de datos como el cuerpo de la noticia o el título porque consideramos que estos procesos requieren un tiempo despreciable.

Parámetros utilizados:

- URL: https://elpais.com/ccaa/2019/05/19/catalunya/1558262174_867037.html
- Parseadores: lxml, html.parser y html5lib
- Cantidad de repeticiones: 10, 100, 1000 y 2000.

El programa ha sido ejecutado tres veces para obtener diversos resultados y así descartar posibles demoras por agentes externos que hayan podido alterar los tiempos. Los resultados obtenidos han sido similares por lo que en la **Tabla 7** se representa la media de los resultados de las tres ejecuciones.

	1 URL	100 URL	1000 URL	2000 URL
lxml	1,29 s	12,70 s	126,17 s	225,64 s
html.parser	1,24 s	13,16 s	136,56 s	246,90 s
html5lib	1,74 s	17,75 s	178,05 s	344,72 s

Tabla 7. Tiempo medio en segundo requerido por los diferentes parseadores

Como vemos en la **Tabla 7**, el analizador html5lib se aleja bastante de los resultados obtenidos por el lxml y html.parser. Estos dos presentan resultados bastante similares, sin embargo, parece ser notable una tendencia en la que el html.parser cuanto más URL tiene que analizar más se aleja de los resultados obtenidos por el lxml.

Por su velocidad respecto al resto de opciones y por su capacidad de analizar documentos HTML estructurados por etiquetas, se ha optado por utilizar lxml.

Se presenta a continuación una consulta simple para extraer el código fuente de una cierta URL donde “r.text” es el contenido de la respuesta obtenida con Request y “lxml” es el tipo de analizador que se utiliza.

```
r = requests.get(“http://www.elpais.com”)

soupPage = BeautifulSoup(r.text, "lxml")
```

4.2.3. MongoDB

La parte fundamental de este proyecto es la extracción de información. La mayor parte del tiempo requerido por un programa de extracción de datos se centra en analizar aquello que queremos extraer, en nuestro caso, noticias de diferentes periódicos digitales, por ello, es recomendable que esa demora que requiere el análisis no se vea más perjudicada por otros procesos, como puede ser el almacenamiento o la lectura de datos. Para satisfacer esta característica se ha optado por la utilización del sistema de gestión de base de datos MongoDB.

MongoDB es un sistema de base de datos NOSQL, esto es, la información es guardada en documentos con una estructura similar a JSON llamada BSON (Binary JSON), a diferencia de las bases de datos relacionales convencionales, donde la información se almacena en tablas. La estructura de una base de datos MongoDB se basa en colecciones de documentos donde cada documento puede tener una estructura diferente al resto de documentos de esta colección (**Figura 8**).

Cuando se utiliza una base de datos MongoDB, para una mejor organización de la información se recomienda almacenar en una colección aquellos documentos que tengan una estructura similar. En este proyecto se han decidido almacenar los documentos de noticias de un mismo diario en una sola colección, por lo que la estructura de los documentos BSON son prácticamente la misma.

Los puntos fuertes de MongoDB son la sencillez y la rapidez, conseguidas en parte por su estructura binaria. Aunque esta estructura puede llegar a utilizar una mayor cantidad de espacio en memoria frente al JSON.

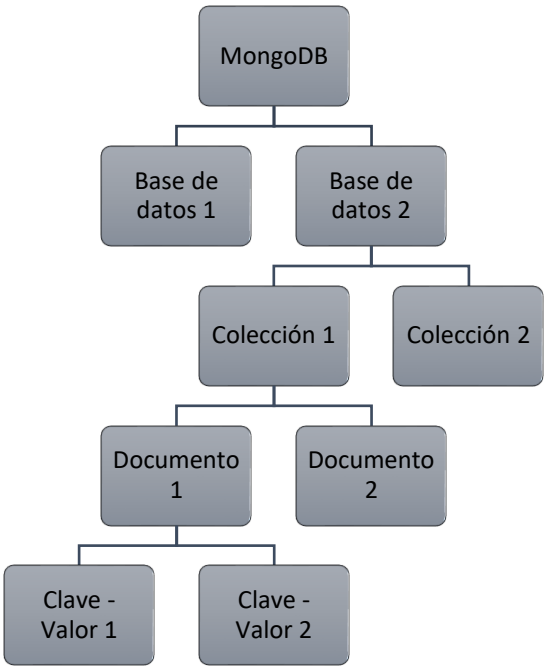

Figura 8. Estructura de MongoDB

MongoDB al ser NOSQL permite como ya hemos dicho una estructura no idéntica entre todos sus documentos, por lo que es una forma dinámica y flexible de guardar información. En cuanto a términos de escalabilidad las bases de datos NOSQL como MongoDB, ofrecen una escalabilidad horizontal que permite añadir más servidores o nodos al sistema, en vez de la escalabilidad vertical que ofrecen los sistemas SQL los cuales solo disponen de un nodo. La escalabilidad horizontal resuelve el problema de la disponibilidad, permitiendo utilizar el resto de los servidores ante cualquier fallo de otro servidor, donde en los sistemas SQL esto resulta más complejo debido a su rigidez estructural.

	SQL	**NOSQL**
Estructura	Basada en tablas	Basada en ficheros JSON
Esquema	Estático	Dinámico
Escalabilidad	Vertical	Horizontal
Modelo de datos	Relacional	No relacional

Tabla 8. Resumen de características de sistemas SQL y NOSQL

Las razones por las que se ha utilizado MongoDB en este proyecto son dos. Su estructura JSON permite con facilidad el tratamiento de datos ya que no necesita ningún tipo de transformación de datos. Segundo, los artículos extraídos de diarios o los comentarios de un foro puede que no tengan los mismos atributos, por ejemplo, una noticia puede tener o no un subtítulo.

4.2.4. PyMongo

Para extraer los datos almacenado en la base de datos MongoDB necesitamos una librería con la que extraer esos datos desde Python. Por ello se ha elegido la librería PyMongo que nos permite establecer una conexión directa con MongoDB y realizar consultas sencillas y/o complejas como "*create*, *read*, *update*, *delete*" (CRUD). A continuación, se muestra un pequeño ejemplo de utilización de PyMongo donde se establece una conexión y se recibe una colección en su totalidad.

```
client = MongoClient('localhost', 27017)

db = client.test_database

collection = db.test_collection
```

4.2.5. Django

En este proyecto se han obtenido muchos datos de diferentes periódicos digitales. El análisis de estos datos es útil para sacar conclusiones. Una buena forma de mostrar estos análisis es representando los datos de forma gráfica.

Hay diversas formas de representar los datos obtenidos, podría desarrollarse una aplicación de escritorio, una página web, una aplicación móvil… En nuestro caso, se ha optado por un portal web desarrollado bajo un Framework que facilite su creación.

Un Framework es un entorno de trabajo que nos ayuda a enfocar de mejor forma el desarrollo de un software específico. La principal tarea de un framework es proporcionar una estructura del proyecto con clases y módulos que facilitan tareas como conectarse a una base de datos, hacer peticiones http o proporcionar una plantilla base con la que diseñar una web. La mayoría de frameworks implementan y hacen uso del patrón de diseño MVC explicado en el punto **4.1**. Además, proporcionan una guía de estilos que ayuda a realizar una buena praxis de programación. Existen diversos frameworks enfocados al desarrollo de plataformas web. A continuación, en la **Tabla 9** se muestran algunos de los frameworks más utilizados hoy en día.

Nombre	Lenguaje
Django	Python
Pyramid	Python
Flask	Python
Ruby on Rails (RoR)	Ruby
Laravel	PHP
Angular	TypeScript
React	JavaScript

Tabla 9. Frameworks web más populares

Los frameworks web más utilizados en webs convencionales son React y Angular, hoy en día están en constante crecimiento y son los términos más buscados en portales como Google y StackOverflow. En la **Figura 9** se muestran los frameworks más buscados en StackOverflow, el primero enfocado a Python es Django con un 13%.

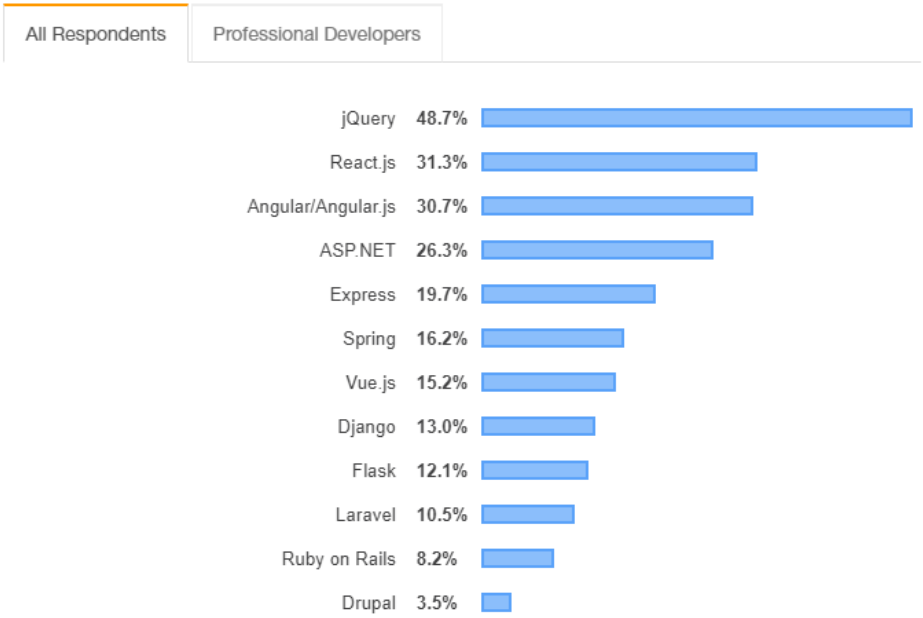

Figura 9. Frameworks más populares **(20)**

Si comparamos los frameworks no desarrollados bajo el lenguaje de programación Python (Angular, React, Laravel, RoR), vemos que tienen similitudes con Django, por ejemplo, que están basados en programación orientada a objetos. Dado que Angular y React no están implementados bajo el patrón de diseño MVC, quedan descartados. Laravel y RoR sí que implementan el patrón MVC, pero su lenguaje de programación no es Python, y en este proyecto necesitamos de librerías especializadas que nos proporcionen técnicas de manipulación de texto avanzadas y librerías de fácil conexión con MongoDB.

Por lo tanto, nos queda saber porque usar Django y no Flask o Pyramid. Elegir uno de estos tres resulta difícil, ya que son muy similares. Presentan pequeñas diferencias como que Django tiene soporte con los principales sistemas de almacenamiento como MySQL. Flask, sin embargo, destaca por su fácil aprendizaje para realizar pequeños proyectos y Pyramid es el más flexible de los tres, es usado tanto para aplicaciones pequeñas como para grandes proyectos (21).

Para concluir esta sección, queda decir que Django ha sido elegido por su popularidad, por su modelo MVC y por estar enfocado al lenguaje de programación Python.

4.3. Diseño detallado

Como hemos presentado en la introducción del punto **4.1**, nuestro sistema se compone de tres elementos:

- Parseadores de datos.
- Django.
- MongoDB.

A continuación, se presenta el diseño desarrollado de los parseadores y la plataforma web Django que nos ha permitido satisfacer las necesidades de nuestra aplicación.

4.3.1. Parseadores de datos

Se han desarrollado tres programas bajo el lenguaje de programación Python. Cada uno extrae información de un periódico digital, estos son, El País, Levante-EMV, Daily Mail.

Cada uno de estos programas se dedica a obtener todas las URL de los artículos publicados comprendidos entre dos fechas. Seguidamente extrae todos los datos en crudo y finalmente almacenan esos datos en una colección de la base de datos MongoDB.

Están implementados de forma desacoplada, esto es, cada uno de ellos no depende del otro y pueden ser ejecutados de forma paralela. Sin embargo, todos ellos guardan la información en una misma base de datos, no así, en una misma colección de MongoDB.

En la **Figura 10** se muestra el diagrama de secuencia de los recolectores de datos relativo al caso de uso 1 (CU01). El usuario trabajará a través de un terminal en el cual tendrá que ejecutar el script y escribir diversos parámetros de entrada como el rango de fechas en la que los artículos hayan sido creados y el nombre de la colección MongoDB. Una vez se obtenga todas las URL de los artículos publicados entre las dos fechas introducidas, el sistema accederá y obtendrá cada uno de los artículos en el portal del periódico digital para luego extraer los datos más relevantes y almacenarlos en una colección MongoDB especificada por el usuario.

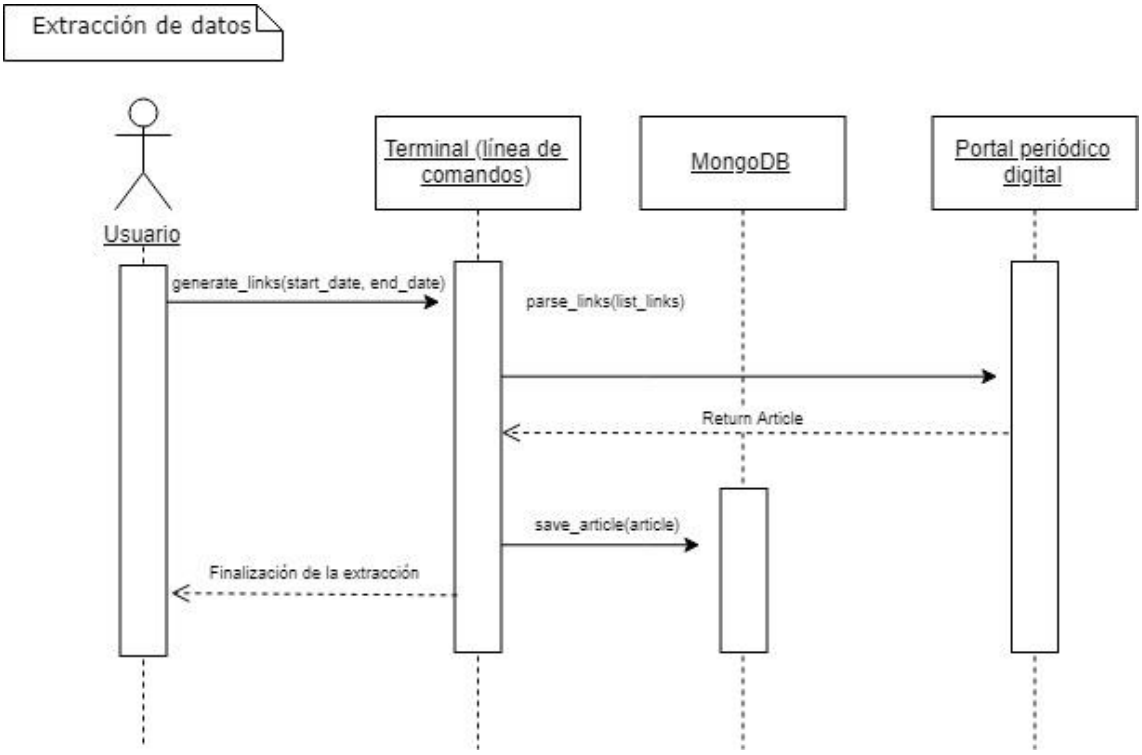

Figura 10. Diagrama de secuencia de los parseadores de datos (CU01, CU02)

4.3.2. Django

Django implementa el patrón de diseño Model-Template-View (MTV). En el punto **4.1** hemos explicado el patrón de diseño MVC y sus características. Sin embargo, Django hace uso de su propio patrón de diseño.

Patrón MTV (22)

- M significa "Model" (Modelo), es la capa de persistencia y tiene acceso a la base de datos. Esta capa contiene toda la información sobre los datos: cómo acceder a estos, cómo validarlos, cuál es el comportamiento que tiene, y las relaciones entre los datos.

- T significa "Template" (Plantilla), la capa de presentación. Esta capa contiene las decisiones relacionadas a la presentación: como algunas cosas son mostradas sobre una página web u otro tipo de documento.

- V significa "View" (Vista), la capa de la lógica de negocios. Esta capa contiene la lógica que accede al modelo y la delega a la plantilla apropiada.

La principal diferencia entre MVC y MTV es quien se responsabiliza de las diferentes capas. En el patrón MVC, la capa de presentación es parte de la vista (V) y la capa de negocio es parte del controlador (C), por el contrario, en el patrón MTV, son la plantilla (T) y la vista (V) quienes se responsabilizan respectivamente de esas capas.

4.3.2.1. Capa de persistencia

La capa de persistencia tiene acceso a MongoDB como hemos comentado anteriormente. En MongoDB se almacenarán todas las noticias obtenidas por los parseadores en diferentes colecciones según al periódico digital que pertenezcan. El fichero models.py con la ayuda de PyMongo tendrá todas las funciones necesarias de consulta a la base de datos MongoDB.

En la base de datos tenemos cuatro colecciones. Cada uno de los periódicos que usamos para obtener las noticias son una colección. Por otro lado, tenemos una colección creada exclusivamente para su uso en la plataforma web.

Como vemos en la **Figura 11** los documentos de las colecciones de los periódicos digitales tienen ocho campos, algunos de ellos de diferente naturaleza que en conjunto conforman una noticia.

La colección “disponibles” contiene documentos con tres campos:

- nombre: Nombre del periódico.
- value: Nombre clave de la colección del periódico.
- idioma: Idioma en la que el periódico publica las noticias.

Estos campos nos sirven para identificar los periódicos digitales que hemos usado y su idioma, de esta forma podremos establecer diferentes criterios en la parte de análisis de datos.

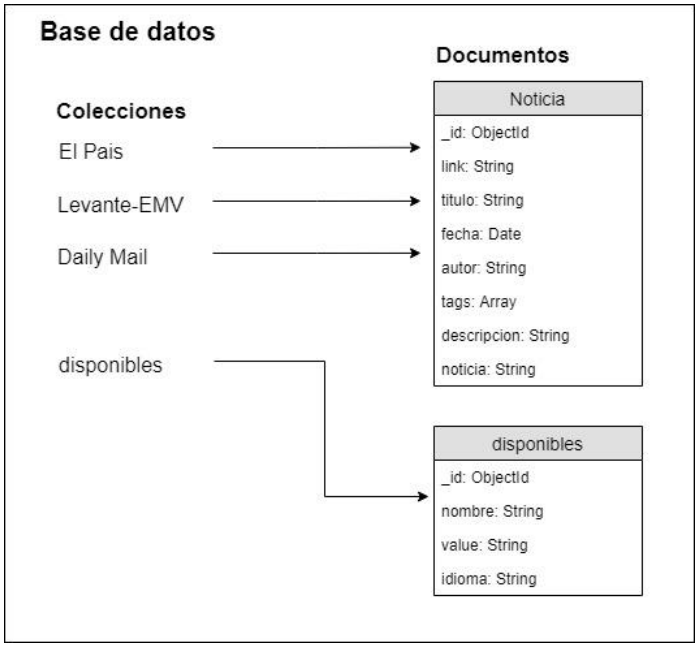

Figura 11. Diseño de la base de datos MongoDB

4.3.2.2. Capa lógica de negocio

En esta capa hemos diseñado y desarrollado todas las funciones de tratamiento de datos, así como la estructura de estos para su correcta representación en la capa de presentación.

En la **Figura 12** se ha diseñado un diagrama de secuencia donde se representa la interacción del usuario con la plataforma web. Inicialmente el usuario selecciona unos parámetros a través de la capa de presentación. Posteriormente entra en juego la capa de negocio donde a través de la capa de persistencia recibe todas las noticias filtradas por los parámetros insertados. Una vez obtenidas las noticias, la capa de negocio aplica varios análisis de datos para convertirlos en información útil y así poder representar y generar estadísticas que serán enviadas a la interfaz.

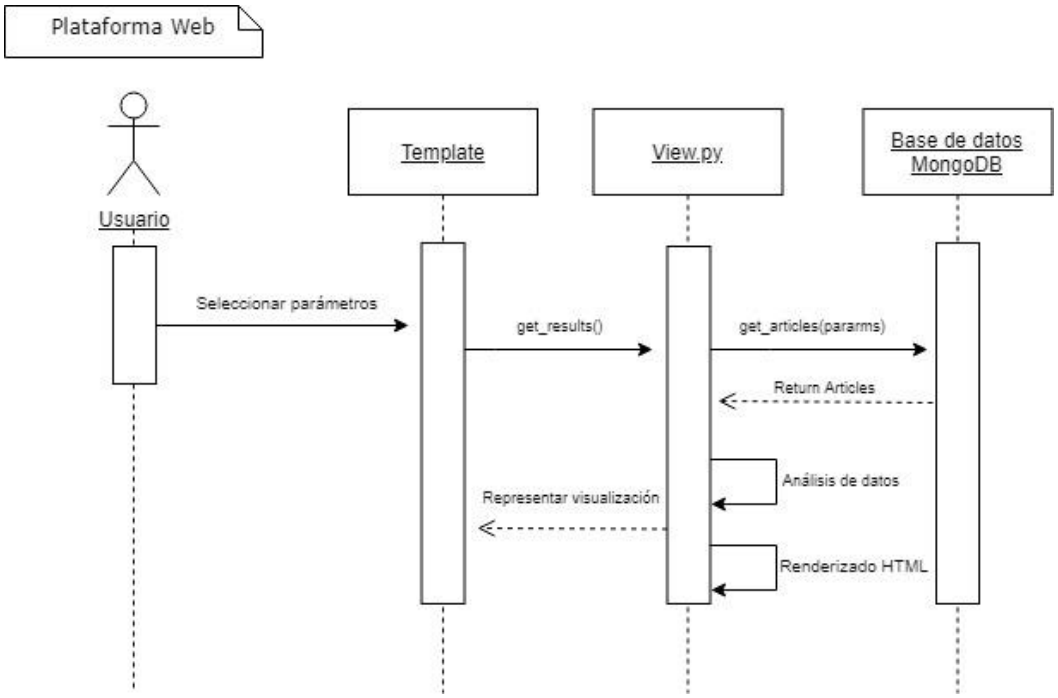

Figura 12. Diagrama de secuencia (CU03 y CU05)

4.3.2.3. Capa de presentación

En esta sección se mostrarán los esquemas de página (Wireframe). Son prototipos creados para dar una idea esquematizada de cómo va a ser la interfaz y su navegación entre páginas

Figura 13. Interfaz de filtros

En la **Figura 13** vemos el prototipo diseñado que representa la ventana principal. En ella hay un formulario con unos filtros que permiten acotar la búsqueda por periódico digital, categorías y rango de fechas. Una vez se pulse el botón “Seleccionar”, el sistema llevará a cabo unas acciones de filtrado donde posteriormente nos llevará a la interfaz de visionado de estadísticas (**Figura 14**).

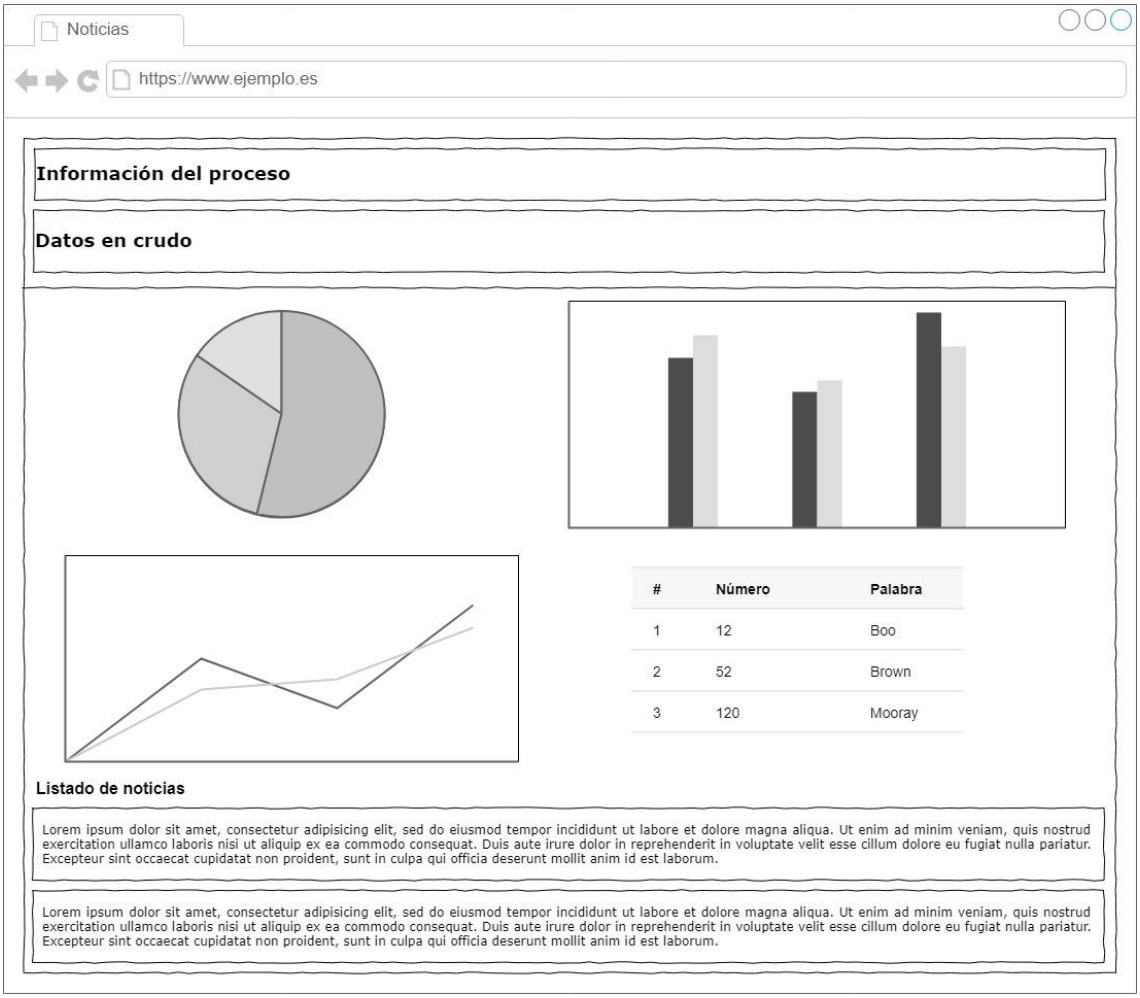

Figura 14. Interfaz visor de estadísticas

En la interfaz de estadísticas (**Figura 14**) el usuario podrá ver información del proceso de análisis de datos, datos en crudo, un conjunto de estadísticas y un listado de todas las noticias usadas en el análisis.

Información del proceso de datos

Se mostrará la información relativa al filtrado que previamente se ha seleccionado y el tiempo requerido del análisis de datos.

Datos en crudo

En este apartado se podrá observar el número de palabras que contienen el total de las noticias a procesar, así como el número de palabras que han quedado después del procesamiento de noticias. También se podrá ver el número de palabras que contienen todos los títulos de las noticias y el número total de noticias procesadas.

Estadísticas

- Nube de palabras (WordCloud) donde se mostrarán las palabras más comunes entre todas las noticias analizadas.

- Nube de palabras de Bigramas (WordCloud) donde se mostrarán el conjunto de dos palabras seguidas más comunes entre todas las noticias analizadas.

- Diagrama de barras que representa las cinco palabras más repetidas.

- Tabla donde se muestra la palabra más repetida y el bigrama más repetido y su número de repeticiones de cada día dentro del rango previamente seleccionado en la interfaz de filtros.

Listado de noticias

Se mostrará un listado de todas las noticias obtenidos después de realizar el filtrado en la interfaz de filtros.

5. Desarrollo de la solución propuesta

Como ya sabemos, nuestro sistema tiene dos subsistemas, parseadores de noticias y la plataforma web. En esta sección vamos a explicar el desarrollo e implementación de ambos subsistemas, así como las dificultades y decisiones que hemos tenido que tomar a la hora del desarrollo.

5.1. Parseadores de datos

El desarrollo de los parseadores de datos plantea dos problemas, como obtener las URL que contienen las noticias y como extraer los datos importantes del código HTML. Finalmente, se almacenan los datos de cada noticia en MongoDB.

5.1.1. Obtención y generación de URL

En el punto **4.2** hemos comentado el uso de diversas herramientas que nos permiten extraer datos de un portal web. La librería Request hace uso de URLs que posteriormente analiza y nos proporciona toda su estructura HTML. Estas URLs en nuestro caso son direcciones web donde se alojan las noticias, por lo tanto, necesitamos obtener de algún modo estas URLs.

Para obtener estas URLs hemos accedido a la hemeroteca de cada periódico utilizado en este proyecto. Esto nos ha permitido ver cuál es el patrón que sigue el formato de las URLs. Una vez obtenido el formato o formatos que usa cada periódico digital, se ha implementado una función para cada periódico que genera automáticamente estos enlaces.

El País

Este periódico presenta dos formateos de URL:

1- https://elpais.com/tag/fecha/[año][mes][dia]

2- https://elpais.com/hemeroteca/elpais/[año]/[mes]/[dia]/[m|t|n]

El primero de ellos es un formato utilizado hasta el 31 de diciembre del 2011 en el cual hay que indicar el año, mes y día y accederemos a un apartado donde están todas las noticias del día especificado.

El segundo es el formato actual implantado el 1 de enero del 2012. Al igual que el anterior formato, se debe especificar el año, mes y día añadiendo al final de esto una de las letras *m*, *t* o *n*, que representan la *mañana*, *tarde* o *noche* respectivamente. Del mismo modo accederemos a un listado de todas las noticias publicada en esa franja del día especificado.

Levante-EMV

El formato para este periódico es único y se forma de la siguiente forma:

- https://www.levante-emv.com/[año]/[mes]/[dia]

Cada una de las URLs generadas contiene un listado de todas las noticias del día especificado.

Daily Mail

Al igual que el Levante-EMV, el formato es único y nos ofrecerá un listado de todas las noticias del día especificado.

- http://www.dailymail.co.uk/home/sitemaparchive/day_[año][mes][dia]

5.1.2. Extracción de datos

Esta parte es la más laboriosa de los programas de extracción de datos. Implica revisar varios artículos de diferentes años para asegurarse de qué estructura HTML se está utilizando para generar la noticia.

El País

En este caso no podemos acceder a cada noticia de forma individual ya que la URL no la podemos generar automáticamente con un patrón por lo que debemos parsear el listado de noticias del que hemos hablado en el anterior punto, para obtener los enlaces de cada noticia.

Hay tres estructuras diferentes que se han ido utilizando a lo largo de los años en este periódico digital para acceder a cada noticia desde el listado.

1- Estructura 1: Desde el 4 de mayo del 1976 hasta el 31 de diciembre del 2011.

La etiqueta HTML que contiene cada una de las noticias en el listado es un <*div*> con la clase *articulo__interior*, y el título se extrae de la etiqueta <*a*> contiene la clase *articulo-titulo* de donde, además, se extrae el enlace de la noticia. Por otro lado, este listado contiene paginación por la que hay que navegar extrayendo el resto de las noticias. Esta paginación es una etiqueta <*li*> con la clase *paginacion-siguiente.*

2- Estructura 2: Desde el 1 de enero del 2012 hasta el 31 de diciembre del 2015.

Las noticias del listado están dentro de un <*div*> con la clase *article*. El título se extrae de la etiqueta <*a*> con el atributo *title="Ver noticia"*.

3- Estructura 3: Desde el 1 de enero del 2016 hasta la actualidad.

La noticia está contenida en un <*div*> con la clase *.articulo__interior*. El título y enlace se extrae de la etiqueta <*a*> con la clase *articulo-titulo*.

	Estructura 1	Estructura 2	Estructura 3
Noticia	 .articulo__interior	 .article	 .articulo__interior
Título y enlace	<a> .articulo-titulo	<a> title="Ver noticia"	<a> .articulo-titulo
Paginación	 paginacion-siguiente	No	No

Tabla 10. Resumen estructura HTML de El País

Una vez explicado cómo extraer el enlace de cada noticia y el título nos queda por obtener la fecha de publicación, el autor, las etiquetas asociadas a la noticia, la descripción y el cuerpo de la noticia, a continuación, mostramos como obtener cada dato.

Primero debemos obtener el código fuente del enlace de la noticia utilizando Request y BS4. A continuación, explicamos como se obtiene cada dato y la porción de código de BS4 que lo extrae.

- Fecha de publicación:

  Obtenemos la fecha a partir del contenido de un metadato.

  ```
  soupNews.select('meta[itemprop=datePublished]')[0].get('content')
  ```

- Autor

  Obtenemos el autor del artículo de un ** con la clase *.autor-nombre.*

  ```
  soupNews.select('.autor-nombre')
  ```

- Etiquetas

  Las etiquetas se obtienen de las palabras clave de la noticia que se encuentra en el contenido de un metadato.

  ```
  soupNews.select_one('meta[name=news_keywords]').get('content')
  ```

- Descripción

  La descripción se obtiene del contenido de un metadato.

  ```
  soupNews.select_one('meta[name="description"]').get('content')
  ```

- Cuerpo de la noticia

  El cuerpo de la noticia está contenido en una etiqueta * * con el atributo *itemprop=articleBody.* A su vez, se divide en párrafos con la etiqueta *<p>.*

  ```
  for p in soupNews.select('div[itemprop="articleBody"] > p'):

          bodyArticle = bodyArticle + p.get_text()
  ```

Levante-EMV

En este periódico digital debemos acceder a cada URL que hemos generado en el punto **5.1.1** para obtener un listado de noticias de un cierto día. Accediendo con BS4 a la etiqueta <*a*> con el atributo *data-tipo="noticia"* obtenemos el enlace que da acceso a una noticia en concreto.

A continuación, presentamos como acceder a cada dato de la noticia:

- Título

  Obtenemos el título a partir del contenido de un metadato.

  ```
  soupPage.select_one('meta[itemprop=name]').get('content')
  ```

- Fecha de publicación:

  Obtenemos la fecha a partir del contenido de un metadato.

  ```
  soupPage.select_one('meta[name=cXenseParse:recs:publishtime]').get('content')
  ```

- Autor

  Obtenemos el autor del artículo de un <*span*> con el atributo *itemprop=autor*.

  ```
  soupPage.select_one('span[itemprop=author]')
  ```

- Etiquetas

  Las etiquetas se obtienen de las palabras clave de la noticia que se encuentra en el contenido de un metadato.

  ```
  soupPage.select_one('meta[name=keywords]').get('content')
  ```

- Descripción

  La descripción se obtiene del contenido de un metadato.

  ```
  soupPage.select_one('meta[name=description]').get('content')
  ```

- Cuerpo de la noticia

  El cuerpo de la noticia está contenido en una etiqueta <*span*> con el atributo *itemprop=articleBody.* A su vez, se divide en párrafos con la etiqueta <*p*>.

  ```
  for p in soupPage.select('span[itemprop=articleBody] > p'):

          body = body + p.get_text()
  ```

Daily Mail

En este periódico digital debemos acceder a cada URL que hemos generado en el punto **5.1.1** para obtener un listado de noticias de un cierto día. Accediendo con BS4 a la etiqueta <*ul*> con la clase *archive-articles* obtenemos el enlace que da acceso a una noticia en concreto.

A continuación, presentamos como acceder a cada dato de la noticia:

- Título

  Obtenemos el título a partir del contenido de un metadato.

  ```
  soupPage.select_one('meta[property=og:title]').get('content')
  ```

- Fecha de publicación:

  Obtenemos la fecha a partir del contenido de un metadato.

  ```
  soupPage.select_one('meta[itemprop=dateModified]').get('content'
  ```

- Autor

  Obtenemos el autor a partir del contenido un metadato.

  ```
  soupPage.select_one('meta[itemprop=name]').get('content')
  ```

- Etiquetas

  Las etiquetas se obtienen de las palabras clave de la noticia que se encuentra en el contenido de un metadato.

  ```
  soupPage.select_one('meta[name=news_keywords]').get('content')
  ```

- Descripción

  La descripción se obtiene del contenido de un metadato.

  ```
  soupPage.select_one('meta[name=description]').get('content')
  ```

- Cuerpo de la noticia

  El cuerpo de la noticia está contenido en una etiqueta <*div*> con el atributo *itemprop=articleBody.* A su vez, se divide en párrafos con la etiqueta <*p*>*.*

  ```
  for p in soupPage.select('div[itemprop=articleBody] > p'):

          body = body + p.get_text()
  ```

5.1.3. Almacenamiento de datos extraídos

Una vez extraídos todos los datos de una noticia, debemos almacenarlos en MongoDB. Para ellos haremos uso de la librería PyMongo explicada en el punto **4.2.4**.

Conexión MongoDB

Lo primero que debemos hacer es establecer una conexión con la base de datos MongoDB y seleccionar la base de datos en la que se encuentra nuestra colección para guardar las noticias.

Para ello hacemos uso de la función *MongoClient* donde se especifica la dirección y el puerto de nuestro servidor de MongoDB.

```
client = MongoClient('localhost', 27017)
```

Una vez conectados, accederemos a la base de datos que queramos y seleccionaremos nuestra colección.

```
db = client.periodicos

coleccion = db[nombreColeccion]
```

Almacenamientos de noticias

Una vez realizada la conexión y adquirida la colección, el siguiente paso es almacenar los datos extraídos. La siguiente porción de código representa el uso de la función *save* proporcionada por PyMongo, donde estructurando los datos en formato JSON somos capaces de enviarlos a MongoDB.

```
coleccion.save(
        {
            'link': enlace,
            'titulo': titulo,
            'fecha': date,
            'autor': autor,
            'tags': listaEtiquetas,
            'descripcion': descripcion,
            'noticia': articulo
        })
```

5.2. Plataforma web

El desarrollo de la plataforma web en Django se enfoca al análisis y estadísticas de los datos extraídos por los parseadores, para así, convertirlos en información útil.

5.2.1. Introducción

El proyecto de Django está estructurado en los siguientes ficheros:

- url.py: Se definen todos los enlaces por los que el usuario podrá navegar.
- models.py: Es el fichero dónde se alojan todas las funciones relativas a la capa de persistencia, es decir, todas aquellas funciones que interactúan directamente con MongoDB.
- views.py: En este fichero están todas aquellas funcionan que pertenecen a la capa de negocio. Recibe y envía datos a la capa de presentación.
- template.html: Cada fichero con la extensión *.HTML* hace el papel de la capa de presentación. Es el diseño de la aplicación y quien recibe los datos que se van a mostrar.

5.2.2. Elección de parámetros de búsqueda

Una vez el usuario acceda a la web mediante un enlace previamente definido en el fichero url.py, deberá introducir unos parámetros de entrada que servirán para filtrar los datos. Estos parámetros son el periódico que se desea analizar, unas categorías opcionales y un rango de fecha. Cuando se pulse el botón seleccionar, la capa de presentación (template.html) enviará un evento a la capa de negocio (views.py) y ejecutará las siguientes funciones:

- getLenguageOfNewspaper(): Obtiene el idioma en el que están escritos las noticias del periódico.
- getArticlesRaw(): Obtiene todas las noticias de la base de datos aplicando los parámetros seleccionados en la interfaz de filtros.
- cleanArticle(): Limpia las noticias dejando solo la información útil del texto.
- getRawInfo(): Obtiene información en crudo, como el número de noticias a analizar, el número total de palabras o el número total de palabras una vez se hayan analizado las noticias.
- getWordCloud: Obtiene un WordCloud con las palabras más repetidas.

- getWordCloudBigrams: Obtiene un WordCloud con los Bigramas más repetidos.
- getGraphFrequency: Obtiene una gráfica de barras con las 5 palabras más repetidas.
- groupArticlesByDate: Agrupa las noticias por fecha.
- getMostFrequencyWordByDate: Obtiene la palabra más frecuente por fecha.
- getMostFrequencyBigramByDate: Obtiene el bigrama más frecuente por fecha.

Limpieza de texto de las noticias

Las funciones que obtienen las palabras o bigramas más repetidos necesitan que el texto esté previamente tratado. La necesidad de hacer esto viene dada por el hecho de que, en los textos, sean noticias o no, viene mucha información que no es útil como pueden ser los números, caracteres especiales o palabras muy comunes o sin significado en un idioma (Stop Words). Dado que la librería que utilizamos para el procesamiento de texto no implementa una gran cantidad de Stop Words en español, hemos añadido una lista más extensa a la propia proporcionada por la librería. De este modo hemos conseguido eliminar una gran cantidad de palabras innecesarias.

Los pasos que hemos realizado para la limpieza del texto son los siguientes:

1. Normalizar el texto a Unicode. De esta forma permitimos mantener palabras con letras propias de un idioma como puede ser la Ñ.
2. Convertir a minúsculas.
3. Eliminar acentos.
4. Eliminar números.
5. Eliminar símbolos especiales y signos de puntuación.
6. Eliminar palabras con una longitud inferior a tres.
7. Eliminar Stop Words (artículos, conjunciones, preposiciones, etc.).

Frecuencia de palabras y bigramas

Para obtener la frecuencia de palabras o bigramas hacemos uso de la librería NLTK. Esta librería nos ofrece una función llamada *FreqDist* que nos devuelve un diccionario donde la clave es una palabra y el valor es su número de apariciones en un texto dado. Además, para su uso en la gráfica de barras, se ha hecho uso de la función *most_common* para obtener solo las n palabras o bigramas más repetidos.

Este análisis de texto se ha realizado para hacer uso de él en la gráfica de barras, en los WordCloud y en las tablas de palabra o bigrama más frecuente de cada día.

6. Implantación

Una vez desarrolladas las aplicaciones procedemos a explicar cuáles son los pasos que realizar y que herramientas necesitamos para hacer uso de ellas en cualquier sistema. Dado que los parseadores de datos y la plataforma web trabajan en paralelo, vamos a dividir la implantación por aplicación.

Herramientas necesarias

- Instalar la última versión de Python3: https://www.python.org/downloads.
- Instalar el motor de base de datos MongoDB siguiendo los pasos indicados en el siguiente enlace: https://docs.mongodb.com/manual/installation.
- Iniciar el servicio de MongoDB descrito en el enlace del manual del paso anterior.

Herramientas y paquetes necesarios para los parseadores de datos

Para ejecutar los parseadores de datos es necesario instalar los siguientes paquetes/librerías de Python:

- BeautifulSoup4: Paquete de Python que permite parsear documentos HTML (https://pypi.org/project/beautifulsoup4)
- lxml: Parseador usado por BeautifulSoup4 (https://pypi.org/project/lxm)
- PyMongo: Librería para interactuar con la base de datos MongoDB (https://pypi.org/project/pymongo)

Una vez instalados, debemos ejecutar el parseador de datos que queramos con el siguiente comando en un terminal o consola:

```
python .\parser.py fechaInicio fechaFin nombreColeccionMongoDB
```

Herramientas y paquetes necesarios para la plataforma web en Django

1. Para el funcionamiento de la web realizada en Django debemos instalar los paquetes indicados en el **Apéndice A** con el siguiente comando:

```
pip install <nombreDelPaquete>
```

2. Iniciar Python en el terminal/consola y escribir los siguientes comandos para descargar los StopWords de la librería nltk:

```
import nltk

nltk.download('stopwords')
```

3. Añadir la colección “disponibles” en MongoDB descrita en el punto **4.3.2.1**

4. Agregar en cada colección de nuestra base de datos en MongoDB un *"index"* sobre el campo "fecha".

   Este índice se crea para poder realizar consultas que contenga una gran cantidad de resultados a partir de cierto campo de la colección. De lo contrario, si no se crea, habrá un exceso de memoria utilizada y MongoDB no será capaz de realizar la consulta.

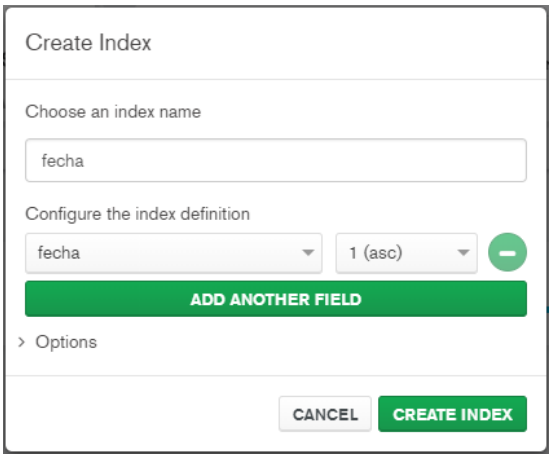

Figura 15. Añadir index en el campo fecha de nuestras colecciones

5. Iniciar el servidor Django con el siguiente comando:

```
python manage.py runserver
```

6. Acceder a la plataforma web desarrollada a través del siguiente enlace: http://127.0.0.1:8000/newspapers

Acceso a las herramientas software

Tanto los parseadores de cada periódico como la plataforma web se encuentra disponibles en los siguientes enlaces de GitHub:

- Parseadores de datos: https://github.com/jagarro3/Parser_Noticias
- Plataforma web en Django: https://github.com/jagarro3/webParser

Como hemos explicado en el punto X, no hemos podido dar acceso libre al corpus por motivos de derechos de uso.

7. Pruebas

En este apartado se van a analizar casos de prueba para comprobar el correcto funcionamiento de los casos de usos descritos en la **Tabla 5**. Además, se explicará una prueba que se ha realizado durante el desarrollo para ver que el comportamiento de la función que se encarga de la limpieza de texto realmente es satisfactoria.

7.1. Casos de prueba

Caso de prueba 1

Caso de uso: CU01 - Extraer datos.

Caso de prueba: Ejecutar el script de cada parseador de datos sin parámetros de entrada.

Resultado: Mensaje de advertencia indicando que faltan parámetros de entrada como la fecha o la colección de MongoDB.

```
Introduce un rango de fecha Dia-Mes-Año y el nombre de la coleccion MongoDB

-> Ejemplo: python .\parser.py [02-06-1976] [12-11-1986] nombreColeccion

-> Nota: Año mínimo 1976
```

Caso de prueba 2

Caso de uso: CU01 - Extraer datos, CU02 - Filtrar la extracción de datos.

Caso de prueba: Ejecutar el script de cada parseador de datos con parámetros de entrada con un rango de fechas mayor al día actual, fecha incorrecta o una fecha anterior a la primera noticia publicada en el periódico digital.

Resultado: Mensaje de advertencia indicando que el rango de fechas es incorrecto.

```
-----------------------------------------------------

Posibles causas:

-> La primera fecha tiene que ser menor que la segunda

-> El año de inicio no puede ser menor que el 1-1-1976
```

Caso de prueba 3

Caso de uso: CU01 - Extraer datos, CU02 - Filtrar la extracción de datos.

Caso de prueba: Ejecutar el script de cada parseador de datos con parámetros de entrada correctos.

Resultado: La ejecución del programa inicia correctamente y empieza a insertar noticias en MongoDB.

Caso de prueba 4

Caso de uso: CU03 - Consultar información útil de los datos, CU05 - Aplicar filtros para afinar los resultados.

Caso de prueba: Búsqueda con filtro y sin filtros a través del formulario principal de la web Django.

Resultado: Se obtienen las noticias según el rango de fecha, palabras o categorías seleccionados.

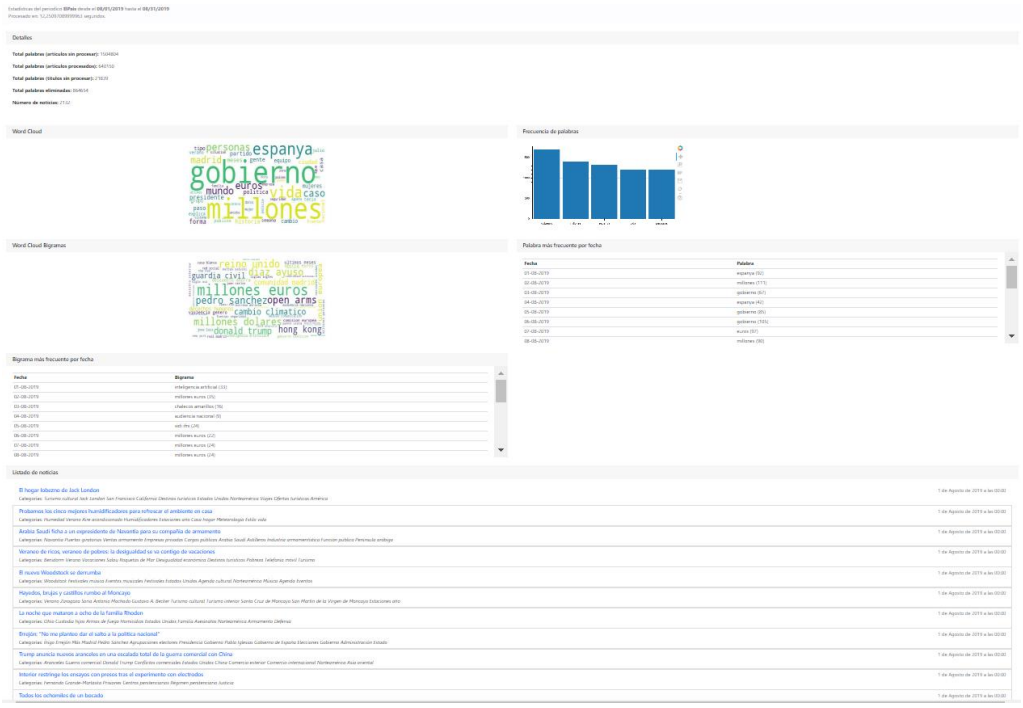

Figura 16. Vista de los resultados obtenidos al aplicar los filtros.

Caso de prueba 5

Caso de uso: CU04 - Descargar estadísticas en formato imagen

Caso de prueba: Interactuar con el control de mandos de las gráficas

Resultado: Se obtiene una imagen en formato png de la gráfica de barras.

7.2. Prueba limpieza de texto

Como se ha explicado en el punto **5.2.2**, el texto de las noticias se tiene que procesar para obtener un texto limpio y así poder hacer un análisis correcto del mismo.

Para comprobar que la limpieza de texto ha sido satisfactoria, hemos cogido diversas noticias al azar y hemos comparado el cuerpo de la noticia sin procesar con el cuerpo de la noticia una vez procesado. A modo de ejemplo, vamos a representar a continuación una porción de texto de una noticia para poder ver de una forma más clara que transformación ha recibido el texto.

las llamas que devoran estos dias la amazonia brasileña son las mismas que alimentan en las redes sociales la indignacion contra el presidente brasileño, jair bolsonaro. mientras la etiqueta #prayforamazon (reza por la amazonia) se convertia el miercoles en tendencia global y los habitantes de las regiones afectadas publicaban imagenes de los daños causados por el fuego, el presidente decia, sin aportar pruebas, que eran miembros de organizaciones de defensa del medioambiente quienes estaban causando las llamas deliberadamente, en venganza por el recorte de fondos decretado por el gobierno.los incendios en la mayor selva del mundo han trascendido de la esfera meramente brasileña: el presidente frances califico los fuegos en la amazonia como "crisis internacional" en su cuenta de twitter. emmanuel macron opto por informar de que incluiria el tema en la agenda del g7 de este fin de semana. el mensaje del mandatario frances en la red social estaba acompañado de una foto, que segun el diario folha desao paulo, no corresponde a los incendios actuales y es en realidad una instantanea tomada en 1989 por la fotoperiodista loryn mciyntire.el secretario general de la onu, antonio guterres, tambien ha mostrado su "profunda preocupacion" por la situacion. "en medio de la crisis climatica mundial, no podemos permitirnos mas daño a una gran fuente de oxigeno y biodiversidad. la amazonia debe ser protegida", ha tuiteado el maximo responsable de naciones unidas.pero bolsonaro, lejos de recular, este jueves ha vuelto a la carga. ha cargado contra macron por su decision de incluir el asunto en la agenda del g7 y ha acusado al mandatario frances de querer "instrumentalizar" una cuestion interna brasileña para "obtener reditos politicos personales", al tiempo que señalaba a los paises que dan dinero para la preservacion de la selva de "interferir en la soberania de brasil".

Figura 17. Porción de una noticia del periódico El País (https://elpais.com/internacional/2019/08/22/actualidad/1566501636_486466.html)

En la **Figura 17** vemos el resultado obtenido una vez procesado el texto. Para poder comparar que palabras o caracteres han permanecido después de la limpieza de texto, hemos utilizado un comparador de texto, comparando la noticia original con las palabras resultantes del procesamiento.

Esta porción de noticia se representa tal cual está publicada, pero con pequeños matices. Hemos quitado los acentos y convertido el texto a minúsculas, de este modo hemos facilitado al comparador la tarea de comprobar que palabras permanecen o no.

El texto con el fondo en blanco representa las palabras que han permanecido después de procesar el texto, por otro lado, en color turquesa, vemos las palabras o caracteres que han sido eliminados ya sea porque son símbolos especiales, Stop Words.

Si leemos las palabras que han permanecido, la noticia aún tiene sentido, por lo tanto, vemos que los caracteres, números o palabras eliminadas no aportan nada a la noticia, aportándonos así una limpieza de texto satisfactoria.

8. Caso de estudio

Una vez acabado el proyecto, se propone hacer uso de ambas herramientas. Primero se hará uso de los tres parseadores de datos que se han desarrollado y posteriormente se estudiarán varios casos reales a través de la plataforma web.

8.1. Características del equipo utilizado

Una característica fundamental del proyecto es optimizar la ejecución de procesamiento de los parseadores y de la obtención de datos, así como de sus análisis en la plataforma web.

El tiempo de computo depende de la conexión de red y del equipo además de otros factores como el propio código de los programas. Durante el desarrollo, las aplicaciones se han ejecutado en diferentes equipos viéndose afectado el rendimiento y rapidez de los programas desarrollados. Por ello, viendo la variación de los tiempos obtenidos en los diversos equipos, mostraremos a continuación las características del equipo usado para realizar el caso de uso de los puntos **8.2** y **8.3**.

Características del equipo

- Velocidad conexión de red: 94.77 Mbps
- Procesador: i5-8600k @3.60 (6 núcleos)
- Memoria RAM: 16GB - 2400 MHz
- Disco duro: SSD - 560 MB / s de lectura, 510 MB / s de escritura

8.2. Parseadores de datos

Lo primero que vamos a ejecutar son los tres parseadores de datos. Para poder realizar este pequeño caso de uso vamos a extraer noticias en los siguientes rangos de fecha:

- **Rango 1**: 1 de enero del 2019 hasta el 25 de agosto del 2019.
- **Rango 2**: 1 de julio del 2018 hasta el 31 de julio del 2018.
- **Rango 3**: 1 de marzo del 2011 hasta el 31 de marzo del 2011.

De este modo podremos demostrar el tiempo de computo que tarda en extraer un año entero, así como un solo mes. Además, compararemos los resultados del mes de

julio en 2018 y 2019. Como se ha explicado en puntos anteriores, la estructura HTML varía entre años, además de la cantidad publicada de noticias. Por ello, adicionalmente ejecutaremos los programas con una fecha muy posterior a la actual (1 de septiembre del 2011 hasta el 31 de septiembre del 2011).

Todos los tiempos en minutos o segundos están obtenidos teniendo en cuenta el tiempo de computo de la descarga de la noticia, la extracción de datos y el guardado en MongoDB.

1 de enero del 2019 hasta el 25 de agosto del 2019 (8 meses)				
Periódico digital	**Cantidad de noticias obtenidas**	**Tiempo requerido (minutos)**	**Tiempo en segundos por noticia**	**Número de palabras (cuerpo de la noticia)**
El País	26415	30,12 min	0,068 s	18.847.482
Levante-EMV	11266	10,29 min	0,054 s	3.738.772
Daily Mail	352538	601,93 min	0,102 s	159.258.002

Tabla 11. Resultado de la ejecución de los parseadores entre el 1 enero 2019 hasta el 25 agosto 2019

1 de julio del 2018 hasta el 31 de julio del 2018 (1 mes)				
Periódico digital	**Cantidad de noticias obtenidas**	**Tiempo requerido (minutos)**	**Tiempo en segundos por noticia**	**Número de palabras (cuerpo de la noticia)**
El País	3164	3,46 min	0,065 s	1.863.880
Levante-EMV	1446	1,28 min	0,053 s	493.093
Daily Mail	52074	79,76 min	0,092 s	22.488.103

Tabla 12. Resultado de la ejecución de los parseadores entre el 1 julio 2018 hasta el 31 julio 2018

1 de marzo del 2011 hasta el 31 de marzo del 2011 (1 mes)				
Periódico digital	**Cantidad de noticias obtenidas**	**Tiempo requerido (minutos)**	**Tiempo en segundos por noticia**	**Número de palabras (cuerpo de la noticia)**
El País	10272	8,2 min	0,047 s	5.034.112
Levante-EMV	1052	0,60 min	0,034 s	509.811
Daily Mail	9464	13,02 min	0,086 s	4.761.692

Tabla 13. Resultado de la ejecución de los parseadores entre el 1 marzo 2011 hasta el 31 marzo 2011

Como vemos en la **Tabla 11**, **Tabla 12** y **Tabla 13**, los resultados en cuanto a cantidad de noticias son bastante dispares. Los periódicos El País y Levante-EMV presentan una cantidad de noticias publicadas bastante pequeña respecto al diario Daily Mail. El principal motivo de esta diferencia tan enorme de noticias publicadas se debe a que la sección Hemeroteca/Archivo de los periódicos El País y Levante-EMV no representa la totalidad de noticias publicadas en una cierta fecha además de no cambiar la estructura HTML respecto a la portada principal del periódico, sin embargo, la hemeroteca del Daily Mail sí contiene todas las noticias publicadas en una cierta fecha en un listado de artículos perfectamente estructurados. Otro motivo de la pequeña cantidad de noticias publicadas por el periódico Levante-EMV es que el acceso a ciertas noticias requiere una suscrición de pago para poder ser leídas, lo que hace imposible la extracción de estas noticias. Por otro lado, El País alberga una mayor cantidad de

noticias en fechas anteriores al año 2012, la principal diferencia que se observa es la estructura HTML siendo más parecida a un listado de noticias como lo hace Daily Mail.

Si observamos los tiempos de computo, vemos que los periódicos El País y Levante-EMV obtienen un tiempo por noticia bastante parecido. Sin embargo, DailyMail presenta un tiempo de cómputo por articulo bastante elevado respecto al resto de periódicos.

Viendo esta anomalía, hemos estudiado los tiempos de descarga de los artículos del DailyMail. Hemos podido observar que la media de descarga de cada noticia es de 0.5 segundos incluso llegando alguna a costar más de 1 segundo en descargarla. Para comprobar porque sucedía esto, hemos mostrado el tamaño recibido en bytes de cada artículo en los parseadores con la siguiente línea de código:

```
r = requests.get(enlaceArticulo)

print("Tamaño en bytes:", len(r.content))
```

El tamaño en bytes obtenido de los artículos del DailyMail comprendían entre 400.000 bytes y 550.000 bytes, por otro lado, los del El País entre 100.000 bytes y 140.000 bytes y por último los del Levante-EMV entre 100.000 bytes y 150.000 bytes. Esto explica porque el tiempo de computo de los artículos del DailyMail duplica al del resto de periódicos.

8.3. Plataforma web Django

A modo de ejemplo, vamos a realizar diversos casos de estudio conocidos por la mayoría de las personas como puede ser un acontecimiento popular, una catástrofe natural o problemas que nos afectan a todos y son conocido socialmente.

Temperatura en Valencia

Si hacemos una búsqueda por el término “tiempo” o “weather” en el periodo comprendido desde el 5 de agosto del 2019 hasta el 15 de agosto del 2019 en cada uno de los periódicos disponibles obtendremos los siguientes resultados y tiempos de computación.

Tiempo de cómputo y resultados realizando la siguiente búsqueda: Término “tiempo” desde el 5 de agosto del 2019 hasta el 15 de agosto del 2019				
Periódico digital	**Cantidad de noticias obtenidas**	**Tiempo requerido (segundos)**	**Palabras en total (antes del procesado)**	**Palabras eliminadas después del procesado**
El País	415	3,67	405.138	234.725
Levante-EMV	78	0,88	44.726	25.993
Daily Mail	555	3,16	334.206	176.609

Tabla 14. Resultados obtenidos a partir de la búsqueda: Término “tiempo” desde el 5 de agosto del 2019 hasta el 15 de agosto del 2019

Para analizar el tiempo en Valencia haremos uso del periódico Levante-EMV ya que es un diario a nivel de la Comunitat Valenciana. Si observamos la **Figura 18** donde se muestra una gráfica del histórico de temperatura desde junio hasta agosto en Valencia, vemos que hay un pico significativo que supera los 40ºC entre el 6 de agosto y 12 de agosto. Si buscamos una relación con los resultados obtenidos en la plataforma web vemos en la **Figura 19** y **Figura 20** como las palabras y bigramas más comunes relacionadas con la meteorología son “calor, altas temperaturas, ola calor, golpe calor”, además, empiezan a aparecer a partir del 7 de agosto y dejan de ser relevantes a partir de 10 de agosto, coincidiendo así con el pico de temperatura que comentábamos.

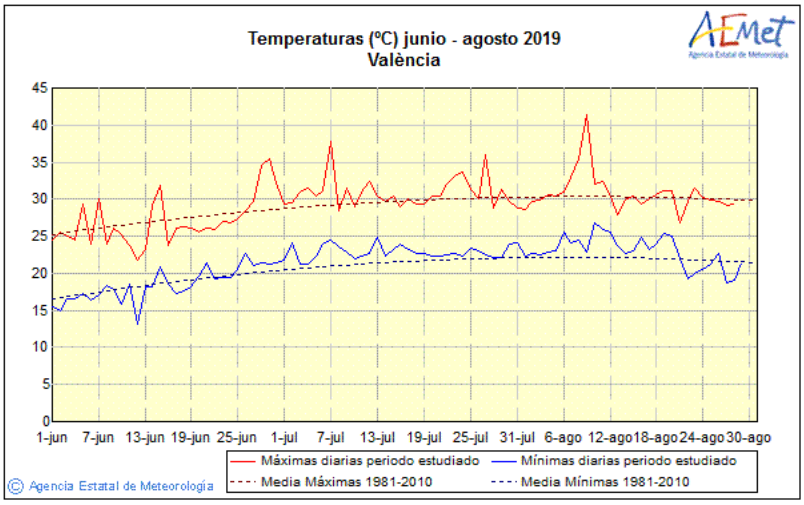

Figura 18. Gráfica de la temperatura (ºC) junio - agosto 2019 en Valencia **(23)**

Fecha	Bigrama
05-08-2019	estrena agosto (8), agosto netflix (6), amazon prime (5), acaip ugt (4), tipo internos (4),
06-08-2019	mejores vistas (2), vistas mundo (2), calles estrechas (2), cabo verde (2), vistas bellas (2),
07-08-2019	ola calor (3), impacto ambiental (2), autoridad portuaria (2), busqueda formulas (2), formulas respetuosas (2),
08-08-2019	baño libre (8), libre lunes (8), comunitat valenciana (7), junio septiembre (7), septiembre horario (7),
09-08-2019	comunitat valenciana (5), altas temperaturas (4), diego ibañez (4), temperaturas superar (3), superar grados (3),
10-08-2019	corrupcion politica (6), cargos publicos (6), golpe calor (4), mujeres cargos (4), campo concentracion (4),
11-08-2019	meterlos lavavajillas (4), platos meterlos (3), maxi gomez (3), mañana lunes (3), lateral izquierdo (2),
12-08-2019	lluvia meteoros (6), ocean viking (4), perseidas lluvia (3), experimentar sensaciones (2), sensaciones unicas (2),
13-08-2019	vicent pastor (7), coca loca (5), paco comes (4), provincia valencia (4), alojarse hotel (4),

Figura 19. Bigramas más frecuentes con el término "tiempo" entre el 5 agosto 2019 y el 15 agosto 2019

Fecha	Palabra
05-08-2019	agosto (17), vida (11), estrena (10), tipo (8), movistar (8),
06-08-2019	vistas (11), pueblo (11), pies (9), calles (8), mundo (7),
07-08-2019	puerto (8), ribo (7), calor (7), españa (6), frente (4),
08-08-2019	valencia (20), viernes (12), plantilla (12), libre (11), horario (11),
09-08-2019	valencia (18), grados (11), temperaturas (10), zonas (10), vida (10),
10-08-2019	riesgo (12), calor (11), valencia (9), temperaturas (8), cargos (8),
11-08-2019	valencia (14), mestalla (11), marcelino (10), soler (9), equipo (8),
12-08-2019	meteoros (13), agosto (9), lluvia (9), perseidas (8), personas (7),
13-08-2019	valencia (16), partido (11), pastor (11), musica (10), vicent (9),

Figura 20. Palabras más frecuentes con el término "tiempo" entre el 5 agosto 2019 y el 15 agosto 2019

Terremoto y tsunami de Japón de 2011

El terremoto y tsunami ocurrido en Japón el 11 de marzo de 2011 fue un acontecimiento de importancia mundial. Haremos la búsqueda en los periódicos El País y Daily Mail con el término “desastre” y “disaster” respectivamente en el periodo comprendido entre el 10 de marzo del 2019 hasta el 17 de marzo del 2019. No haremos la búsqueda en el periódico Levante-EMV por su ámbito regional y su corpus tan pequeño respecto a los otros dos periódicos. Además, realizando unas pruebas previas hemos visto que, dado que es un periódico enfocado a la Comunitat Valenciana, los términos y noticias más relevantes en marzo del 2011 son “Las Fallas”.

En los resultados reflejados en la **Figura 21** y **Figura 22** vemos como buscando el término “desastre” o “disaster” encuentra las noticias relacionada con este acontecimiento dónde las palabras y bigramas más frecuentes son “nuclear”, “japón”, “centrales”, “terremoto”, “nuclear power” o “earthquake tsunami”, además un dato curioso es que el propio termino que hemos buscado no aparece entre los más comunes.

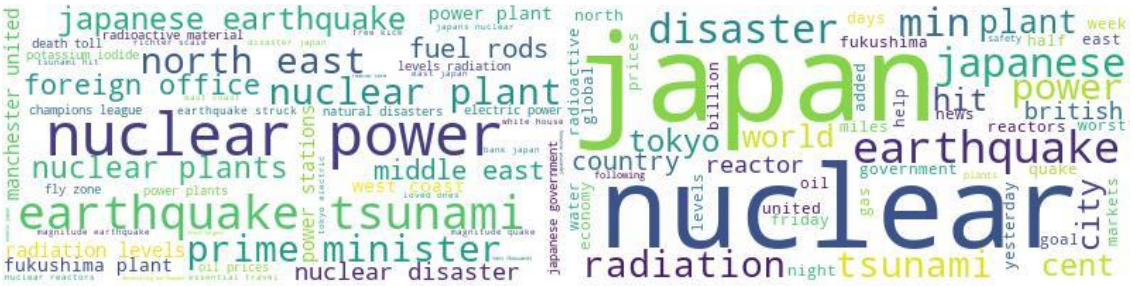

Figura 21. Resultados obtenidos con el término "disaster" en el periódico Daily Mail

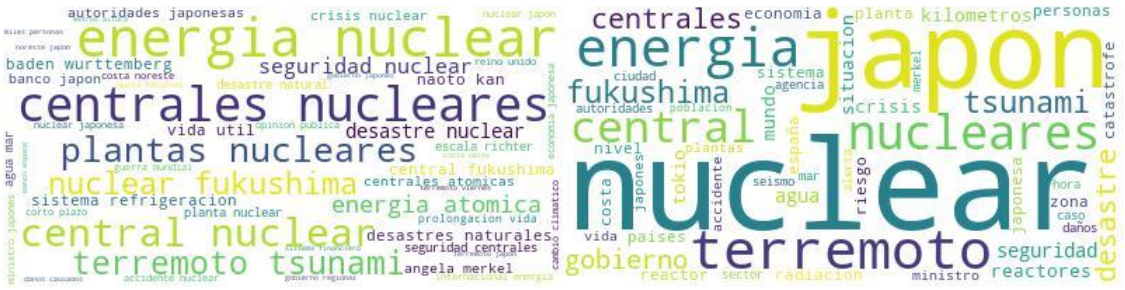

Figura 22. Resultados obtenidos con el término "desastre" en el periódico El País

¿Qué ha sucedido en agosto del 2019?

Se han obtenido las estadísticas dentro del periodo comprendido entre el 1 de agosto de 2019 y el 31 de agosto del 2019.

Resultados obtenidos				
Periódico digital	**Cantidad de noticias obtenidas**	**Tiempo requerido (segundos)**	**Palabras en total (antes del procesado)**	**Palabras eliminadas después del procesado**
El País	2840	16,09	2.034.189	1.161.301
Levante-EMV	1486	4,03	465.364	265.482
Daily Mail	50074	153,07	22.809.182	11.984.889

Tabla 15. Tiempo de cómputo y cantidad de noticias y palabras obtenidos desde el 1 de agosto del 2019 hasta el 31 de agosto del 2019

8 palabras y bigramas más frecuentes		
Periódico digital	**Palabras**	**Bigramas**
El País	gobierno, mundo, España, vida, personas, caso, Madrid, presidente	EEUU, Pedro Sánchez, Donald Trump, reino unido, Díaz Ayuso, Open Arms, cambio climático, Hong Kong
Levante-EMV	Valencia, personas, gobierno, policía, valenciana, caso, agosto, España	Policía local, policía nacional, guardia civil, Valencia CF, Comunitat valenciana, carne mechada, Open Arms, Juan Carlos
Daily Mail	police, Trump, government, week, million, president, world, season	prime minister, Donald Trump, social media, Hong Kong, premier league, president Donald, el Paso, los Ángeles

Tabla 16. Resultados obtenidos desde el 1 de agosto del 2019 hasta el 31 de agosto del 2019

En la **Tabla 16** vemos las palabras y bigramas más frecuentes. Podemos apreciar que en los tres periódicos hay términos en común relacionados con la política como pueden ser “gobierno, Donald Trump,” y con temas mundiales como pueden ser “Open Arms” o “cambio climático”.

Además, si vemos el listado de las palabras y bigramas más frecuentes en cada periódico, hay periodos de tiempo donde hay un tema más popular respecto al resto como ha sido el brote de listeriosis por la carne mechada en Andalucía (brote, carne mechada, Magrudis) (24), o el incendio en el Amazonas (Bolsonaro, incendio, Amazonas, Brasil) (25).

9. Conclusiones

En este proyecto se han desarrollado dos herramientas, los parseadores de noticias y la plataforma web para el análisis de estas.

En el apartado **1.2** se citaron los objetivos a cumplir en el proyecto, se han cumplido cada uno de ellos. El primero objetivo fue adquirir datos de diferentes plataformas de información. Podrían ser foros, periódicos digitales, Wikipedia etc. Vimos que uno de los foros más importante era StackOverflow, pero tuvimos problemas de conexión con los servidores al extraer información ya que nos limitaban el número de conexiones. El desarrollo de extraer información de la Wikipedia resultaba sencillo ya que Wikimedia proporciona el corpus de toda la Wikipedia por lo que el trabajo era mínimo. De donde podíamos sacar más datos era de los periódicos digitales, dado que hay una cantidad enorme de ellos y en diferentes idiomas. Al principio hicimos uso de una librería Python llamada Newspaper2k, la cual automatizaba la extracción de noticia simplemente proporcionándole el enlace del periódico digital que queríamos parsear. Resultó no ser satisfactorio ya que carecía de personalización y no extraía todos los artículos del periódico, además de carecer de parámetros como sería un rango de fechas. Por lo que hicimos nuestra propia implementación utilizando librerías como BS4 y Request.

El segundo objetivo era el almacenamiento de los datos, no tuvimos ningún problema, y estuvo claro desde el principio que tenía que ser una base de datos donde se almacenaran en formato JSON. En cuanto al objetivo de limpieza de datos, hicimos uso de Stopwords y expresiones regulares, añadiendo nosotros mismos algunas Stopwords más que no aportaban nada al estudio.

La plataforma web en Django surgió más tarde. Era una oportunidad de darle sentido a los datos extraído sin requerir semanas en tiempo de computación. A partir de esto pudimos cumplir el último objetivo nacido a partir de la plataforma web, casos de estudio, de esta forma hemos analizado los datos y visto de una forma interactiva y curiosa la utilidad del proyecto.

Las dificultades que ha presentado el desarrollo han sido el análisis de la estructura de cada periódico, dado que era un tiempo considerable analizar ciertos rangos de fechas para comprobar que la estructura HTML de un artículo del periódico seguía siendo la misma. Aun así, los parseadores han recibido muchos cambios, adaptándolos a los cambios estructurales de los artículos. El desarrollo en Django me resulto difícil hasta que me adapte después de realizar diversos tutoriales y ver varios videos de ayuda. Una vez cogí soltura todo parecía más sencillo y requería de menos cambios de código cuando quería modificar alguna parte.

En cuanto a las tecnologías utilizadas las únicas dificultades han sido como he dicho Django y BS4 que tiene un extenso manual de uso dada su amplia funcionalidad. Durante el curso he realizado diversas asignaturas en la rama de Computación que me han proporcionado los conocimientos para desarrollar en Python y tener una base de la adquisición de datos y su tratamiento y análisis.

10. Trabajos futuros

Para concluir se van a presentar diversas mejoras en las herramientas.

Podría hacerse de forma automática la ejecución de los parseadores de datos además de hacer una trazabilidad de los días que ya han sido parseados sin tener que ir a la base de datos a consultar por qué fecha se paró la última ejecución. La plataforma web tiene varias funcionalidades que añadir. Una funcionalidad útil para el análisis de datos sería dividir estos por categoría geográfica. En cuanto a tiempo de computación, hemos visto que los tiempos adquiridos son buenos, pero se puede hacer un análisis de coste para ver donde mejorar el tiempo.

El uso del corpus se puede utilizar para aprender word embeddings (**Tabla 4**) u otras aplicaciones dentro del área del procesamiento del lenguaje natural, por lo que tendría un uso más interesante.

Bibliografía

1. **Swersky, Dave.** Raygun. [En línea] 31 de Mayo de 2018. [Citado el: 24 de Marzo de 2019.] https://raygun.com/blog/software-development-life-cycle.

2. **Wirth, Rüdiger y Hipp, Jochen.** CRISP-DM: Towards a Standard Process Model for Data. [En línea] 2000. [Citado el: 28 de Marzo de 2019.] https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.198.5133&rep=rep1&type=pdf.

3. **AIMC.** AIMC - ASOCIACIÓN PARA LA INVESTIGACIÓN DE MEDIOS DE COMUNICACIÓN. [En línea] [Citado el: 3 de septiembre de 2019.] https://www.aimc.es/a1mc-cont3nt/uploads/2019/01/marco19.pdf.

4. **REAL ACADEMIA ESPAÑOLA.** Corpus del Español del Siglo XXI (CORPES XXI). [En línea] [Citado el: 31 de agosto de 2019.] https://www.rae.es/recursos/banco-de-datos/corpes-xxi.

5. —. **Corpus de Referencia del Español Actual (CREA).** [En línea] [Citado el: 31 de agosto de 2019.] https://www.rae.es/recursos/banco-de-datos/crea.

6. —. **Nuevo diccionario histórico del español (CDH).** [En línea] [Citado el: 31 de agosto de 2019.] https://www.rae.es/recursos/banco-de-datos/cdh.

7. —. **Corpus Diacrónico del Español (CORDE).** [En línea] [Citado el: 31 de agosto de 2019.] https://www.rae.es/recursos/banco-de-datos/corde.

8. **BBVA, Fundéu.** PROYECTO ARACNE. [En línea] [Citado el: 31 de agosto de 2019.] http://www.fundeu.es/aracne/.

9. **corpusdelespanol.** Corpus del Español. [En línea] [Citado el: 31 de agosto de 2019.] https://www.corpusdelespanol.org.

10. **Sketchengine.** Sketchengine: Timestamped JSI web corpus. [En línea] [Citado el: 31 de agosto de 2019.] https://www.sketchengine.eu/jozef-stefan-institute-newsfeed-corpus/.

11. **Hemero.** hemero. [En línea] [Citado el: 31 de agosto de 2019.] http://www.hemero.es.

12. **Linguistic Data Consortium.** [En línea] https://www.ldc.upenn.edu.

13. EGM. Ranking tipología de soportes de internet. [En línea] Asociación de la Investigación de medios de la Comunicación, 1 de 7 de 2019. [Citado el: 2019 de 7 de 4.] http://reporting.aimc.es/index.html#/main/internet.

14. Trends, Google. Periódicos digitales valencianos más buscados en Google. [En línea] Julio de 2019. https://trends.google.com/trends/explore?geo=ES-VC&q=%2Fg%2F12195wvj,%2Fm%2F012hr570,%2Fm%2F07p0y4.

15. Comscore. Ranking digital media. [En línea] Mayo de 2019. https://www.comscore.com/Insights/Rankings.

16. cultura, Ministerio de. Ley de Propiedad Intelectual. [En línea] [Citado el: 05 de septiembre de 2019.] https://www.boe.es/eli/es/rdlg/1996/04/12/1/con.

17. País, El. Condiciones generales. [En línea] [Citado el: 05 de septiembre de 2019.] https://elpais.com/estaticos/aviso-legal/index.html.

18. Levante-EMV. Condiciones Generales de Uso. [En línea] [Citado el: 05 de septiembre de 2019.] https://micuenta.levante-emv.com/protecciondatos?gdprTipo=3.

19. DailyMail. Terms. [En línea] [Citado el: 05 de septiembre de 2019.] https://www.dailymail.co.uk/home/article-1388146/Terms.html.

20. Stackoverflow. Developer Survey Results - Web Frameworks. [En línea] 2019. https://insights.stackoverflow.com/survey/2019#technology-_-web-frameworks.

21. Brown, Ryan. Django vs Flask vs Pyramid: Choosing a Python Web Framework. [En línea] [Citado el: 21 de 5 de 2019.] https://www.airpair.com/python/posts/django-flask-pyramid.

22. Uniwebsidad. El patrón de diseño MTV. [En línea] [Citado el: 2019 de 22 de 20.] https://uniwebsidad.com/libros/django-1-0/capitulo-5/el-patron-de-diseno-mtv.

23. AEMET. AEMET.ES. [En línea] http://www.aemet.es/es/serviciosclimaticos/vigilancia_clima/analisis_estacional?w=4&l=8416&datos=temp.

24. LÓPEZ, CRISTIAN. El País. [En línea] 16 de agosto de 2019. https://elpais.com/sociedad/2019/08/15/actualidad/1565895747_480784.html.

25. Ibbetson, Ross. Daily Mail. [En línea] 21 de agosto de 2019. https://www.dailymail.co.uk/news/article-7379579/Wild-blazes-destroying-swathes-Amazon-rainforest-seen-satellite-images.html.

Apéndice A

Paquetes que instalar para Django

```
Package         Version
--------------- -------
Django          2.2.4
nltk            3.4.5
pymongo         3.9.0
matplotlib      3.1.1
bokeh           1.3.4
wordcloud       1.5.0
Unidecode       1.1.1
```