UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Diseño de un programa en Python para el análisis de parámetros SEO que afectan al posicionamiento web en Google

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Martínez Birlanga, Ariel

Tutor/a: Rodríguez Rodríguez, Alejandro

CURSO ACADÉMICO: 2021/2022

Resum

En l'actualitat, el posicionament web s'ha convertit en una de les grans prioritats per a qualsevol empresa que vullga tindre presència web. Per a aconseguir la dita tasca s'ha intentat estudiar els algoritmes PageRank de Google des de fa dècades, però les seues contínues actualitzacions i complexitat han fet del SEO una àrea de desenrotllament àmplia i complexa.

No és difícil hui en dia trobar-nos amb diverses aplicacions, com els famosos web scrapers, capaços d'extraure informació diversa de la World Wide Web i que el seu ús s'ha tornat indispensable dins del sector del posicionament web. Pero, el seu ús garantix realment un bon posicionament? Desgraciadament no, estes aplicacions només analitzen una fracció dels múltiples factors que afecten el PageRank i moltes vegades no es mostra la importància relativa de cada factor, obligant a augmentar el treball manual realitzat.

Esta situació és la que ha motivat la creació d'este projecte, on es tractarà d'automatitzar l'anàlisi dels distints factors que afecten el SEO per a una busca (query) determinada dins del buscador de Google. El codi elaborat tractarà de determinar la importància de cada factor en l'actualitat.

Paraules clau: SEO; posicionament en cercadors; Google; Big data; Scraper; Python; Html; Query; PageRank; API

Resumen

En la actualidad, el posicionamiento web se ha convertido en una de las grandes prioridades para cualquier empresa que quiera tener presencia web. Para lograr dicha tarea se ha intentado estudiar los algoritmos PageRank de Google desde hace décadas, pero sus continuas actualizaciones y complejidad han hecho del posicionamiento orgánico (SEO o Search Engine Optimization) un área de desarrollo amplia y compleja.

No es difícil hoy en día encontrarnos con diversas herramientas, como los famosos web scrapers, capaces de extraer información diversa de la World Wide Web y que su uso se ha vuelto indispensable dentro del sector del posicionamiento web. Pero, ¿su uso garantiza realmente un buen posicionamiento? Desgraciadamente no, estas herramientas solo analizan una fracción de los múltiples factores que afectan al PageRank y en muchas ocasiones no se muestra la importancia relativa de cada factor, obligando a aumentar el trabajo manual realizado.

Esta situación es la que ha motivado la creación de este TFG, en donde se tratará de automatizar el análisis de los distintos factores que afectan al SEO para una búsqueda (query) determinada dentro del buscador de Google. El código elaborado tratará de determinar la importancia de cada factor en la actualidad.

Palabras clave: SEO; posicionamiento en buscadores; Google; Big data; Scraper; Python; Html; Query; PageRank; API

Abstract

Nowadays, Search Engine Optimization (SEO) has become one of the top priorities for any company that wants to have some presence on the web. To achieve this task, the study on Google's PageRank algorithms has been going on for decades, but their continuous updates and complexity have made SEO a wide and complex development area.

It is not difficult nowadays to find various tools, such as the famous web scrapers, capable of extracting diverse information from the World Wide Web, and their use has become indispensable in the SEO sector. But, does their use really guarantee a good positioning? Unfortunately not, these tools only analyze a fraction of the multiple factors that affect PageRank and in many occasions the relative importance of each factor is not shown, forcing to increase the manual work done.

This situation is what has motivated the creation of this project, where we will try to automate the analysis of the different factors that affect SEO for a given search (query) within the Google search engine. The elaborated code will try to determine the importance of each factor at present.

Key words: SEO; Search Engine Optimization; Google; Big data; Scraper; Python; Html; Query; PageRank; API

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1
Introducción

Hoy en día, el uso del buscador de Google se ha vuelto indispensable tanto para usuarios que tratan de ofrecer algún tipo de contenido, como para aquellos que consumen dicho contenido, independientemente de si presenta un carácter navegacional, informacional o transaccional. Debido a esta dependencia casi global hacia el buscador, el posicionamiento de páginas web de Google se ha convertido en uno de los factores más importantes a tener en cuenta para la gran mayoría empresas, instituciones y cualquier otro tipo de entidad con algún tipo de presencia online.

Aunque este hecho se ha vuelto más conocido estos últimos 15 años, el posicionamiento de páginas webs lleva siendo importante desde casi la creación de Google. En ese entonces, los distintos webmasters eran capaces de aprovecharse de los evidentes factores de posicionamiento existentes para aumentar el ranking de sus webs dentro del buscador. No fue hasta 2007 que Google comenzó a tomarse en serio su propio sistema de posicionamiento, complicando la manipulación del posicionamiento mediante el uso de malas prácticas, las cuales sufrieron un duro golpe gracias, especialmente, a los algoritmos diseñados por Google entre el 2011 y el 2015 [1], como por ejemplo: Pandas (evita contenido escueto o duplicado), Penguin (evita webs saturadas de palabras claves) o Pigeon (prioriza las páginas con un domino local igual al usuario).

Actualmente, la manipulación del ranking de posicionamiento de Google sigue siendo posible, pero en comparación a los simples y escasos factores que lo componían hace más de diez años, ahora nos encontramos con una amalgama de factores que se estima en cientos [2] y de considerable complejidad.

Todo esto ha provocado un aumento en la investigación y demanda de este sector, en donde se busca que factores existen, tienen más peso, y se han de modificar para poder mejorar el posicionamiento de tu web.

En el caso de que alguien se pregunte el porqué las empresas y otros creadores de contenido online tienen en tan alta estima este sector, la respuesta es simple. Los usuarios del buscador se han acostumbrado a lo largo de los años a prestar atención solamente a los primeros resultados mostrados, algo que demuestran diversos estudios basados en la visión (eye tracking) y el acceso al enlace (Google CTR) [3], los cuales mencionan que los usuarios no suelen buscar más lejos que los primeros dos o tres resultados. Mejorar el posicionamiento de una web no implica un mero aumento en el tráfico de usuarios y número de transacciones dentro de la web (si son posibles), sino que por un lado se reduciría la popularidad y el tráfico de la competencia, y por otro lado, en caso de que no se utilicen técnicas para mejorar el posicionamiento, se podría producir el aislamiento cuasi completo de la web en respecto a visitas.

Lo que se pretende crear dentro de este trabajo de fin de grado, es un código en lenguaje Python capaz de recolectar una cantidad considerable de factores de posiciona-

miento para búsquedas determinadas y determinar cuáles son los más influyentes dentro de esa búsqueda local. Posteriormente, los resultados obtenidos también serán utilizados para identificar aquellos factores más relevantes a nivel global.

En este momento alguien podría preguntarse qué tipos de factores componen el algoritmo de Google y de porque los resultados para una misma web difieren entre dos búsquedas distintas.

Respondiendo a las dos preguntas, podríamos clasificar dichos factores en dos grupos. El primero de ellos seria aquel grupo de factores que aumentas el posicionamiento de una web de manera global y que por lo tanto sus valores son independientes de la búsqueda realizada, entre lo que nos encontramos: la autoridad de dominio (valoración de una web en la que se estima la probabilidad de obtener un buen posicionamiento), calidad de backlinks (enlaces o dominios de referencia externos que apuntan hacia una página web), longitud de distintos segmentos de la web (URL, título,descripción...), longevidad del dominio y uso de datos estructurados, entre otros. En caso del segundo grupo nos encontramos con aquellas variables relacionadas con las palabras claves, o keywords, utilizadas en la búsqueda, que, dependiendo de sus valores, afectan el posicionamiento web para una búsqueda especifica.

El estudio de palabras claves es, probablemente, el sector más investigado dentro del posicionamiento web. Algo lógico teniendo en cuenta que por muy positivos que sean los factores más genéricos de un web, si aquellos relacionados con las palabras clave, como su inclusión en el título o su porcentaje de aparición en el web, no son adecuados, la web podría acabar en posiciones muy bajas (o nula) dependiendo de la búsqueda. Como ejemplo podríamos usar la web “http://www.upv.es/es”, para la búsqueda “universidades en Valencia” obtenemos dicha URL como primer resultado, pero si en su lugar usamos “universidades tecnológicas” la obtenemos en el decimotercero lugar, un resultado bastante indeseado y que se podría mejorar aplicando las técnicas apropiadas.

Con el objetivo de que se puedan mejorar los resultados para casos como el del ejemplo anterior, dentro de este trabajo de fin grado se estudiarán múltiples factores pertenecientes a ambos grupos, analizando su importancia y variación entre distintos ejemplos de búsquedas.

A continuación, y con las ideas algo más claras, se proseguirá a explicar la motivación y objetivos en los que se basa este trabajo de fin de grado, no sin antes recomendar a los menos expertos en el contexto presentado, la lectura del glosario de términos que se encuentra al final de este mismo documento.

1.1 Motivación

Como ya se ha comentado en la introducción, la motivación que constituye este trabajo de fin de grado es la creación de un programa, en este caso en lenguaje Python, que permita la extracción y el análisis de diversos factores de posicionamiento SEO para búsquedas (también llamadas queries) específicas, para así poder comprender el valor de cada uno de estos factores.

Todo webmaster competente ha sido testigo de cómo las variaciones en el algoritmo de Google han provocado constantes fluctuaciones en el posicionamiento de sus webs y la inconsistencia de sus rankings para distintas queries relacionadas. La creación de un programa capaz de determinar en qué factores fallan dichas webs en comparación de las que se encuentran en las primeras posiciones podría ser realmente útil para aquellos webmasters que tengan problemas, aumentando el posicionamiento de sus webs tan solo

teniendo que ejecutar nuevamente el programa en el caso de que la importancia de dichos factores sea modificadas en un futuro.

En lo personal, mi motivación para la elaboración de este trabajo vino originalmente de mi deseo de realizar un TFG que estuviera relacionado con el análisis de datos, y que al descubrir la asignatura impartida por mi tutor (Social Web Behaviour and Network Analysis), descubrí no solo de qué manera quería hacer mi TFG, sino también sobre de que temática. El análisis de datos SEO no me parecía solo entretenido e interesante, sino que también me motivó la utilidad, popularidad e importancia que se le atribuye a dicha temática. Lo cierto, es que el análisis de parámetros no fue la primera opción para el trabajo, sino que fue la de automatizar la clasificación de palabras claves dependiendo de su tipo de búsqueda (navegacional, informacional y transaccional), algo que fue descartado debido a la dificultad que presenta el análisis del lenguaje natural. Una segunda alternativa fue la de la búsqueda y análisis de keywords para las páginas de una determinada búsqueda, alternativa que fue evolucionando hasta convertirse en la temática actual del este trabajo de fin de grado.

No pretendo simplemente crear un producto que pueda llegar a ser útil para el SEO, sino que también planeo familiarizarme y aprender más sobre un tema que hasta ahora solo conocía de manera superficial.

1.2 Objetivos

Con la premisa de crear un código capaz de extraer el valor de distintos parámetros de búsquedas específicas y analizar su importancia, se han establecido una serie de objetivos, primarios y secundarios, que tratan de describir de manera clara y concisa qué se pretende conseguir con el desarrollo de este trabajo.

A continuación, se mostraran los objetivos de este trabajo de fin de grado ordenados según su importancia:

Objetivos Principales

P.1 - Crear un programa en Python capaz de extraer una cantidad notable de parámetros SEO para las queries deseadas.

P.2 - Calcular la importancia relativa de cada parámetro en función de los resultados obtenidos de las queries extraídas.

P.3 - Permitir la comparación de los datos extraídos durante la ejecución del código para compararlo con una URL específica, recomendando los posibles cambios en caso de que se considere que hagan falta. Se pretende demostrar que este código podría llegar a ser útil a cualquier webmaster.

Objetivos Secundarios

S.1 - Extraer para cada query la posición, densidad y cantidad de cada keyword, para analizar de manera independiente si este hecho afecta de alguna forma al posicionamiento web.

S.2 - Extrapolar los resultados locales obtenidos en diversas ejecuciones del código y usarlos para ilustrar cuales deberían ser los pesos y los valores más adecuados para cada parámetro indiferentemente de la query utilizada.

S.3 - Plasmar los resultados extraídos por el programa en un formato que facilite su comprensión al lector, como por ejemplo en una hoja de cálculo (xslx).

Con los objetivos anteriores se pretende no solo alcanzar conclusiones sobre los diversos factores que componen el SEO, sino que también se pretende crear un código funcional y útil para aquellas personas que quieran mejorar su posicionamiento web.

Los resultados obtenidos al final de este trabajo serán valorados a partir de los objetivos planteados, observando de que manera y hasta que punto cada uno de ellos ha sido alcanzado.

1.3 Estructura de la memoria

Para situar un poco al lector dentro del trabajo, a continuación, se detallará a modo de índice comentado qué nos podremos encontrar en los siguientes capítulos.

En el segundo capítulo se explicarán las bases más teóricas necesarias para comprender el trabajo, tanto relacionadas con el contexto SEO (estrategias, parámetros y aplicaciones) como aquellas herramientas en Python que permiten la creación de programas de extracción y análisis SEO. También se mencionarán algunos de los estudios ya existentes que facilitan la evaluación de las conclusiones. El capítulo concluirá con una pequeña propuesta que pretende demostrar la individualidad de este trabajo.

En el siguiente apartado se expondrán todas las opciones posibles a considerar con anterioridad al desarrollo del código y cuáles han sido seleccionadas en el diseño final. También se detallará la estructura a seguir por el programa.

El tercer capítulo es utilizado a modo de biblioteca tecnológica, detallando que herramientas (lenguaje, librearías y APIs) han sido utilizadas en el desarrollo del programa y de qué manera.

El cuarto punto incluirá información relativa a todo lo que tenga relación con la elaboración del código creado, detallando los algoritmos utilizados, que decisiones se han tenido que tomar en cuanto a su implementación y los problemas encontrados a lo largo del desarrollo.

Los últimos tres capítulos definirán los resultados y conclusiones obtenidas del trabajo.

En el sexto se mostrarán y valorarán los resultados obtenidos del código, tanto de funcionalidad del código como de su precisión en el análisis.

Los resultados obtenidos serán utilizados para crear tanto las conclusiones del séptimo capítulo, como las posibilidades de futuras mejoras del programa detalladas en el capítulo final.

CAPÍTULO 2
Estado del arte

En este capítulo se explicará el contexto tecnológico actual del estudio sobre la obtención y clasificación de parámetros SEO, destacando las técnicas y aplicaciones más eficaces e importantes usadas en la actualidad. Esta sección se dividirá en cuatro partes.

En primer lugar, se documentará las estrategias SEO más importantes además de algunas de las aplicaciones más relevantes que ofrecen una funcionalidad similar a la elaborada en este trabajo, o en otras palabras, aplicaciones que realicen la extracción, análisis o valoración de distintos parámetros SEO.

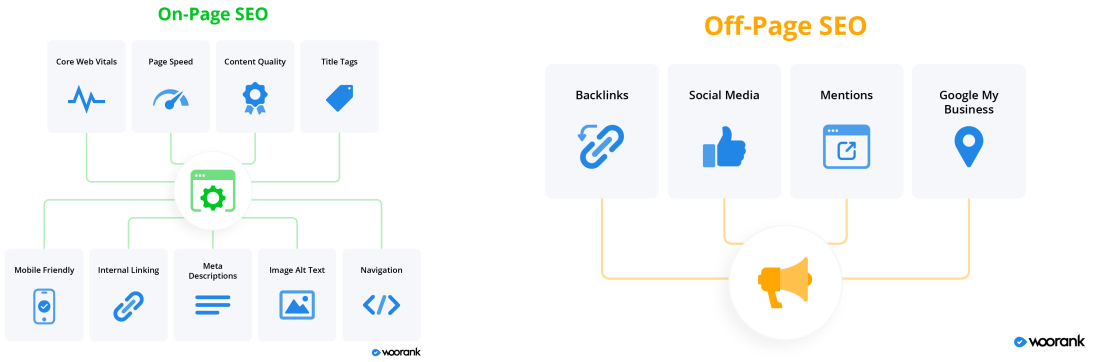
En la segunda sección se mostrarán algunas de las herramientas, bibliotecas y APIs disponibles en Python, las cuales facilitan o permiten la elaboración de aplicaciones centradas en el manejo de parámetros SEO.

En tercer lugar, y a modo de sustituto de la clásica "Crítica al estado del arte", se mostrarán algunos estudios interesantes, no necesariamente pertenecientes a la ETSINF, relacionados con el estudio de parámetros SEO y el posicionamiento en buscadores. Se ha decidido realizar dicha sustitución debido a la falta de trabajos en donde se realice, mediante programación, la extracción y análisis de parámetros SEO de manera conjunta. No obstante, los ejemplos mostrados en esta sección serán de utilidad a la hora de comparar los resultados y conclusiones obtenidas.

Finalmente, se justificará las distintas aportaciones presentes dentro de este trabajo y en que lo diferencia a otros ya existentes.

2.1 Estrategias y aplicaciones SEO existentes

Cualquier aplicación desarrollada en un entorno tan enrevesado como es el SEO precisa del uso de algún tipo estrategia que defina su enfoque y características. Actualmente, las estrategias SEO utilizadas por las distintas aplicaciones SEO suelen estar divididas en dos grupos: **on-page SEO** y **off-page SEO** [4].



(a) Factores on-page **(b)** Factores off-page

Figura 2.1: Estrategias SEO

Fuente: On-Page vs. Off-Page SEO: What's the Difference? https://www.woorank.com/en/blog/on-page-vs-off-page-seo-whats-the-difference

2.1.1. On-page SEO

El primer grupo, al cual se podría decir que pertenece el trabajo desarrollado en este documento, se basa en la optimización de aquellos parámetros que pueden ser directamente modificados por el autor de una web, y que, por norma general, suelen estar directamente relacionados con la experiencia percibida por los usuarios y la facilidad de indexación que poseen los crawlers del motor de búsqueda para la web en cuestión.

Los parámetros que se suelen trabajar dentro del SEO on-page son los siguientes:

- **Keywords:** El parámetro más básico del SEO, formado por el grupo de palabras que componen una query. Se suele estudiar la inclusión, densidad y posicionamiento dentro de las distintas etiquetas que componen el HTML de una web. A diferencia de los otros parámetros, las keywords solamente influyen en el posicionamiento en caso de que formen parte de la query, aunque sea parcialmente.
- **Etiquetas HTML:** La inclusión de ciertas etiquetas, más concretamente las cabeceras (<h1>a <h6>), las cuales son útiles para facilitar la lectura a los usuarios y la indexación de los crawlers de Google.
- **Meta título/descripción:** Elementos del HTML de una web los cuales componen el titulo y la descripción mostrados en Google Search. Su estudio va enlazado al de las keywords y es importante para captar la atención de los usuarios. La inclusión y posicionamiento de las keywords, y la longitud del texto, suelen ser los principales casos de estudio.
- **URL:** Por motivos similares al parámetro anterior, la longitud, el protocolo y la amigabilidad de las URLs (carencia de caracteres extraños) suelen ser estudiadas por su influencia en los SERPs.

- **Links internos:** Este tipo de links hacen referencia a los aquellos enlaces dentro una página que conducen a otra página del mismo dominio. Se estudia su cantidad y relevancia en el contexto de la página inicial, y puede ser útil para que los usuarios y buscadores descubran nuevo contenido de cierto dominio.
- **Imágenes:** El uso, cantidad y relevancia (definido por la etiqueta Alt) de imágenes deben ser investigados en cualquier análisis on-page. Una buena implementación puede ayudar al posicionamiento, pero si no se presta atención al tamaño y exceso de imágenes, esto podría afectar negativamente a la velocidad de carga de una página.
- **Datos estructurados:** Complemento dentro de un HTML de inclusión opcional que facilita la clasificación y la compresión de la información al motor de búsqueda, y que en el caso de Google, se aceptan tres formatos distintos [5]: JSON-LD (recomendado), RDFa y Microdatos. Actualmente, Schema.org ofrece el mejor y más amplio vocabulario para la construcción de datos estructurados.
- **Amigabilidad móvil:** A día de hoy, la mayoría de las búsquedas realizadas en Google no se hacen mediante ordenador, sino que se hacen a través de dispositivos móviles. Desgraciadamente para los webmasters, el rendimiento y el tamaño de dichos dispositivos varía enormemente a los de los dispositivos sobremesa, haciendo necesario adaptar sus webs a dispositivos móviles. Dicha adaptación es denominada como "amigabilidad con dispositivos móviles" y debido al tráfico que proporcionan dichos dispositivos, no es de extrañar su influencia dentro de los SERPs.
- **Core Web Vitals:** Se entiende como Core Web Vitals a la cuantificación de la experiencia de los usuarios en una determinada página. Está compuesto por múltiples factores de los cuales se podrían destacar la velocidad de carga de una página, su interactividad y su estabilidad visual. Sus valores no tienden a ser constantes ya que varían según la calidad de la conexión y el dispositivo utilizado, pero esto no evita convertirla en una de las métricas más importantes dentro del SEO.
- **Navegación:** Una web debe garantizar el acceso a todo su contenido, ya que no cumplirlo podría influir negativamente en el posicionamiento. Errores de conexión y enlaces rotos (errores 404) afectan negativamente a la navegación, mientras que otros elementos que la facilitan, como los sitemaps, la afectaría positivamente.

Cada uno de estos parámetros ha sido incluido de algún modo dentro del producto desarrollado en este trabajo de fin de grado.

2.1.2. Off-page SEO

Por otro lado, en el caso del off-page SEO, las estrategias tratan de promover el contenido de un determinado dominio en páginas de terceros, aumentando así su autoridad y, por consiguiente, su posicionamiento.

Los parámetros que se suelen trabajar dentro del SEO off-page son los siguientes:

- **Backlinks:** Principal y más importante factor dentro del off-page SEO, el cual define la relevancia de una web. Planteado como parámetro, se podría definir como la cantidad y calidad de enlaces que hacen referencia a ella desde paginas externas.
- **Menciones:** Versión reducida del parámetro anterior. En este caso se mide el número de páginas externas hacen mención de una determinada página, pero en este

caso sin el uso de backlinks. Aunque útiles para el SEO, carecen en comparación a la autoridad.

- **Edad:** Tiempo transcurrido desde la creación de un dominio. Se presume que no afecta de manera directa al posicionamiento, pero a mayor edad mayor tiende a ser el número de backlinks y la calidad del contenido, por lo que su estudio puede llegar a ser útil a la hora de comparar distintos dominios.
- **Redes Sociales:** Su uso no afecta al posicionamiento per se, pero se consideran importantes a lo hora de promocionar tu contenido y comprobar las opiniones de los usuarios.
- **Uso de Google My Business:** Herramienta de Google especializada para negocios, la cual permite enviar a Google información específica sobre la localización, horario y datos de contacto. Indispensable para aquellos negocios en donde las transacciones son principalmente in situ.
- **Links externos:** Parámetro de menor importancia que hace referencia a los enlaces de una página que cita a páginas de un dominio externo. Útiles para mejorar la experiencia del usuario, pero no afecta de manera directa al posicionamiento.

Debido a la dificultad que supone la extracción de los parámetros off-page, dentro de este trabajo solo se pudo extraer factores relacionados a la edad y a los links internos.

2.1.3. Aplicaciones SEO

Para concluir esta sección se explicarán algunas de las aplicaciones y herramientas SEO más populares en la actualidad las cuales presentan un funcionamiento similar al del programa desarrollado en este trabajo de fin de grado, independientemente de la estrategia utilizada. [6] [7]

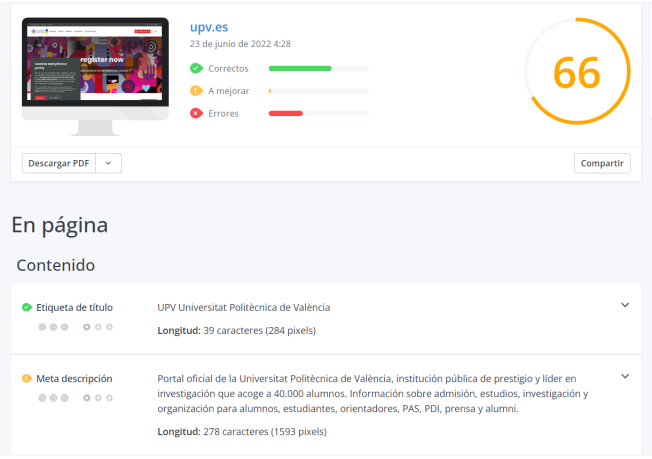


Figura 2.3: Woorank: snippet del informe

Fuente: Elaboración propia a partir de Woorank https://www.woorank.com/es/teaser-review/upv.es?usecase=all

Woorank: Herramienta capaz de analizar parámetros on-page y off-site, y de descubrir las keywords que intenta cubrir cierta web, mostrando entre otros: valores recomendados, las carencias y una puntación general para dicha web. Los resultados son presentados en un informe dentro de su propia web (Figura 2.3), lo cual sirvió como inspiración

inicial a la sección "Informe" generada por el código presentado en este trabajo [8]. A pesar de esto, a diferencia del trabajo desarrollado en este TFG, Woorank no intenta obtener ni mostrar el pesado de cada uno de sus parámetros.

Moz: Plataforma que permite el análisis on-page y de backlinks de una página web. Ofrece, no solo recomendaciones de los parámetros investigados, sino que también es capaz de determinar en que keywords la web debería centrarse para mejorar su SERPs.

Ahrefs: Considerado el crawler más grande y rápido del mundo, a excepción del propio Google. Dentro de sus servicios se encuentran: recomendaciones de que factores mejorar, comprobación de enlaces rotos, mostrar las páginas más populares dentro de un mismo dominio y de investigar los backlinks usados por competidores, los cuales pueden ser usados como punto de partida a la hora de buscar backlinks en potencia.

Screaming Frog: Herramienta famosa por su popularidad entre grandes empresas como Disney y Dell. Sus principales funciones son el análisis de enlaces internos (enlaces rotos, errores de servidor y redireccionamientos) y el análisis de etiquetas HTML, donde se comprueba la falta de etiquetas (cabeceras), duplicidad de meta etiquetas y su longitud.

SEMrush: Número uno en cuanto a comparaciones dominio contra dominio. SEMrush tiene la capacidad de obtener las estrategias SEO usadas por los competidores de cierta web, útil para comprar resultados e imitar aquellas que parecen funcionar a los competidores.

Majestic SEO: Herramienta que incorpora la mayor base de datos de índices de enlaces a nivel mundial, haciéndola indispensable en el estudio de backlinks. Facilita la búsqueda de backlinks de calidad gracias a su capacidad de mostrar el primer millón de webs dentro del ranking de posicionamiento de Google, independientemente de las keywords usadas.

seoQuake: Extensión de Chrome completamente gratuita que permite un análisis sencillo de parámetros on-page.

Google Search Console: Herramienta gratuita para cualquier propietario de una página web y recomendable para usuarios con poco conocimiento sobre el SEO. Otorga la capacidad de estimar el rendimiento de una web, detectar errores básicos del SEO y te ayuda a asegurar que las webs sean fáciles de comprender para Google.

2.2 Herramientas Python existentes

Uno de los principales motivos que hace posible la creación de este trabajo de fin de grado, es la abundante cantidad de librearías y APIs creadas por la comunidad. En esta sección se recopilará y documentará las distintas librerías que permitan la extracción o el análisis de parámetros SEO.

2.2.1. Extracción - Manejo de URLs y HTML

A pesar de la falta de librerías para la extracción directa de parámetros SEO, hay multitud de librerías diseñadas para el manejo de conexiones a web (**Requests** y **Urllib**), análisis de URLs (**Tldextract** y Urllib) y análisis de HTML (**Beautiful Soup**). Dichas librerías combinadas entre sí y entre algunas APIs, que facilitan la extracción de factores específicos (**Google APIs** y **WHOIS**), posibilitan la creación de un **Web Scraper** personalizado como el que implementa el código desarrollado en este TFG.

Conexión Web

Urllib, más concretamente su modulo **urllib.request**[1] se podría considerar como la librería más básica para la lectura y extracción del archivo HTML a partir de su URL, pero palidece en comparación a la librería **Requests**[2], la cual emplea urllib de manera interna, pero que en comparación con ella, su uso es mucho más sencillo e intuitivo, convirtiéndola en la librería favorita de la comunidad en lo que se refiere a conexiones web y extracción del HTML.

Las principales funciones que caen dentro de esta sección son las siguientes:

- Petición GET/HEAD para Abrir URLs (extraer HTML completo/cabecera), permitiendo el uso de cabeceras para evitar el uso de la cache

- Comprobación del código de estado devuelto por una URL. Útil para comprobar enlaces rotos (404) o errores de conexión (5xx).

- Otro tipo de peticiones de menor importancia: DELETE, PATCH, POST y PUT.

Análisis de URLs

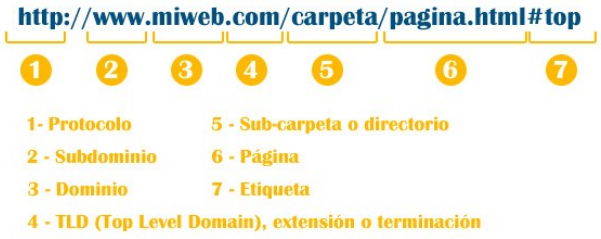


Figura 2.5: Segmentos de una URL

Fuente: https://partesde.info/direccion-web/

En ocasiones no basta con el análisis orgánico de una URL, en algunos casos el uso de secciones específicas de dicha URL (Figura 2.5) puede facilitar el análisis SEO. Dentro de Python se encuentran dos librerías capaces de realizar dicha función: **urllib.parse**[3], módulo de urllib, y **urllib.request**[4].

Aunque parezca que los servicios presentados por urllib.parse y tldextract no son de mucha importancia, lo cierto es que son realmente útiles para la obtención de múltiples parámetros SEO.

Por ejemplo, el protocolo de una web, al formar parte de una URL, puede ser obtenida de manera directa mediante el uso de estas librerías. Por otro lado, mediante la extracción de la raíz del dominio de una URL, se pueden comparar con links dentro de la misma web y diferenciar si se tratan de links internos o externos, además también se puede utilizar para la búsqueda del archivo *robots.txt* (el cual siempre está en la raíz), y que entre otras cosas, puede llegar a tener la dirección de los sitemaps.

[1]Manual urllib.request: https://docs.python.org/3/library/urllib.request.html
[2]Manual Requests: https://www.w3schools.com/python/module_requests.asp
[3]Manual urllib.parse: https://docs.python.org/3/library/urllib.parse.html
[4]Repositorio en GitHub tldextract: https://github.com/john-kurkowski/tldextract

Análisis HTML

Una vez obtenido el archivo HTML, un Web Scraper debe ser capaz de analizar cada una de sus secciones para realizar la extracción de parámetros. Para completar esta tarea nos encontramos con **Beautiful Soup**[5], la librería predilecta en la creación de Web Scrapers con Python.

Beautiful Soup es capaz de organizar el objeto HTML de una manera comprensible para Python, permitiendo filtrar su información por:

- Etiqueta HTML usada (<a>, ,<alt>,<src>...).
- Segmento del HTML (title,body,h1,h2,h3...).
- Variables usadas (ej: "href" para enlaces referenciados), ya sea por el nombre o el valor de la variable.

Al permitir la extracción de información de manera precisa, Beautiful Soup puede llegar a ser realmente útil en la obtención de los siguientes parámetros SEO:

- **Enlaces internos y externos:** Filtrando mediante la etiqueta <a>(hiperenlace) y su variable "href".
- **Uso de etiquetas:** El filtrado por etiquetas permite extraer información sobre las cabeceras usadas, número de imágenes, etc...
- **Keywords y longitud:** La filtración por segmentos (más específicamente su texto), nos permitiría contabilizar su longitud, número de keywords que aparecen por segmento.

APIs

A diferencia de las librerías, sí que existen APIs capaces de ofrecer el valor de cierto parámetro SEO de forma directa.

Por un lado tenemos a la **WHOIS API** que es accedida mediante la librería WHOIS[6] de la cual se puede extraer múltiples variables de una web útiles para el SEO como son su dominio, cabecera, estado de la página y sobre todo la fecha de creación de la web, ya que el resto se pueden obtener fácilmente mediante las librerías anteriores.

Por otro lado, tenemos a las APIs de Google, que, a pesar de poseer una velocidad de ejecución qué solo puede ser definida como lenta, dichas APIs ofrecen, de manera muy directa, el valor de algunos factores SEO, y que al pertenecer a Google, su autenticidad siempre será absoluto para los análisis SEO centrados para su buscador.

Dentro de sus APIs nos encontramos con dos que ofrecen todo lo anterior: **mobileFriendlyTest**[7], para la obtención de la amigabilidad móvil de una web, y **PageSpeed Insights**[8], para la obtención de la velocidad de carga en dispositivos móviles y sobremesa, y el análisis de los Core Web Vitals.

[5]Manual Beautiful Soup: https://www.crummy.com/software/BeautifulSoup/bs4/doc/
[6]Repositorio en GitHub WHOIS: https://github.com/richardpenman/whois
[7]Guía mobileFriendlyTest API: https://developers.google.com/webmaster-tools/search-console-api/reference/rest/v1/urlTestingTools.mobileFriendlyTest/run
[8]Guía PageSpeed Insight API: https://developers.google.com/speed/docs/insights/v5/get-started

2.2.2. Análisis - Creación de clasificadores

Uno de los objetivos del programa de en este TFG es el de analizar en qué medida afecta cada factor SEO al posicionamiento, tanto de manera automática para queries especifica, como la estimación de sus pesos globales de manera manual en base a los pesos locales.

Debido a esto, el uso de alguna técnica de aprendizaje automático que permita observar el peso de cada elemento por los que están compuestos sus datos es indispensable. Por lo tanto, en esta sección se documentara las herramientas para el análisis SEO estudiadas para este trabajo, haciendo especial hincapié en el estudio de algoritmos de clasificación.

Clasificador

Para ser exactos, se ha investigado un tipo concreto de algoritmo de aprendizaje automático, **Gradient Boosting**. Existen dos motivos para limitar la investigación, siendo el primero su popularidad la cual proviene de su alto rendimiento para el procesamiento de datos tabulares (en formato de tabla) y, como algunas comparaciones sugieren [9], puede considerarse como el algoritmo más eficaz dentro del machine learning. Según Brownie (2020) [10], "Gradien Boosting posee buen rendimiento, si no el mejor, en una amplia gama de conjuntos de datos tabulares, y las versiones del algoritmo como XGBoost y LightBoost a menudo juegan un papel importante para ganar competencias de aprendizaje automático".

El segundo motivo se debe a que se puede solventar la principal desventaja de dicho algoritmo que, como describe Brownie (2020) [11], "Un problema importante del Gradient Boosting es que es lento para entrenar el modelo. Esto es particularmente un problema cuando se usa el modelo en grandes conjuntos de datos con decenas de miles de ejemplos (filas)".

Afortunadamente, el uso del programa está pensado para un número reducido de queries y el número de búsquedas por query no tiende a salirse del rango 10-200, ya que no suele interesar las páginas con un rango peor a ellas, por lo que estaríamos muy lejos de un programa de decenas de miles de filas.

Explicado de manera simple, Gradient Boosting es un algoritmo de aprendizaje automático iterativo y potente, basado en arboles de decisiones y en la minimización de la función de perdida.

Hay tres elementos principales en el cual gira el algoritmo [12]:

- **Función de pérdida:** Determina en que grado afecta una mala clasificación en el entrenamiento del modelo y varía según el tipo de problema a resolver.
- **Clasificadores débiles:** Esta construido en base arboles de decisión que funcionan como clasificadores débiles, los cuales por sí solo no ofrecen una precisión aceptable.
- **Modelo aditivo:** En cada iteración del modelo se añadirá uno de los clasificadores débiles Dichas adiciones se harán aplicando descenso por gradiente para minimizar perdidas al añadir el árbol e intentar reducir la función de perdida con cada iteración.

Como cualquier otro clasificador, Gradient Boosting posee una serie de variables las cuales pueden afectar a su precisión y a su ejecución en general, siendo las más importantes la tasa de aprendizaje (contribución de cada árbol) y el número de árboles de decisión añadidos al modelo (número de etapas). Por su parte, Friedman (1999) [13] menciona que, "similar a una tasa de aprendizaje en la optimización estocástica, la contracción reduce la influencia de cada árbol individual y deja espacio para que futuros árboles mejoren el modelo".

Es importante experimentar con los valores de dichas variables, ya que sus valores óptimos pueden variar según los datos a estudiar, pero se considera que una tasa de aprendizaje entre 0.1 y 0.3 suele ser lo óptimo. Además, cabe destacar qué, aunque disminuir el valor de la tasa de aprendizaje suele aumentar la precisión del clasificador y que disminuir el número de árboles reduce el tiempo de ejecución, esto solo es posible cuando hay un buen equilibrio entre ellos. Esto viene indicado por Friedman (2001) [14], que sugiere que, "disminuir el valor de v [la tasa de aprendizaje] aumenta el mejor valor para M [el número de árboles]".

Afortunadamente, implementar dicho clasificador no es relativamente sencillo. Actualmente Python cuenta con numerosas librerías que permiten su uso como XGBoost, LightGBM, CatBoost y **Sckit-Learn**[9], destacando la última de ella debido a su simplicidad, popularidad y variedad de herramientas de aprendizaje automático que incluye mucho más que el algoritmo Gradient Boosting.

Tratamiento de datos

Para el análisis generado por el programa a desarrollar en este trabajo, el uso de un clasificador no es suficiente. Con un clasificador se podría predecir en que rango de posiciones una web se podría encontrar y ofrecer la importancia (peso) para cada una de las variables SEO analizadas, pero dicho clasificador sería incapaz de decidir el nivel de error de cada parámetro, ni de establecer una serie de valores recomendados, algo necesario si se quiere mostrar al usuario en que falla la web a analizar y que valores serian recomendados.

En los programas SEO actuales, el análisis del error normalmente se decide mediante la aplicación de métricas previamente creadas (constantes en su mayoría) con los resultados obtenidos. En el caso de este trabajo, se pretende obtener los **valores recomendados** de manera dinámica a partir de los resultados obtenidos de una selección de queries. Por ello, la obtención de medias y varianzas de los datos es necesaria para una buena selección de dichos valores. Afortunadamente, Python posee una biblioteca llamada **NumPy**[10], la cual incorpora diversas funciones para la creación y operación de vectores y matrices, y es lo suficientemente versátil para cubrir por si sola este requisito menor.

Por último, dentro del tratamiento de datos SEO no se pueden dejar pasar las **stopwords**. En un gran número de ocasiones, los usuarios suelen incluir dentro de sus búsquedas palabras que presentan una alta densidad en la mayoría de los documentos debido a a su necesidad gramatical (artículos, pronombres, preposiciones...). Estos términos tienden a ser excluidos (stopwords) en el análisis de keywords debido al bajo valor que suelen poseer y a su facilidad de corromper un análisis de densidad. Debido a esto el uso de APIs como **NLTK**[11] son indispensables para el preprocesado de parámetros SEO relacionados con keywords.

[9] Manual sklearn GradientBoostingClassifier: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html
[10] Manual NumPy: https://numpy.org/doc/stable/numpy-ref.pdf
[11] Documentación NLTK: https://www.nltk.org/search.html?q=stopwords&check_keywords=yes&area=default

2.3 Estudios relacionados

A causa de su utilidad, el SEO ha sido investigado en infinidad de ocasiones debido a la importancia de las SERPs, pero, a pesar de esto, la mayoría de los estudios se centran en la investigación manual (usado las herramientas vistas en 2.1.3.) mediante la cual se sacan conclusiones de la importancia, valores medios y valores recomendados de distintos factores SEO. Algo que difiere bastante con el objetivo de este trabajo, donde se pretende automatizar el proceso mediante el simple uso de un script Python, el cual pueda generar dichas conclusiones por sí mismo.

Un punto a favor de esta situación, es que al ser trabajos de investigación basándose en el uso de herramientas profesionales y de calidad, sus resultados y conclusiones pueden llegar a ser muy beneficiosas a la hora de comparar los obtenidos en este trabajo, siendo las tesis doctorales (UCM) desarrolladas por *Urosa Barreto* [15] y *Carreras Lario* [16] las que se consideraron más adecuadas y completas para realizar dicha comparación.

Las conclusiones obtenidas en las tesis de *Urosa* y *Carrera* se centran en el valor medio y probabilidad de algunos factores SEO en los primeros tres resultados de distintas búsquedas.

De dichas conclusiones se han filtrado aquellas que permiten una comparación directa con los resultados que proveerá este trabajo:

	Urosa Barreto	**Carreras Lario**
Antigüedad media del dominio (años)	15	8,15
Carga de timpo media (segundos)	2,45	-
Densidad de keywords	1,38 %	2,35 %
Probabilidad de aparición keyword (Título)	81 %	93,3 %
Probabilidad de aparición keyword (Descripción)	73 %	-
Probabilidad de aparición keyword (H1)	75 %	76,6 % (h1 y/o h2)
Probabilidad de aparición keyword (Alt)	11 %	34,8 %

Tabla 2.1: Conclusiones de parámetros SEO de *Urosa Barreto* y *Carreras Lario*

Fuente: Elaboración propia a partir de las fuentes consultadas para esta sección.

Hay dos factores a tener en cuenta antes de observar la tabla. El primero es la diferencia temporal en la publicación de las tesis de *Urosa* y *Carreras*, estando publicadas en el 2020 y 2012 respectivamente, lo cual se refleja claramente en los datos sobre la antigüedad del domino, y el segundo factor sería la temática sobre la cual se ha realizado dicho análisis. Convenientemente, en el caso de *Carreras* será realizado un análisis general sin entrar en ningún tipo de contexto, pero en el caso de *Urosa*, su investigación se centra en los resultados dentro de búsquedas relacionados con el sector educativo superior, algo que podría afectar ligeramente a los resultados.

Desgraciadamente para esta sección, los trabajos procedentes de la ETSINF tienden a salirse mucho del enfoque dado en este proyecto, siendo más común el análisis de herramientas SEO en si o en el análisis actual o histórico del SEO. A pesar de no dar recomendaciones directas sobre los valores de los factores, encuentro necesario mencionar que, en las fases iniciales de desarrollo, el trabajo presentado por *Alós Moya* [17] fue de gran ayuda para la elaboración de este trabajo, ya que a pesar de presentar un enfoque totalmente distinto al presente, el contexto del posicionamiento web presentado por Alós fue indispensable para asentar las bases teóricas básicas necesarias para la elaboración de este TFG.

2.4 Propuesta

La propuesta de este estudio, como se ha ido comentando a lo largo del documento, es el de crear un programa que tenga la capacidad de estimar los valores recomendados y pesados de un cierto número de para una variedad de parámetros SEO de manera similar al de herramientas como Woorank.

Lo que diferencia mi proyecto de otras herramientas es la individualidad de cada análisis, donde al contrario de otras aplicaciones, no se utilizan datos previamente recolectados o métricas previamente establecidas para obtener los resultados. Los algoritmos de posicionamiento de Google son modificados de manera constante a lo largo de los años, algo que se tuvo muy en cuenta para la realización de este trabajo, donde se pretende crear un código ejecutable que permita realizar un análisis in situ de diversos parámetros SEO, pensado para evitar posibles sesgos creados por métricas y análisis previos (en el caso de que los algoritmos de búsqueda cambien).

La representación de datos y la sencillez del código también difiere notablemente de cualquiera de las aplicaciones mostradas en 2.1.3. Como se menciona en el resumen del trabajo, un elemento que no muestran dichas aplicaciones es el peso de cada uno de los factores analizados, algo muy importante para saber cuál priorizar en caso de que sus valores se salga de lo recomendable, limitándose solo a mostrar el nivel de error de cada uno de ellos (en comparación a los valores recomendados). El código, desarrollado como un simple script de Python, pretende demostrar que no hace falta el uso de ninguna base de datos o análisis SEO manual previo para poder crear un programa que funcione como web crawler y analizador SEO, siendo simplemente necesario un entorno Python para su creación.

El programa a desarrollar, aunque difícilmente pueda llegar al nivel de calidad de las herramientas encontradas en el mercado, pretende realizar un análisis similar al de ellas de una manera más dinámica, sencilla (a nivel de código) y mejorando la representación de los factores por medio de la visualización de pesados obtenidos mediante un clasificador potente como es Gradient Boosting, lo cual ayudaría a comprender la importancia de cada uno de los factores.

CAPÍTULO 3
Metodologías y diseño

Ya obtenidas las bases teóricas necesarias, podemos comenzar a plantearnos las distintas formas de las que podemos abordar la creación del programa a desarrollar, seleccionando, de entre todas las posibilidades, las que más se ajusten a los objetivos planteados y al tiempo disponible. Descartaremos aquellas opciones que no se ajusten o que su implementación no seo realista o factible.

3.1 Identificación y análisis de metodologías posibles

Para elaborar un código lo más completo posible, se deberán estudiar qué factores SEO conviene incluir en el análisis, y, en el caso de las keywords, al estar investigando un elemento relacionado con el lenguaje natural, se deberá tener especial precaución a la hora de analizarlos. Al ser un código el producto a desarrollar, será conveniente para garantizar el mejor resultado posible, el planteamiento de distintos métodos para la optimización y practicidad del código.

3.1.1. Posibles factores a investigar

En el capítulo anterior se observaron los principales parámetros usados dentro de las estrategias SEO. A continuación, se planteará un listado de dichos parámetros y sus posibles métodos de obtención para hacernos una idea cuales incluir en el modelo final del programa.

Parámetro/s	**Método/s de obtención**
Coincidencia total de las KWs (title/description/body/h1/alt/src)	Localización de coincidencias en el texto HTML.
Coincidencia parcial de las KWs (title/description/body/h1/alt/src)	Localización de coincidencias en el texto HTML a una cierta distancia de edición.
Uso de etiquetas de la cabecera (H1-H6)	Buscar si hay instancias al extraer (Beautiful Soup)
Longitud del meta título/descripción	-Obtener texto de meta etiqueta (Beautiful Soup) y medir. -Obtener texto desde Google Search (Beautiful Soup) y medir.
Longitud del URL	Obtener 'href' desde Google Search.

Protocolo del Url	Obtener 'href' desde Google Search (Beautiful Soup) y seccionar (urllib).
Amigabilidad de la URL	Obtener 'href' desde Google Search (Beautiful Soup), buscar secciones/símbolos indeseados (urllib).
Enlaces internos (follow/nofollow)	Obtener 'href' del HTML (Beautiful Soup) y comparar raíz del URL (urllib/tldextract).
Uso de imágenes	Buscar etiqueta <img>en el HTML (Beautiful Soup).
Uso de videos	Buscar etiqueta <video>en el HTML (Beautiful Soup).
Uso de datos estructurados	Buscar coincidencias con el vocabulario propio de JSON-LD/Microdata/RDFa.
Uso marcado Schema (Schema.org)	Buscar coincidencias con el vocabulario propio de Schema.org.
Compatibilidad con dispositivos móviles	Mandar petición a la API de Google(mobileFriendlyTest).
Velocidad de carga de la página (móvil/sobremesa)	Mandar petición a la API de Google (PageSpeed Insights).
Rendimiento de la página (móvil/sobremesa)	Mandar petición a la API de Google (PageSpeed Insights).
Enlaces rotos (errores 404)	Obtener código de estado de los links internos/externos (Request).
Errores de conexión	Obtener código de estado de los links internos/externos (Request).
Uso de robots.txt	-Buscar en el directorio y obtener código de estado (Urllib/Request).
Uso de sitemap.xml	-Buscar en el directorio y obtener código de estado (Urllib/Request). -Buscar posible mención en robots.txt (Beautiful Soup).
Backlinks	- Uso de API de pago. -Búsqueda manual mediante herramientas SEO. -Backlinks limitado al resto de webs analizadas.
Menciones	Búsqueda manual limitada al resto de webs analizadas.
Antigüedad del dominio	Uso de API WHOIS.
Opinio publica en redes sociales	Análisis manual.
Uso de Google My Business	Busqueda de elementos característicos en Google Search.
Enlaces externos (follow/nofollow)	Obtener 'href' del HTML (Beautiful Soup) y comparar raíz del URL (urllib/tldextract).

Tabla 3.1: Posibles factores SEO a analizar

Fuente: Elaboración propia.

Como se observa en la tabla, la totalidad de los parámetros que caen dentro de la categoría on-page poseen métodos de extracción muy realistas pero, en el caso de alguno de ellos, se habrá que reconsiderar si deben ser implementados al ser menos preciosos o al poseer un coste temporal demasiado elevado.

Los parámetros que reconsiderar son los siguientes:

- **Amigabilidad de la URL :** No es posible determinar de manera exacta si Google considera una URL amigable con los medios disponibles en Python, pero sí que podremos realizar una aproximación bastante razonable mediante la búsqueda de símbolos extraños, pertenecientes a un listado creado de forma manual y la búsqueda de segmentos secundarios como las etiquetas o las queries[1], dado que normalmente solo sirve para filtrar la información de una misma página.

- **Datos estructurados y marcado Schema :** Los datos estructurados no se considera un segmento especifico de un HTML, sino que es un tipo de estructura dentro de ella la cual facilita a Google entender la estructura de la página. Debido a esto, no es posible buscar su existencia de manera directa, sino que deberemos buscar coincidencias en el HTML con el vocabulario propio de algún formato de datos estructurados (JSON-LD, Microdatos y RDFa) y buscar si se hace mención a Schema.org. Realizado de la manera apropiada se puede llegar a detectar correctamente todas las webs que usen estos formatos pero, aunque es poco probable, una web podría ser clasificada erróneamente como que usa datos estructurados en caso de que parte de su texto coincida con el vocabulario, pero este no sea usado para formar datos estructurados.

- **Compatibilidad móvil, rendimiento y velocidad de carga:** Nos encontramos, dentro de la estrategia on-page, con los parámetros que mejor describen la experiencia del usuario, haciendo evidente su importancia dentro del SEO. La única contra en la extracción de dichos parámetros es que, al obtener los resultados mediante peticiones a las APIs de Google, el tiempo de ejecución podría llegaría a ser relativamente elevado.

- **Errores 404 y de conexión:** Para obtener posibles errores se tendrá que pedir una respuesta a todos los links internos y externos de una página de manera secuencial, haciendo de este parámetro costoso en cuanto a tiempo, pudiendo extender de manera exagerada el análisis de páginas con un alto numero de links, como, por ejemplo, las paginas de wikipedia.org. Además, en el caso de errores de conexión tendremos que seleccionar si centrarse en errores de conexión con el servidor (5xx) o cualquier otro tipo de error que impida la conexión con la página (4xx) a excepción del código 404.

En el caso de los parámetros off-page, los métodos de extracción planteados tienden a ser bastante indeseable, en algunos casos su extracción imposibilitarían la automatización total que pretende tener el programa a desarrollar (menciones y opinión en redes) o presentan una complejidad de extracción demasiado grande (backlinks) o imprecisa (Google My Business) para ser incluidos.

[1] Estructura general de una URL: Protocolo://dominio/dirección;parámetros?query#etiquetas

3.1.2. Análisis de keywords

Para la elaboración de un buen análisis SEO, el estudio de las keywords siempre ha sido de extrema importancia, ya que, como se comentaba en la introducción, los parámetros relacionados con ellos son los encargados de filtrar los resultados obtenidos dependiendo de la query realizada en el buscador, e incluso podría situar una web por encima de mayor PageRank si dicha web presenta una mejor cantidad y posicionamiento de keywords. Debido a estos motivos, la inclusión de los parámetros relacionados es indispensable.

Respecto a la manera en la que se analizaran las coincidencias totales, nos encontramos con tres posibilidades:

- **Densidad:** Que porcentaje del texto (de una etiqueta determinada) está compuesto por las keywords. Útil para una comparación objetiva entre distintas webs.
- **Cantidad:** Número de instancias de cada keyword en el texto. Los resultados pueden variar considerablemente dependiendo del tamaño del documento. Útil como complemento a la densidad, pero de baja importancia desde la inclusión del algoritmo Penguin, el cual evita que páginas con una alta saturación de keywords se sitúen en las primeras posiciones [1].
- **Posición:** El posicionamiento de las keywords, especialmente en las primeras posiciones del documento, es un factor en potencia a analizar, pero debido a la naturaleza del factor, generar recomendaciones y pesados sobre él es contraproducente debido a la alta varianza entre el posicionamiento de keywords para cada web.

Es posible realizar los tres análisis de manera conjunta, pero habría que replantearse la manera de incluir la cuantificación y el posicionamiento de las keywords por a las dificultades que presentan a la hora de comparar los resultados obtenidos de diferentes webs.

Normalmente Google no solo busca paginas según la aparición exacta de una keyword, sino que también considera la aparición de palabras derivadas de esta, siendo las *coincidencias parciales de las keywords* el parámetro que contabiliza dichas palabras. El análisis de las coincidencias parciales es idéntico del de las coincidencias totales, pero en este caso también tendremos que plantearnos su obtención.

Entre las diferentes opciones para su extracción se encuentran:

- **Stemming:** Método para la reducción de una palabra a su raíz léxica. Al aplicarla a todas las palabras y keywords se podría realizar una segunda búsqueda de coincidencias totales (de raíces) para encontrar el número de coincidencias parciales. El mayor contra de este método es que en el caso de que existan palabras con la misma raíz, estas serán contabilizadas.
- **Distancia levenshtein:** Algoritmo utilizado para obtener el número de operaciones necesarias para transformar una palabra a otra (distancia de editado). Hace uso de un límite arbitrario para decidir el número máximo de operaciones, superado dicho límite, las palabras no generaran una coincidencia. En esta ocasión algunas palabras pueden ser erróneamente clasificadas en el caso de que se escriban de manera similar a pesar de la diferencia en sus significados. Muchos de estos errores pueden ser evitados si se utiliza un valor límite adecuado el cual dependa de la longitud de la palabra (keyword) a comparar.

- **Personalizado:** Más sencillo y menos eficaz a los anteriores. Un ejemplo de este método seria la estimación de la raíz de una manera cruda, reduciendo la longitud de palabra un cierto número de caracteres.

Una última técnica que debe considerarse en el análisis de keywords es la eliminación de stopwords. Como se mencionó en el capítulo anterior, las stopwords son indispensable para evitar echar a perder el análisis de densidad. Desafortunadamente, en algunos casos el usuario puede llegar a desear el análisis de una keyword que es considerada una stopword por el programa. Debido a esta contradicción, debemos de plantearnos que grado de libertad ofrecer al usuario a la hora de utilizar stopwords.

Se han planteado las siguientes ideas para el uso de stopwords:

- Uso estricto de stopwords (no recomendado, problemas en casos concretos).
- Prescindir de stopwords (no recomendado, problemas con la densidad).
- Activar/Desactivar stopwords (versátil).
- Prescindir de stopwords e introducir manualmente las keywords (versátil).

3.1.3. Practicidad y elecciones del código

Dentro del código a desarrollar existen otros elementos que afectan a la extracción de parámetros y la elaboración de resultados. Esta sección explicara cuales son dichos elementos y las distintas opciones disponibles para su elaboración.

Obtención de URLs

Un paso previo a la extracción de cualquiera de los parámetros a investigar es la obtención de las URLs en las primeras posiciones para las queries a investigar.

La obtención de URLs dentro de una web es bastante directa, hay que obtener todas las instancias de 'href' dentro del HTML, pero en el caso de las URLs que componen las primeras posiciones de una query la cosa cambia.

Python ofrece más de una opción para su obtención, siendo las más lógicas el uso de la librería **googlesearch** [2] o el análisis directo de la página web de Google Search correspondiente.

En el caso de utilizar la librería, el código sería bastante sencillo ya que sus módulos se encargarían de todo el proceso de extracción. En caso contrario se deberán de usar librerías como *Requests* y *Beautiful Soup* en el HTML de un URL con un determinado esqueleto[3], para así obtener las primeras instancias de *href* correspondientes a la query. A pesar de ser una extracción similar a la que se harían a las webs a analizar (para links internos/externos), esta segunda opción requerirá de un filtrado para remover enlaces no deseados como los correspondientes a sitelinks (complementarios a otros enlaces) o anuncios (de pago/anuncios).

[2] Documentación Googlesearch: https://python-googlesearch.readthedocs.io/en/latest/
[3] Esqueleto del análisis directo: URL = https://google.com/search?q= + query

Previo al desarrollo final se han realizado una serie de pruebas para seleccionar que método de extracción utilizar. Se han ilustrado las cualidades positivas y negativas de cada método en la siguiente tabla:

	Positivo	**Negativo**
Análisis directo	-Muy rápido (menos de un segundo por enlace) -Extremadamente difícil de obtener error 429 (solo fue obtenida al paralelizar mucho la extracción)	-Requiere de un código más elaborado -Mayor dificultad de personalizar las opciones del buscador -Precisa de un filtrado de sitelinks y anuncios
Librería Goooglesearch	-Obtención directa y sencilla -Personalización (ej: cambiar el idioma de la búsqueda)	Lento (varios segundos por enlace) -Búsquedas grandes o seguidas puede llegar a producir error 429: Demasiadas peticiones -Precisa de filtrado de sitelinks

Tabla 3.3: Cualidades de los métodos de extracción de URLs en Google Search.

Fuente: Elaboración propia.

Paralelismo y concurrencia

Es imprescindible para cualquier programa con tiempos de ejecución elevados la implementación de algún método de paralelismo que permita reducir el coste temporal del programa. El volumen del código a desarrollar dependerá del número de búsquedas que se quieran realizar y, al ser necesario el envío de peticiones a diversas URLs para obtener su HTML o su código de estado, el coste temporal obtenido podría llegar a ser muy elevado si se realiza de manera secuencial. Esto es debido al tiempo consumido en la espera a dichas peticiones, las cuales pueden ralentizar mucho el análisis de una web y en el peor de los casos podría llegar a bloquear el código. El segundo problema puede ser resuelto de manera simple mediante el uso de **timeouts**, los cuales descartan la búsqueda en caso de que cierto tiempo haya concurrido sin obtener respuesta del servidor. Desgraciadamente, el primer problema no tiene una solución sencilla y habrá que utilizar algún método de paralelismo o concurrencia para reducir el tiempo de ejecución del programa.

Existen dos métodos que nos permitirían la ejecución paralela de varias webs:

- **Multiprocesamiento:** Permite el uso de los distintos núcleos disponibles en el ordenador, permitiendo la ejecución paralela de tantos procesos como núcleos utilizados. Aunque conveniente, el número de procesos está limitado al número de núcleos disponibles y su uso aumentaría notablemente los recursos del ordenador utilizados.
- **Multithreading:** Considerado como una forma falsa de paralelismo, multithreading permite la creación de threads, subprocesos que el procesador va accediendo de manera cíclica, haciendo parecer que se ejecutan paralelamente. En el caso de Python, si no se realiza ninguna modificación, los threads son ejecutados en un solo núcleo, haciendo inconveniente su uso en segmentos con una alta cantidad de operaciones, y convenientes para códigos que pasan largos periodos de tiempo en espera de algún tipo de evento (como recibir la respuesta de un servidor web). Una cantidad excesiva de threads podría causar problemas en la ejecución del código, por lo que conviene realizar pruebas para determinar un límite apropiado.

Aunque el uso concurrente de ambos métodos es posible, esto podría afectar drásticamente a la CPU y memoria consumida en el ordenador, además de que aumentaría enormemente la aparición del error 429 debido al exceso de peticiones enviadas a Google en un corto periodo de tiempo. Esto haría del threading la opción ganadora ya que el motivo por el que se plantea el uso de paralelismo en un principio es para aprovechar el tiempo que el programa estará inactivo (esperando respuesta de servidores), algo en lo que el multiprocesamiento se vería muy limitado por el número de núcleos del hardware.

3.1.4. Apartado visual

Aunque el enfoque de este trabajo se centra en la funcionalidad del código, la interfaz utilizada para interactuar con el usuario y la representación de resultados es algo a tener muy en cuenta.

Para mejorar la estética del código y la experiencia del usuario (a pesar de no ser objetivos del trabajo) se ha decido utilizar una interfaz gráfica, ya que, por muy simple que sea, siempre será más sencillo y agradable que una simple terminal. Por motivos similares, se ha decidido empaquetar todo el programa en un ejecutable que simplificará su ejecución.

En el caso de la representación de resultados su utilidad es mucho más practica y esencial. El programa diseñado debe mostrar múltiples parámetros de distintas queries y resultados derivados de estas. Debido a la estructura matricial de los resultados, lo más razonable sería plasmarlos en un formato que admita la visualización de tablas, como es el caso de xslx, formato utilizado por programas como Excel y que facilita el manejo de este tipo de datos.

3.2 Diseño final

Tras analizar todos los elementos que podrían ser incluidos en el código y sus posibles métodos de implementación, se ha llegado una conclusión de cuales de entre todos ellos serán los que definan el programa a desarrollar. A continuación, se explicará de manera ordenada la estructura del código, mencionando en todo momento las elecciones realizadas.

3.2.1. Inicialización

Al ejecutar el programa se mostrará mediante una interfaz gráfica las distintas opciones a rellenar:

- **Numero de búsquedas/queries:** Define el número de queries distintas sobre las que se buscarán resultados.
- **Numero de resultados:** Numero de URLs obtenidas por cada búsqueda. Los resultados corresponderán a las URLs en las posiciones [1-Numero de resultados].
- **Keywords = Query:** Opción booleana que, en caso de ser positiva, las keywords de cada query serán las que palabras que componen la query, ofreciendo la implementación clásica de keywords con filtrado de **usando stopwords**). En caso contrario se deberán de introducir las keyword de manera manual para cada búsqueda, siendo útil si se quiere analizar alguna palabra que comúnmente se considera una stopword. Como se puede observar esta opción ha sido diseña para incluir las dos opciones más versátiles para la inclusión de stopwords.
- **Ranking inferior límite:** Numero de URLs por query consideradas como mejores resultados y que servirán como el estándar a seguir. Los resultados se verán afectadas por dicho límite ya que las URLs con un ranking menor o igual al especificado, en esta opción serán utilizadas para crear la clase óptima para el clasificador y definirán en gran medida los valores recomendados mostrados por el programa.
- **Comparar resultados con una URL:** Opción booleana por si se quiere comparar los resultados obtenidos con una cierta URL. Eficaz para demostrar la utilidad del programa, ya que les serviría a los webmaster a orientarse en que parámetros sus webs fallan y que les diferencia de otras webs con mejores posiciones.

Dependiendo de los valores escritos anteriormente, se mostrarán algunas de las siguientes opciones adicionales:

- **URL del Informe** Como su nombre indica, se escribirá el nombre de la URL que se dese comparar. Solo en caso de seleccionar *Comparar resultados con una URL*.
- **Query:** Query a la que analizar. Se deberá escribir el mismo número de queries como las seleccionadas en la primera opción.
- **Keywords:** Keywords a analizar en una query específica. Ejecutado una vez por query en caso de que la opción *Keywords=Query*. Ejecutada una vez en caso de seleccionar *Comparar resultados con una URL*.

3.2.2. Búsqueda de parámetros SEO y resultados locales

Teniendo en cuenta que la extracción de parámetros consume mucho tiempo, se debería prescindir de librerías como *googlesearch*, que tiene facilidad para producir errores 429 si se utiliza un número alto de búsquedas, se ha decidido realizar el análisis directo de las URL a pesar de ser más laborioso.

Debido a que cada página de Google Search solo presenta alrededor de 10 resultados, en el caso de querer superar dicha cifra habrá que modificar el esqueleto de la URL usada para sacar los enlaces de las web a analizar[4].

Ya obtenidos los datos básicos necesarios, se comenzará con la extracción de parámetros. Para reducir el tiempo consumido en la extracción, se crearán un thread por cada URL a analizar, o en otras palabras:

Threads creados = Número de búsquedas * Número de resultados

Además, como se mencionó anteriormente, las peticiones usadas para pedir información de una URL incorporarán un timeout que limite el tiempo máximo de espera a una respuesta. Para no dejar ningún thread activo durante la ejecución del programa, algo posible en caso de que su URL correspondiente tengan cientos de enlaces a analizar y estos tengan una respuesta muy lenta, estos también incluirán un timeout, aunque en esta ocasión mucho más extenso.

Cada thread deberá de recolectar diversos elementos asociados a su URL (Tabla 3.5) que se contabilizarían en:

- **Número de busquedas/queries:** Define el número de queries distintas sobre las que se buscarán resultados.
- **Parámetros SEO (Principales):** 39
  - On-page: 35
    - On-page estáticos (no cambian según la búsqueda): 23
    - Relacionados a keywords: 12
  - Off-page: 4
- **Parámetros SEO (Secundarios):** Número de keywords * 24
- Misceláneo: 5

[4]Esqueleto del análisis directo: URL = https://google.com/search?q= + query + &num=10 + &start= numero-de-URLs-ya-obtenidas

Global On-page	Global Off-page	Keywords (On-page)	Miscelaneo
Tamaño URL	Antigüedad del dominio	Densidad Keywords Totales	URL
Tamaño título	Enlaces Externos	Densidad Keywords Parciales	Título
Tamaño descripción	Enlaces Externo Follow	Cantidad Keywords Totales	Descripción
Uso de HTTPS (protocolo)	Enlaces Externo NoFollow	Cantidad Keywords Parciales	Fecha de Creación
Amigabilidad del URL		Posicionamiento Keywords Totales	Ranking
Uso cabeceras (h1-h6)		Posicionamiento Keywords Parciales	
Enlaces Totales			
Enlaces Internos			
Enlaces Interno Follow			
Enlaces Interno NoFollow			
Imágenes			
Vídeos			
Datos estructurados			
Marcado Schema			
Compatibilidad móvil			
Velocidad de carga Sobremesa			
Velocidad de carga Móvil			
Rendimiento Sobremesa			
Rendimiento Móvil			
Errores 404			
Errores de conexión			
Uso de robots.txt			
Uso de sitemap.xml			

Tabla 3.5: Listado de elementos a extraer

Fuente: Elaboración propia.

**Recordatorio: existen instancias de parámetros Keywords (On-page) para las etiquetas: <title>, <description>, <body>, <h1>, <alt>, <src>*

Como se puede observar, los únicos parámetros que no han sido incluidos son algunos de los que pertenecen a la categoría off-page. Debido a su naturaleza basada en la opinión de agentes externos, su extracción no ha sido considerada demasiado compleja (o imposible dentro de los límites del trabajo) y por lo tanto ha sido descartada en el desarrollo del programa.

A partir de los datos obtenidos, se elaborarán cuatro tablas para cada query las cuales serán ilustradas dentro de una de las página de un archivo xslx. Las tablas por representar serán las siguientes:

- **Tabla Principal:** Muestra los parámetros SEO de un numero especificado de resultados para una query concreta.

- **Tabla Secundaria:** Muestra los parámetros SEO secundarios (Marcados en rojo en la Tabla 3.3) para cada keyword de una query concreta y un numero especificado de búsquedas. No se utilizará para el cálculo de valores recomendados ni se tendrá en cuenta para el clasificador, su uso es puramente ilustrativo y sirve como complemento de la tabla principal.

- **Tabla de Medias:** Tabla de una sola línea en la que se muestran las medias de los parámetros SEO de **una** query para un número especificado de resultados.

- **Tabla de Medias Superior:** Tabla de una sola línea en la que se muestran las medias de los parámetros SEO de **una** query para un número especificado de búsquedas, pero en esta ocasión está delimitado por la opción *Ranking inferior límite* en lugar de *Numero de resultados*. Estas medias afectarán considerablemente a las recomendaciones y al clasificador.

3.2.3. Generación de informe y resultados globales

Una vez ya se hayan completado la extracción de todas las queries y sus tablas hayan sido creadas, se procederá a crear una nueva página que actuará a modo de resumen e incluirá las siguientes tablas:

- **Tabla de Medias Global:** Tabla de una sola línea en la que se muestran las medias de los parámetros SEO de **todas** las query para un número especificado de resultados.

- **Tabla de Medias Superior Global:** Tabla de una sola línea en la que se muestran las medias de los parámetros SEO de **todas** las query para un número especificado de búsquedas, pero en esta ocasión está delimitado por la opción *Ranking inferior límite* en lugar de *Numero de resultados*. Estas medias afectarán considerablemente a las recomendaciones y al clasificador.

En caso de haber seleccionado *Comparar resultados con una URL*, una tabla principal y una tabla secundaria de una solo búsqueda (la URL a comparar) también aparecerán en la misma página.

Finalmente, solo quedaría la creación de la página informe, la cual mostrará los pesados, los valores recomendados de cada variable y, además, en caso de que se desee comprar una URL, se mostrará dos instancias de una puntación y una estimación de su posicionamiento para las keywords seleccionadas.

En el caso de los valores recomendados, los parámetros con resultados binarios (uso de https, amigabilidad del URL, datos estructurados, marcado Schema, compatibilidad móvil, uso de robots.txt y uso de sitemap.xml) serán recomendados como necesarios si casi la totalidad de las URLs las incorporan o si las URLs en posiciones altas (Tabla de Medias Superior) las utilizan en mayor medida. En el resto de los parámetros, el rango de valores recomendado se basará en los resultados de la Tabla de Medias Superior Global y de la desviación típica encontrada entre las diferentes Tabla de Medias Superior (locales).

Como se explicó en el capítulo anterior, el algoritmo de clasificación estudiado para este trabajo es Gradient Boosting. Se implementarán dos clasificadores para el estudio de parámetros, siendo entrenados, en ambos casos, por todos los resultados de todas las queries buscadas, diferenciándose únicamente en su división de clases. Uno de ellos, creará múltiples clases (hasta un máximo de 5) dependiendo del número de búsquedas, perteneciendo a la primera clase solo aquellas URLs con posicionamiento menor al *Ranking inferior límite* y la segunda clase aquellas en el intervalo [*Ranking inferior límite*-10]. En el caso del segundo clasificador, solo se dispondrá de dos clases, una para aquellos URLs en el Top 10 y otra para aquellos fuera del top. El segundo clasificador fue creado para ofrecer una mayor precisión (al coste de generalizar) en el caso de que los resultados del primer clasificador sean muy negativos.

Cada uno de los clasificadores ofrecerá tres tipos de resultados:

- Una serie de pesados que estima la importancia de cada parámetro SEO y que constituye la base de la toma de decisiones (predicciones) del clasificador.

- La estimación de la posición de una URL (si se seleccionó *Comparar resultados con una URL*).

- La creación de una puntuación sobre 100 en base al nivel de error de los parámetros SEO de una URL en comparación a los valores recomendados y al pesado de importancia. A mayor importancia mayor será su efecto en la puntuación (si se seleccionó *Comparar resultados con una URL*).

3.2.4. Pruebas a realizar

Este trabajo consiste en la automatización de la extracción y el análisis de parámetros SEO. Para asegurarse de que el código hace honor al título, este deberá pasar una serie de pruebas para evaluar su funcionalidad y grado de complexión de los objetivos planteados al inicio del trabajo.

Al final del trabajo se probará la capacidad del código para:

- **Extracción de parámetros SEO:** Básicamente se tratará de comprobar el correcto funcionamiento del programa, tanto su capacidad de extracción como la presentación de los resultados. (Objetivo principal 1) (Objetivo secundario 3)
- **Valorar parámetros y clasificar webs:** Se comprobará la precisión del clasificador en el pesado de parámetros y clasificación de webs. Además, se mostrara las conclusiones obtenidas sobre el pesado de cada parámetro. (Objetivo principal 2 y 3) (Objetivo secundario 2)
- **Recomendación de parámetros:** Fuertemente ligado al anterior. Se probará si es posible la obtención valores recomendables para cada parámetro de manera global. (Objetivo secundario 2)
- **Análisis de keywords:** Se observará si los parámetros relacionados con las keywords extraídos por el programa (densidad, cantidad y posicionamiento) ayudan al posicionamiento en buscadores. (Objetivo secundario 1)

CAPÍTULO 4
Tecnología Utilizada

Con anterioridad al desarrollo de este trabajo, mi conocimiento sobre el SEO era muy escaso, especialmente en lo que se refiere a programación dentro de dicho contexto. Debido a que tuve que partir de cero, las horas utilizadas en la búsqueda de distintas herramientas que permitan o faciliten la creación del programa no fueron pocas.

En esté capitulo se detallarán las distintas herramientas y tecnologías usadas para la elaboración, porqué han sido elegidas y de que manera son utilizadas.

4.1 Lenguaje de programación

El lenguaje de programación utilizado para el desarrollo de la aplicación ha sido Python como se indica en el título, más concretamente Python 3.7.0.

Existen diversos motivos por los cuales se eligió este lenguaje nada más comenzar el trabajo:

- Se trata de un lenguaje interpretado, dinámico y multiplataforma.

- Cuenta con una comunidad muy grande y activa, la cual aumentan día a día la versatilidad del lenguaje mediante la creación de nuevos módulos, extensiones y librerías.

- Según la encuesta realizada en 2021 en *Jet Brains* [18], Python no solo es considerado el lenguaje más popular (en cuanto a usuarios totales), sino que también es considerado uno de los lenguajes con mayor crecimiento.

- La flexibilidad del lenguaje. Python presenta una sintaxis y una serie de herramientas que facilita la creación de un código limpio y comprensible. Destacando, para este trabajo, la flexibilidad a la hora de crear listas y matrices.

- Experiencia personal previa con el lenguaje. Debido a mi inexperiencia dentro del contexto SEO, el uso de un lenguaje familiar reduce considerablemente el tiempo de estudio necesario.

Además, cabe mencionar que se eligió la versión 3.7.0 ya que era necesaria para el correcto funcionamiento de todas las librerías utilizadas por el programa.

4.2 Entorno de desarrollo y gestión de dependencias

Se han utilizado tres entornos distintos para el desarrollo del programa:

- **Visual Studio Code:** Editor de código fuente usado para la creación del código completo desarrollado. Los principales motivos por el que se usó dicho editor fueron debido a mi experiencia previa con él y su capacidad de linting y debugging.
- **Jupyter Notebook:** Entorno de desarrollo web dividido en celdas el cual permite la ejecución individual de cierta sección del código. Utilizada para comprobar el funcionamiento individual de cada función del programa desarrollado.
- **Anaconda:** Programa que permite la instalación y gestión de distintos entornos, cada uno con su propia colección de librerías y dependencias instaladas. Utilizada dentro del desarrollo para la creación de un entorno especialidad y la instalación de todas las librerías externas utilizadas.

El motivo para desarrollar el código en múltiples entornos fue únicamente para facilitar la gestión de este, dividiendo las tareas en elaboración de código, testeo de módulos y gestión de librerías.

4.3 Librerías y APIs

Durante el desarrollo y creación del programa se tuvo que hacer uso de una gran variedad de librerías, muchas de ellas independientes al núcleo de Python.

A continuación se listarán las librerías usadas más esenciales:

Librerías generales

- **Bs4:** Versión 4.0 de Beautiful Soup. Usado para el análisis de archivos HTML, facilitando la búsqueda y extracción de sus distintos elementos.
- **Urllib:** Librería estándar de Python usada para la extracción, acceso y manejo de URLs
- **Tldextract:** Librería para la extracción de segmentos específicos de una URL.
- **Requests:** Basada en el modulo request de urllib, requests facilita el envió de peticiones a distintas webs.
- **sklearn:** Como se describió en el capítulos 2, el algoritmo de aprendizaje automático utilizado para el programa es Gradient Boosting. Más concretamente la librería desarrollada por *Scikit learn*, la cual se encarga de obtener el pesado de cada uno de los parámetros SEO.
- **NumPy:** Librería que facilita el manejo de listas y tablas, utilizada para gestión los datos de entrenamiento y las etiquetas del clasificador, así como para calcular la varianza encontrada en cada uno de los parámetros SEO, algo que modificará los valores recomendados.
- **Xlsxwriter:** Permite la creación de un archivo xslx utilizado por programas como Excel. Usado para representar todos los resultados obtenidos de una manera clara y ordenada.

- **Threading:** Librería estándar de Python usada para el paralelismo basado en threads. Permite la extracción parámetros SEO para múltiples URLs de manera simultánea, reduciendo considerablemente el tiempo de ejecución.
- **Tkinter:** Librería estándar de Python para la creación de interfaces gráficas.

Librerías y APIs para parámetros concretos

- **WHOIS:** Librería que permite el acceso a los datos de la API de WHOIS. En nuestro caso, entre dichos datos solo se precisa del la fecha de creación.
- **Google APIs:** Uso de las APIs de Google para obtener la compatibilidad móvil(MobileFriendlyTest API) por un lado, y la puntuación y rendimiento para dispositivos móviles y sobremesa (PageSpeed Insights API). Son accedidas mediante su URL correspondiente[1] de la cual obtendremos un archivo en formato JSON con el valor de los parámetros SEO correspondientes. Precisa del uso de la librería JSON (librería estándar de Python).

[1] MobileFriendlyTest API URL: https://searchconsole.googleapis.com/v1/urlTestingTools/mobileFriendlyTest:run
PageSpeed Insights API URL: https://www.googleapis.com/pagespeedonline/v5/runPagespeed

CAPÍTULO 5
Desarrollo del Programa

En este capítulo se describe el funcionamiento, los problemas y dificultades encontrados durante el desarrollo de cada uno de los segmentos del código desarrollado según el diseño final planteado en la sección 3.2, centrándose, sobre todo, en los algoritmos utilizados para la extracción y clasificación de los resultados. También, aunque de una manera más superficial, se explicará la estructura de la interfaz gráfica y de la generación de resultados.

Para facilitar la comprensión del lector, este capítulo ha sido estructurado de manera similar a como se ejecuta el programa.

5.1 Interfaz Gráfica

Como se mencionó anteriormente, para facilitar el lanzamiento del programa se ha creado un archivo ejecutable. Desafortunadamente, debido a errores en el empaquetamiento de la biblioteca *sklearn*, no se ha podido comprimir el ejecutable en un archivo único, usando por consiguiente un directorio, el cual contiene el ejecutable, sus dependencias, y los archivos generados por el código.

Tras lanzado el programa, se mostrará una interfaz gráfica con la siguiente estructura:

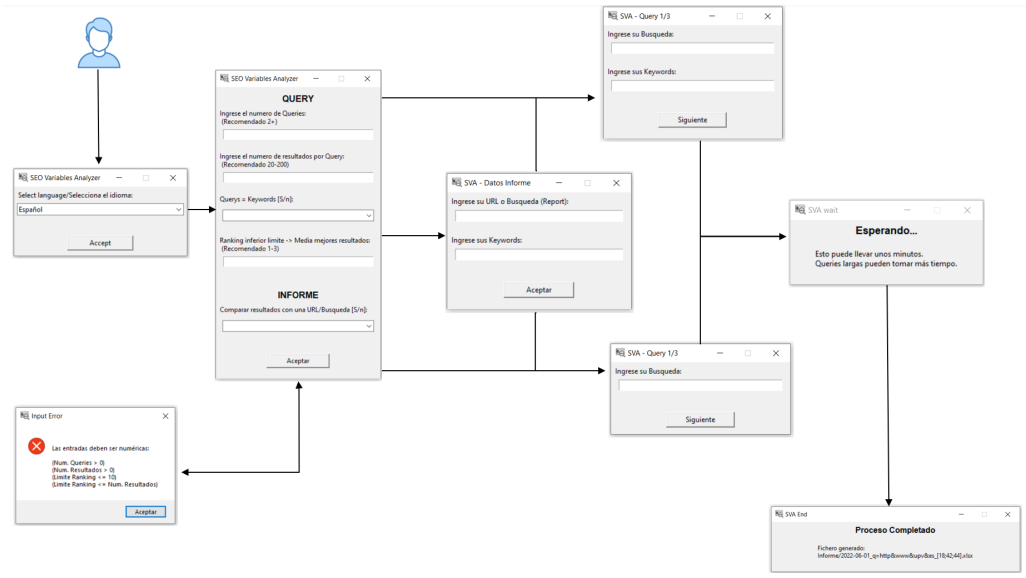


Figura 5.1: Interfaz gráfica del programa desarrollado

Fuente: Elaboración propia.

Al iniciar la interfaz, lo primero que nos encontramos es un selector de idiomas donde, actualmente, solo están disponibles las opciones de inglés y español. El valor cambiará el lenguaje tanto de la interfaz como el de los archivos generados.

Una vez seleccionado el lenguaje se abrirá el selector de parámetros que incluye todas las opciones mencionadas en la sección 3.2, incluyéndose una serie de recomendaciones para la ejecución óptima del programa.

El número de queries se ha recomendado como mayor o igual a 2, debido a que si se realiza una búsqueda de una única query, sería imposible calcular de manera precisa el peso de cada variable SEO, por estar extremadamente sesgado a esta query en particular.

En el caso de las búsquedas, se aconseja usar un mínimo de 20 para asegurar tener información suficiente de queries con posiciones (moderadamente) malas con las que comparar a las del Top 10. Además, **se recomienda un máximo de 200 ya que, en ocasiones, el buscador de Google no nos proporciona los suficientes resultados**. El programa se ejecutaría sin problemas, pero aumentaría falsamente la precisión del clasificador y las tablas estarían desequilibradas en cuanto número de resultados.

Respecto al *límite inferior* (o *ranking inferior límite*), se recomienda no pasar de 3 ya que este valor representara las posiciones consideradas como un modelo a seguir (recordemos que estudios los usuarios no suelen mirar más allá de los primeros tres resultados [3]) y serán las que más afecten a los valores recomendados generados.

Para evitar búsquedas vacías o ilógicas, el programa lanzará una ventana de error (esquina inferior izquierda de la Figura 5.1) conteniendo los límites establecidos, tras la cual se podrán reintroducir los datos.

Al introducir los parámetros correctamente, siguiendo el diseño del capítulo 3, se habrá que introducir la URL (o búsqueda) y keywords a comparar en el informe (opcional), y las queries (y las keywords si se quieren introducir manualmente) de manera secuencial.

Finalmente, la GUI simplemente mostrará un mensaje de espera mientras se ejecuta el programa, y uno de finalización al terminar.

5.2 Threading y extracción de URLs

Una vez introducidos todos los datos en la interfaz gráfica, el código comenzará el proceso de extracción de parámetros SEO para las queries y número de resultados seleccionado. Al no ser posible la extracción de todas las búsquedas de manera simultánea (debido al riesgo de obtener el error 429), se ha seguido el siguiente procedimiento basado en threads:

1. Se seleccionarán hasta un máximo de 5 queries sin analizar.

2. Se extraerá, desde la página de Google Search, las URLs correspondientes al Top *Número de búsquedas*[1] de cada una de las queries seleccionadas. El proceso se hará de manera concurrente (mediante threads).

3. Una vez obtenidas las URLs, comenzará el procesó de extracción de sus parámetros SEO, pero para evitar el error 429, las extracciones se lanzarán en bloques de **50 threads** entre todas las queries seleccionadas[2].

[1] Segunda variable introducida en la interfaz

[2] 1 query = 50 threads/queries, 2 queries = 25 threads/queries , 3 queries = 16 threads/queries, 4 queries = 12 threads/queries, 5 queries = 10 threads/queries

4. Se espera a que finalice el bloque de threads, el cual posee un tiempo límite de 10 minutos

5. Se continuarán lanzando bloques de 50 threads (máximo) hasta que no queden URLs a analizar en las queries seleccionadas.

6. Si quedan queries: seleccionarán las siguientes 5 (máximo) y volver al paso 2. En caso contrario: fin de la extracción.

Es importante mencionar qué para asegurarnos que obtenemos resultados objetivos, las búsquedas en Google Search son realizadas con una serie de cabeceras que evitan que sean afectadas por la cache. En caso de que se desee observar el código de obtención de URLs, consultar la figura B.1 en los anexos.

5.3 Módulos para la extracción de parámetros SEO

Teniendo en cuenta que el algoritmo de posicionamiento de Google cuenta con, posiblemente, cientos de parámetros a tener en cuenta, uno de los factores de calidad más importantes (sino el más importante) de los programas de análisis SEO, es el número de parámetros extraídos y representados. En consecuencia, se ha visto necesario detallar y dar constancia del proceso de extracción utilizado para cada uno de los parámetros obtenidos en el programa.

5.3.1. Módulos externos al HTML

A continuación, se detallará la obtención de los parámetros que no requieren el acceso directo al HTML de la URL a analizar y que, por lo tanto, no suelen generar ningún tipo de problema a la hora de extraerlos.

Longitud URL

Uno de los parámetros más sencillos de obtener ya que, normalmente, la longitud en SEO hace referencia al número de caracteres, número que es convenientemente devuelto cuando se mide un String con la función de Python básica *len()*. (Longitud URL = len(URL)).

Protocolo de la URL

Similar al anterior en cuanto a complejidad. Basta con obtener el atributo *scheme* de la función *urlparse()* perteneciente a la librería *urllib*. (Protocolo = urlparse(URL).scheme).

Amigabilidad de la URL

Debido a que no se encontró ninguna herramienta que obtenga, de una manera más directa, la amigabilidad de una URL, se tuvo que crear en base a los consejos de Google [19], una implementación muy básica para dicha tarea. En esta se comprueban que no se utilicen símbolos comúnmente considerados extraños, como los mostrados en la línea 14 de la figura 5.3, y también se comprueba que la URL no incorpore una query o un fragmento que alarguen innecesariamente la URL (ya que solo se trata de un filtrado). El uso del módulo *urlparse* de *urllib* para seccionar de manera precisa la URL ha facilitado enormemente la tarea.

```
def urlAmigable(link):

    """
    Num ejecuciones: Num Queries * Num Resultados + 1

    Decide a partir de un listado basico de simbolos y de si incorpora segmentos concretos de una URL el si la
     URL especificada
    es amigable o no,

    param:  "link": URL a analizar en crearTabla.

    return: "friURL": Indica si la URL es amigable en formato string (Yes/No).
    """

    noAmigable=['_', '&', '.','=','@','#','$','%','?','!',';',':','+','*']
    Path=str(urlparse(link).path)
    if any([x in Path for x in noAmigable]):
        #Url: No amigable
        return "No"
    else:
        if(not(urlparse(link).query == "" and urlparse(link).fragment == "")):
            #Url = No amigable
            return "No"
        else:
            #Url = Amigable
            return "Yes"
```

Figura 5.3: Obtención de la amigabilidad de una URL

Fuente: Elaboración propia.

Fecha de creación

La obtención de este parámetro es bastante sencillo y directo como se ve en la figura 5.5. La fecha de creación de la web es obtenida de manera directa mediante una petición a un servidor de WHOIS (línea 17).

Para obtener el número de días solo es necesario obtener la diferencia entre la fecha actual con la obtenida (linea 20).

Desafortunadamente, a pesar de poseer una implementación sencilla, es, **posiblemente el parámetro que más fallos de extracción presenta**, ya sea por un error de conexión con la API de WHOIS, o porque casualmente no está disponible esa información en los servidores.

```
def creacion(link):

    """
    Num ejecuciones: Num Queries * Num Resultados + 1

    Obtiene los Days Alive y Fecha de Creaccion de una URL especifica a partir de los servidores de WHOIS.

    param:  "link": URL a analizar en crearTabla.

    return: "CDate": Fecha de Creacion de la URL.
            "CAlive": Dias desde la creacion de la URL.
    """

    hoy=datetime.datetime.now().date()

    try:
        wis = whois.whois(link)
        if isinstance(wis.creation_date, list):
            wisd=wis.creation_date[0].date()
            wisdold=hoy-wisd
        elif wis.creation_date is None:
            #"Creacion: Desconocido"
            #"Dias desde creacion: Desconocido"
            return "Unknown", "Unknown"
        else:
            wisd=wis.creation_date.date()
            wisdold=hoy-wisd
    except Exception:
        return "Unknown", "Unknown"

    #Creacion = wisd
    #Dias desde creacion = wisdold.days
    return str(wisd), str(wisdold.days)
```

Figura 5.5: Obtención de la edad de una web

Fuente: Elaboración propia.

Robots.txt y Sitemaps.xml

La búsqueda de un archivo de robots.txt es algo sencillo. Solo hay que comprobar que existe un archivo con ese nombre dentro del directorio raíz (siempre se encuentra aquí). Debido a que en muchas ocasiones las direcciones de los sitemaps se encuentran referenciados dentro de robots.txt, lo primero que se intenta en el módulo mostrado en la figura 5.7 es buscar coincidencias dentro de él (línea 45). Desgraciadamente para nuestro programa, los webmasters disponen de distintas herramientas para ocultar sus sitemaps. En caso de que no se encuentre ninguna coincidencia ya sea porque no poseen dicho documento o porque está ocultado, nuestro programa hará un último intento buscándolo en el directorio raíz (línea 50), y en caso de no encontrar resultados, se asumirá que el programa no posee sitemaps.

```
1  def robotsNsitemaps(link, scheme, headers):
2
3      """
4      Num ejecuciones: Num Queries * Num Resultados + 1
5
6      Busca en la raiz de la URL si existe robots.txt y busca sitmap.xml tanto en robots.txt como en la raiz.
7
8      param:  "link": URL a analizar en crearTabla.
9              "scheme": Protocolo de la URL a analizar (http/https normalmente).
10             "headers": Encabezados utilizados para que los requests no den problemas.
11
12     return: "IsRob": Indica si se ha encontrado robots.txt en la url en formato string (Yes/No/Unknown).
13             "IsSite": Indica si se ha encontrado sitemap.xml en la url en formato string (Yes/No/Unknown).
14     """
15
16     #Busqueda Robots.txt
17     robStr=""
18     IsRob="No"
19     try:
20         gdesRob= scheme + "://" + urlparse(link).netloc + "/robots.txt"
21         responseg = requests.get(gdesRob, headers=headers, timeout=10)
22         status_codeR = responseg.status_code
23
24         if status_codeR >= 400:
25             isRob="No"
26             #Robots.txt = No
27         else:
28             #Robots.txt = Si
29             IsRob="Yes"
30             robStr=str(BeautifulSoup(responseg.content, "html.parser"))
31     except requests.exceptions.RequestException:
32         #Robots.txt: Timeout error
33         IsRob="Unknown"
34
35
36     #Busqueda Sitemap.xml
37     try:
38         gdesSmap= scheme + "://" + urlparse(link).netloc + "/sitemap.xml"
39         responseg = requests.get(gdesSmap, headers=headers, timeout=10)
40         status_codeS = responseg.status_code
41     except requests.exceptions.RequestException:
42         status_codeS=0
43
44     if "Sitemap:" in robStr:
45         robStr=robStr.count("Sitemap:")
46         #Sitemap.xml: Si, cantidad = robStr
47         return IsRob, "Yes (" + str(robStr) + ")"
48     elif status_codeS < 400:
49         #Sitemap.xml: Si
50         return IsRob, "Yes (1)"
51     else:
52         #Sitemap.xml: No
53         return IsRob, "No"
```

Figura 5.7: Búsqueda de archivos Robots.txt y Sitemaps.xml

Fuente: Elaboración propia.

Rendimiento y Velocidad de carga

Este módulo muestra como el programa realiza una petición a Google PageSpeed Insights API para dispositivos móviles y para dispositivos sobremesa. Para ser exactos, lo que nos devuelve son los *Core Web Vitals* de la web para el dispositivo correspondiente, que son filtrados por el código para obtener el rendimiento y la velocidad de carga.

A pesar de que la API devuelve varias métricas para analizar la velocidad de carga de una web, se ha decidido extraer exclusivamente lo que se conoce como *Largest Contentful Paint*[3] (LCP) debido a que es la de mayor peso (25 %) de entre todas las métricas [20].

```
def score(link):

    """
    Num ejecuciones: Num Queries * Num Resultados + 1

    Se obtiene la Velocidad de Ejecucion (LCP) y el Puntuaci n de Rendimiento a paritr de la informacion
     obtenida de googleapis runPageSpeed.

    param:  "link": URL a analizar en crearTabla.

    return: "mop": Indica en un valor del 0-100 la Puntuaci n de Rendimiento en dispositivos moviles de la web
      de la URL especificada.
            "mls": Velocidad de Ejecucion en segundos en dispositivos moviles para la web de la URL
     especificada.
            "dop": Indica en un valor del 0-100 la Puntuaci n de Rendimiento en dispositivos sobremesa de la
     web de la URL especificada.
            "dls": Velocidad de Ejecucion en segundos en dispositivos sobremesa para la web de la URL
     especificada.

    """

    #Google Api runspeed Links
    urlMobil = "https://www.googleapis.com/pagespeedonline/v5/runPagespeed?url=" + link  + "&strategy=mobile&
     locale=es&key=*********" #Key ocultada por privacidad
    urlDesktop = "https://www.googleapis.com/pagespeedonline/v5/runPagespeed?url=" + link  + "&strategy=desktop
     &locale=es&key=*********" #Key ocultada por privacidad

    #Datos Mobil
    try:
        responseMobil = urlopen(urlMobil)
        dataMobil = json.loads(responseMobil.read().decode('utf-8'))
        overall_scoreMobil = dataMobil["lighthouseResult"]["categories"]["performance"]["score"] * 100
        overall_scoreMobil=int(overall_scoreMobil)
        lcpMobil = dataMobil["lighthouseResult"]["audits"]["largest-contentful-paint"]["displayValue"]
        #Rendimiento movil = overall_scoreMobil
        #Velocidad de carga movil = lcpMobil
    except Exception:
        overall_scoreMobil = "Unknown"
        lcpMobil = "Unknown"


    #Datos Desktop
    try:
        responseDesktop = urlopen(urlDesktop)
        dataDesktop = json.loads(responseDesktop.read().decode('utf-8'))
        overall_scoreDesktop = dataDesktop["lighthouseResult"]["categories"]["performance"]["score"] * 100
        overall_scoreDesktop = int(overall_scoreDesktop)
        lcpDesktop = dataDesktop["lighthouseResult"]["audits"]["largest-contentful-paint"]["displayValue"]
        #Rendimiento Sobremesa = overall_scoreDesktop/100
        #Velocidad de carga sobremesa = lcpDesktop
    except Exception:
        overall_scoreDesktop = "Unknown"
        lcpDesktop = "Unknown"

    return overall_scoreMobil, lcpMobil, overall_scoreDesktop, lcpDesktop
```

Figura 5.9: Obtención del rendimiento y velocidad de carga de una web

Fuente: Elaboración propia.

[3]LCP: Métrica que marca el tiempo en el que se pinta la pintura o texto más grande

Compatibilidad móvil

Como se puede observar en la figura 5.11, la estructura seguida por este módulo es muy similar al previo. Al igual que en el caso anterior, se hace una llamada a la API de Google correspondiente, en este caso *mobileFriendlyTest*. Dicha API, al mandarle una petición a través de su URL, devolverá, entre otras cosas (las cuales se filtran), si Google considera esa web apta para dispositivos móviles.

Cabe mencionar que, aunque poco probable, la API no es siempre capaz de obtener un resultado, incluso si la web de la URL a analizar esta completamente funcional (al contrario del módulo anterior).

```
def compMovil(link):

    """
    Num ejecuciones: Num Queries * Num Resultados + 1

    Comprueba si la web de la URL es amigable para un entorno movil a partir de googleapis.

    param:  "link": URL a analizar en crearTabla.

    return: "compM": Indica si la URL es amigable para moviles en formato string (MOBILE_FIRENDLY/
     No_MOBILE_FRIENDLY/Unknown).
    """

    MFurl = 'https://searchconsole.googleapis.com/v1/urlTestingTools/mobileFriendlyTest:run'
    MFparams = {

                'url': link,

                'key': "*********" #Key ocultada por privacidad

            }
    try:
        MFx = requests.post(MFurl, data = MFparams)
        MFdata = json.loads(MFx.text)
        if MFdata["testStatus"]["status"] == "PAGE_UNREACHABLE":
            #Error Compatibilidad Moviles: MFdata["testStatus"]["status"]
            return "Unknown"
        else:
            #Compatibilidad Moviles: = MFdata["mobileFriendliness"]
            return MFdata["mobileFriendliness"]
    except Exception:
        return "Unknown"
```

Figura 5.11: Comprobación de la compatibilidad móvil de una web

Fuente: Elaboración propia.

5.3.2. Módulos internos al HTML

En esta ocasión, al contrario que en el apartado anterior, se detallará la obtención de parámetros que hagan uso del archivo HTML de la URL analizar.

En ocasiones, la extracción de esta colección de parámetros no es posible debido a algún tipo de error en la conexión con el servidor, pero, afortunadamente, debido a que eso de por sí solo afecta al posicionamiento de una web, no se suele encontrar este problema en los primeros resultados de una búsqueda. En los peores casos que he experimentado, este error podría llegar a encontrarse en una décima parte de los resultados.

Longitud meta título/descripción

Durante el desarrollo inicial de este módulo se encontró un problema que complicó notablemente su desarrollo. A pesar de que se extraían sin muchos problemas el meta título y la meta descripción a partir del HTML de la web, estos no siempre correspondían a los que se mostraban en la página de Google Search.

Debido a este problema, se tuvo que modificar el código hasta obtener el que se ve en la figura 5.13. El módulo en cuestión busca a través del HTML de Google Search donde se

obtuvo el URL a analizar. Partiendo desde la posición desde donde se encontró el URL, se buscarán una serie de etiquetas y clases (propias de Google Search) que corresponden a títulos y descripciones.

En caso de que no se encuentren, se extraerán mediante el método original, buscando etiquetas correspondientes al meta título y meta descripción dentro del URL a analizar. Esta búsqueda auxiliar puede o puede no dar la respuesta correcta, pero seguiría siendo mejor que una respuesta en blanco, ya que, aunque no sea lo que se muestre en Google, técnicamente el HTML los considera como el título y la descripción correctas.

Una vez obtenidos el título y la descripción, solo hace falta del uso de la función *len()* para sacar la longitud de cada uno.

```
def MetaTD(glink, link, headers):

    """
    Num ejecuciones: Num Queries * Num Resultados + 1

    Se obtiene el Meta Titulo y Meta Descripci n mostrados en Google Search ademas de sus longitudes para una
     URL especificada.

    param:  "glink": URL de la pagina de Google search en donde aparece la URL a analizar.
            "link": URL a analizar en crearTabla.
            "headers": Encabezados utilizados para que los requests no den problemas.

    return: "titulo": Muestra en formato String el Meta Titulo de la URL especificada.
            "tituloL": Muestra la longitud en car cteres del Meta Titulo.
            "descripcion, descripcionL": Muestra en formato String la Meta Descripci n de la URL especificada.
            "descripcionL": Muestra la longitud en car cteres de la Meta Descripci n.

    """

    #Meta Titulo & Logitud
    #Longitud = len(tit)
    Palabras = len(tit.split())
    tit=""
    gdes2=glink.find_all("h3", class_="LC20lb MBeuO DKV0Md")
    if(len(gdes2) >=1):
        tit=gdes2[-1].get_text()


    #Meta Descripcion & Logitud
    text=""
    gbool=True
    gdes2=glink.find_all("div", class_="VwiC3b yXK7lf MUxGbd yDYNvb lyLwlc lEBKkf")
    if(len(gdes2) >=1):
        gdes3=gdes2[-1].find_all("span")
        if(len(gdes3) >=1):
            gdes3 = str(gdes3[-1])
            if not gdes3.startswith("<span class"):
                gbool=False
                gdes3=gdes3.replace('<span>','')
                gdes3=gdes3.replace('</span>','')
                gdes3=gdes3.replace('<em>','')
                gdes3=gdes3.replace('</em>','')
                text=gdes3
        else:
            gbool=False
            gdes3=gdes2[-1].get_text()
            text=gdes3

    #Busqueda a la fuerza en caso extremo
    if(gbool):
        try:
            responseg = requests.head(link, headers=headers, timeout=10)
            status_code = responseg.status_code
            if status_code < 400:
                htmlg =  requests.get(link, headers=headers, timeout=10)
                soupg1 = BeautifulSoup(htmlg.content, features="html.parser")
                metas = soupg1.find_all('meta') #Get Meta Description
                for m in metas:
                    if m.get ('name') == 'description':
                        text = m.get('content')
        except Exception:
            pass
            #"Fallo en la obtencion de la Descripcion"

    return str(tit), len(tit), str(text), len(text)
```

Figura 5.13: Extracción de Meta Título/Descripción

Fuente: Elaboración propia.

Cantidad de Imágenes y Vídeos

Obtener la cantidad de imágenes y vídeos es una tarea extremadamente sencilla gracias a la librería *Beautiful Soup*. Gracias a su módulo **find_all()** (utilizado en una gran cantidad de módulos del código), que obtiene todas las instancias de un tipo de etiqueta (o clase) dentro de un HTML, y gracias a que tanto las imágenes como los vídeos hacen uso de etiquetas características (<img>y <video>respectivamente), solo se ha necesita hacer dos llamadas a dicho modulo para obtener los valores de estos parámetros.

Encabezados

Este módulo trata de obtener un listado de que tipos de cabecera incorpora la web de la URL a analizar. Para ello solo habrá que comprobar, uno por uno, si hay instancias de ellos dentro del objeto **soup** utilizado en la figura 5.15, el cual guarda de manera estructurada el HTML de la web.

```
def encabezados(soup):

    """
    Num ejecuciones: Num Queries * Num Resultados + 1

    Obtiene los headers (h1-h6) utilizados para la web de la URL espec ficada.

    param:  "soup": Sopa en la que se gurda la informaci n HTML de la URL.

    return: "encabezado": Lista de los encabezados usados.
    """

    #Comprobacion Exisistencia h1-h6
    encabezados = []
    if(not(soup.h1 == None)):
        encabezados.append("h1")
    if(not(soup.h2 == None)):
        encabezados.append("h2")
    if(not(soup.h3 == None)):
        encabezados.append("h3")
    if(not(soup.h4 == None)):
        encabezados.append("h4")
    if(not(soup.h5 == None)):
        encabezados.append("h5")
    if(not(soup.h6 == None)):
        encabezados.append("h6")

    #Encabezados usados = encabezados
    return str(encabezados)
```

Figura 5.15: Obtención de encabezados (h1-h6) usados por la web

Fuente: Elaboración propia.

Datos estructurados y marcado Schema

Como se mencionó anteriormente, los datos estructurados son simplemente un conjunto de vocabularios y estructuras que se pueden incluir en un HTML para facilitar a Google a comprender como está estructurado el contenido. Debido a esto, en el módulo representado por la figura 5.17, se busca la serie de caracteres (String) que representa la inicialización de cada tipo de datos estructurados, además de la posible mención a *schema.org*, ya que en caso de usar el vocabulario propio de dicha web (marcado Schema), su enlace estará incluido en los datos estructurados.

Debido a la naturaleza de este parámetro, es difícil encontrar una manera más exacta de extraerlo. Aunque es cierto que siempre podrá encontrar los datos estructurados en caso de que se usen, existe una posibilidad muy baja (pero existente) de que una web incorporé los Strings buscados fuera de los datos estructurados, clasificando erróneamente dicha web.

```
def DatosEstructurados(soup):

    """
    Num ejecuciones: Num Queries * Num Resultados + 1

    Busca el tipo de Datos Estructurados de la web de la URL y si utiliza Markado Schema.

    param:  "soup": Sopa en la que se gurda la informaci n HTML de la URL.

    return: "markup": Indica que tipo de Datos Estructurados tienen la web de la URL especificada (JSON-LD/
     Microdata/RDFa/No).
            "mschema": Indica si se utiliza Markado Schema en la web en formato string (Yes/No).
    """

    #Tipo de Datos Estructurados
    strSoup=str(soup).lower()
    strDS = "Datos estructurados:"
    strDSB=True
    if "application/ld+json" in strSoup:
        strDs= strDS + " JSON-LD"
    elif "itemscope itemtype" in strSoup:
        strDs= strDS + " Microdata"
    elif "vocab=" in strSoup:
        strDs= strDS + " RDFa"
    else:
        strDs= strDS + " No"
        strDSB=False

    #Marcado Schema
    if strDSB and "schema.org" in strSoup:
        #Tipo de Marcado Schema = strDs
        return strDs, "Yes"
    else:
        #Marcado Schema = No
        return strDs, "No"
```

Figura 5.17: Búsqueda de datos estructurados y marcado Schema

Fuente: Elaboración propia.

Links y Errores

El módulo utilizado para la extracción de enlaces y errores (Figura 5.19), es el módulo que más parámetros recolecta en una única ejecución. Los cuales se obtienen de las siguientes maneras:

- **Enlaces totales:** Se buscan todas las instancias de la etiqueta utilizada en HTML para referenciar enlaces (<a>), y dentro de la misma buscamos el valor del parámetro 'hreff, el cual corresponde con una URL.
- **Enlaces internos y externos:** Se extrae el dominio del HTML y de todas las URLs encontradas. En caso de que coincidan, serán consideradas como enlaces internos y en caso contrario como externos.
- **Enlaces Follow y NoFollow:** Dentro de la misma etiqueta utilizada para referenciar el enlace (<a>), se buscara el valor del parámetro 'rel', el cual indica si se quiere que los buscadores influyan en el ranking del enlace referenciado (Follow) o no (NoFollow).

A pesar de su utilidad, este es, sin lugar a dudas, el **módulo más lento de todo el programa**. Siendo el motivo el alto número de peticiones que se realizan a distintas URLs para comprobar su estado, llegando a tardar minutos en caso de que el HTML contenga cientos de URLs.

```
def linkCount(soup, link, headers):

    """
    Num ejecuciones: Num Queries * Num Resultados + 1

    Se obtiene el numero de links Totales/Internos(NoFollow y Follow)/Externos(NoFollow y Follow)
    y el numero de Errores (404 y por conexi n) encontrados en la web de la URL especificada.

    param:  "soup": Sopa en la que se gurda la informaci n HTML de la URL.
            "link": URL a analizar en crearTabla.
            "headers": Encabezados utilizados para que los requests no den problemas.

    return: "linksT": N mero de links Totales encontrados.
            "linksI": N mero de links Internos encontrados.
            "linksE": N mero de links Externos encontrados.
            "linksIF": N mero de links Internos con etiqueta Follow encontrados.
            "linksEF": N mero de links Externos con etiqueta Follow encontrados.
            "linksIN": N mero de links Internos con etiqueta NoFollow encontrados.
            "linksEN": N mero de links Externos con etiqueta NoFollow encontrados.
            "error404": N mero de Errores 404 encontrados en los links.
            "errorC": N mero de Errores por Conexi n encontrados en los links.

    """

    #Parametros iniciales
    DesIF=0
    DesEF=0
    DesIN=0
    DesEN=0

    extracted = tldextract.extract(link)
    extracted = "{}.{}".format(extracted.domain, extracted.suffix)

    urlPart = urlsplit(link)
    netl = urlPart.netloc
    schem = urlPart.scheme

    numerror=0
    timero=0


    for a in soup.find_all("a"):
        href=a.get('href')

        if(href == None): continue

        if(bool(urlparse(href).netloc)):
            extracted2 = tldextract.extract(href)
            extracted2 = "{}.{}".format(extracted2.domain, extracted2.suffix)

        #Calificaci n Externo/Interno Follow/NoFollow
        if(not bool(urlparse(href).netloc) or extracted == extracted2):
            try:
                if(a.get('rel') is None or a.get('rel')[0] != 'nofollow'):
                    DesIF=DesIF+1
                else:
                    DesIN=DesIN+1
            except IndexError:
                DesIF=DesIF+1
        else:
            try:
                if(a.get('rel') is None or a.get('rel')[0] != 'nofollow'):
                    DesEF=DesEF+1
                else:
                    DesEN=DesEN+1
            except IndexError:
                DesEF=DesEF+1


        if(urlparse(href).netloc == ''): href = schem + '://' + netl +href
        elif(urlparse(href).scheme == ''): href = schem + ':'  +href


        #Calificaci n Dead Links
        try:
            response = requests.head(href, headers=headers, timeout=10)
            status_code = response.status_code
            if 'redlink=1' in href or status_code == 404 :
                if 'redlink=1' in href or requests.get(href).status_code == 404:
                    numerror = numerror + 1
            elif status_code >= 400:
                timero = timero + 1
        except Exception:
            timero = timero + 1


    #Muestreo
    DesTot = DesEF+DesEN+DesIF+DesIN

    return DesTot, DesIF+DesIN, DesEF+DesEN, DesIF, DesEF, DesIN, DesEN, numerror, timero
```

Figura 5.19: Extracción de enlaces y errores de una web

Fuente: Elaboración propia.

Densidad, Cantidad y Posicionamiento de keywords

El código ilustrado en la figura 5.23 es el encargado de obtener la densidad, cantidad y posicionamiento de las keywords. Es ejecutado un total de 6 veces por cada URL a analizar, ya que **se extraen resultados de seis etiquetas distintas (<title>, <description>, <body>, <h1>, <alt>y <src>)**.

Para las coincidencias totales (se encuentra la keyword sin modificar), una vez se selecciona el tipo de etiqueta, se buscará, para cada keyword, si existen coincidencias dentro del texto (tokenizado) de dicha etiqueta. En caso afirmativo, se buscarán las palabras una por una hasta encontrar las posiciones de todas las coincidencias.

En el caso de las coincidencias parciales, el método es muy similar al anterior, lo único que ahora habrá que aplicar un algoritmo para calcular la distancia de editado. Para este programa se ha decidido utilizar la variante del algoritmo de la *Distancia de Levenshtein* (Figura 5.21) debido a estudios previos que he realizados en la universidad (más concretamente en la asignatura de Algorítmica), donde dicho algoritmo ofrecía el menor tiempo de ejecución de entre todas las variantes.

Solo se aceptarán como coincidencias parciales, todas aquellas palabras con una distancia de editado menor o igual a 3 (o menos dependiendo de la longitud de la keyword), ya que una mayor distancia facilitaría mucho clasificar como correctas palabras de distinta familia léxica a la keyword.

Este módulo es, posiblemente, el más importantes del código en lo que se refiere a la extracción de parámetros SEO, ya que es el que **obtiene todos los parámetros relacionados con las keywords** (dentro de los seleccionadas para este trabajo), uno de los factores más importantes del SEO.

```
def dp_levenshtein_threshold(x, y, th):

    """
    Num ejecuciones: (Num Queries * Num Resultados + 1) * 6 * Num Keywords * (Num palabras de cada segmento)

    Calcula la Distancia de Levenshtein de dos palabras a partir de un threshold especifico.

    param:  "x": Palabra 1.
            "y": Palabra 2.
            "th": Threshold l mite.

    return: Distancia obtenida

    """

    current_row = [None] * (1+len(x))
    previous_row = [None] * (1+len(x))
    current_row[0] = 0
    for i in range(1, len(x)+1):
        current_row[i] = current_row[i-1] + 1
    for j in range(1, len(y)+1):
        previous_row, current_row = current_row, previous_row
        current_row[0] = previous_row[0] + 1
        for i in range(1, len(x)+1):
            current_row[i] = min( current_row[i-1] + 1,
                previous_row[i] + 1,
                previous_row[i-1] + (x[i-1] != y[j-1]))
        if(min(current_row) > th):
            return th+1
    return min(current_row[len(x)],th+1)
```

Figura 5.21: Distancia de Levenshtein

Fuente: Elaboración propia.

```
def KeywordSearch(soup, tipo, keywords, aux, cTable, numg):

    """
    Num ejecuciones: (Num Queries * Num Resultados + 1) * 6

    Busca las Keywords seleccionadas (de manera Parcial y Total) dentro de un segmento espec fico de la web de
      la URL especificada.

    param:  "soup": Sopa en la que se gurda la informaci n HTML de la URL.
            "tipo": Segmento en el que se buscaran las Keywords (Body/Title/H1/alt/src/descripci n)
            "keywords": Keywords para la query a la que pertence la URL especificada.
            "aux": En caso de que seleccionemos Titulo o Descripci n, pasaremos su texto que ya habiamos
     obtenido previamente (por conveniencia).
            "cTable": Tabla Datos Keywords.
            "numg": Posici n de la URL en la Query.

    return: "TOcurrences": Ocurrencias Totales de las Keywords encontradas en el segmento especificado.
            "POcurrences": Ocurrencias Parciales de las Keywords encontradas en el segmento especificado (
     Usando Levenshtein).
    """

    #Seleccion de Texto
    CTotales = 0
    CParciales = 0
    tex=""
    try:
        if tipo == "body":
            tex = soup.body.get_text()
            cId=len(keywords)*4
        elif tipo == "title":
            tex= aux
            cId=0
        elif tipo == "h1":
            if not(soup.h1 == None):
                tex = soup.h1.get_text()
                cId=len(keywords)*6
            else:
                #H1 No Existe
                return CTotales, CParciales
        elif tipo == "alt":
            for img in soup.find_all('img', alt=True):
                try:
                    alts=img['alt']
                    tex=tex+" "+alts
                except KeyError:
                    tex=tex
            cId=len(keywords)*8
        elif tipo == "src":
            for img in soup.find_all('img', alt=True):
                try:
                    srcs=img['src']
                    tex=tex+" "+srcs
                except KeyError:
                    tex=tex
            cId=len(keywords)*10
        else:
            tex = aux
            cId=len(keywords)*2
    except AttributeError:
        return CTotales, CParciales


    text = tokenizer.sub(' ', tex.lower()).split()
    for word in keywords:
        indices=[]
        indices2=[]
        indices3=[]
       #Coincidencias Totales
        if word in text:
            indices = [i for i, x in enumerate(text) if x == word]
        if(len(word)==4):
            indices2 = [i for i, x in enumerate(text) if dp_levenshtein_threshold(word,x,1) == 1]
        elif(len(word)==5):
            indices2 = [i for i, x in enumerate(text) if dp_levenshtein_threshold(word,x,2) in range(1,3)]
        elif(len(word)>5):
            indices2 = [i for i, x in enumerate(text) if dp_levenshtein_threshold(word,x,3) in range(1,4)]


        #Datos de Tabla de Keywords
        cId=cId+2
        cStrT= "(" + str(len(indices)) + ") " + str(indices)
        indicesAux=indices2+indices3
        indicesAux.sort()
        cStrP= "(" + str(len(indices2)+len(indices3)) + ") " + str(indicesAux)
        cTable[numg][cId]= cStrT
        cTable[numg][cId+1]= cStrP

        #Cuantificaci n Coincidencias
        CTotales = CTotales + len(indices)
        CParciales = CParciales + len(indices2)+len(indices3)
    lt=len(text)
    if lt==0: lt=1
    return (CTotales/lt)*100, (CParciales/lt)*100
```

Figura 5.23: Extracción de parámetros relacionados con keywords

Fuente: Elaboración propia.

5.4 Entrenamiento de los clasificadores

El código desarrollado hace uso de dos clasificadores Gradient Boosting los cuales, en ambos casos, son entrenados por los parámetros SEO de todas las búsquedas obtenidas (*Número de queries * Número de búsquedas*). La única diferencia entre ellos son las etiquetas utilizadas en sus datos de entrenamiento.

El primer clasificador puede incorporar hasta un máximo de 5 clases (+ 1 auxiliar) estructuradas de la siguiente manera:

- **Clase 0:** Clase auxiliar utilizada para clasificar URLs que den error en su extracción.
- **Clase 1:** Pertenecen aquellas URLs en el Top *Límite inferior*[4].
- **Clase 2:** Pertenecen aquellas URLs en el Top [*Límite inferior*-10].
- **Clase 3:** Primer tercio de los resultados superiores a 10 para un *Número de búsquedas* mayor a 25.

  Primera mitad de los resultados superiores a 10 para un *Número de búsquedas* entre 20 y 25.

  Resultados superiores a 10 para un *Número de búsquedas* menores a 20.
- **Clase 4:** Segundo tercio de los resultados superiores a 10 para un *Número de búsquedas* mayor a 25.

  Segunda mitad de los resultados superiores a 10 para un *Número de búsquedas* entre 20 y 25.
- **Clase 5:** Último tercio de los resultados superiores a 10 para un *Número de búsquedas* mayor a 25.

El segundo clasificador, **creado debido a los pobres resultados obtenidos por el primer clasificador**, puede incorporar hasta un máximo de 2 clases (+ 1 auxiliar) estructuradas de la siguiente manera:

- **Clase 0:** Clase auxiliar utilizada para clasificar URLs que den error en su extracción.
- **Clase 1:** Resultados pertenecientes al Top 10.
- **Clase 2:** Resultados superiores al Top 10.

Ambos realizan las mismas funciones mencionadas en la sección 3.2. Para empezar, los clasificadores obtendrán el pesado de cada parámetro, qué serán utilizados tanto de manera orgánica para ilustrar la importancia de los parámetros SEO a los que corresponden, como para calcular (junto al nivel de error de los valores recomendados) un puntuación de la URL a comparar en el informe (opcional). Además, también realizaran una predicción de clase para la URL a analizar (opcional), permitiendo la estimación de su posición para las keywords introducidas en base al rango de posiciones de los datos de entrenamiento pertenecientes a la clase predicha.

El último apartado que falta por cubrir, en cuanto a la creación del clasificador, es la elección de los valores óptimos de sus parámetros internos. Para ello, se comprobará

[4]Tercera variable introducida en la interfaz

la precisión del clasificador mediante la combinación de distintos valores para la tasa ([0.1,0.25,0.5,0.75,1]) y número de árboles ([100,250,500,750,1000]).

El método usado para comprobar la precisión de cada una de las combinaciones es el **k-fold cross-validation**, que divide los datos usados para validar en un número de segmentos a elegir. Cada segmento será utilizado una vez como datos de validación mientras que el resto son utilizados como datos de entrenamiento. La precisión vendrá dada por la media de todos los resultados pero, debido a la naturaleza estocástica del algoritmo, dicho proceso se deberá de repetir más de una vez para sacar una media más precisa En este programa, se utilizan los datos de entrenamiento para la validación y, siempre que el número de URLs extraídas nos lo permita, se utilizarán 5 particiones y 3 repeticiones.

El clasificador que obtenga una mejor precisión será el elegido para ilustrar el pesado de los parámetros SEO y clasificar la URL del informe (opcional). Este proceso de entrenamiento deberá ser realizado por los dos clasificadores.

En caso de que se desee observar el código de entrenamiento de calificadores, consultar la figura B.3 en los anexos.

5.5 Valoración de datos y Generación de resultados

En está última sección del capítulo, se describirá el proceso utilizado por el programa para generar los resultados que serán escritos en el archivo xslx.

Cada documento generado xslx por el programa estará compuesto por tres segmentos: páginas de queries, página Data y la página del Informe.

Páginas de queries

Al terminar la extracción de parámetros SEO para todos los resultados de una query, el programa plasmará los datos en la página del xslx correspondiente a esa query en dos tablas, una para los parámetros primarios y otra para los secundarios. En caso de que algún parámetro no haya podido ser extraído, la celda correspondiente mostrará el string Ünknown"(desconocido) a modo de sustituto.

También se genera, para cada página, dos tablas de medias, una correspondiente a todas las búsquedas, y la otra a las búsquedas delimitadas por textitLímite inferior (valores Top). El programa transformará el formato de los datos no numéricos para permitir la realización de dichas medias, como en el caso de los parámetros de carácter binario (si/no), donde transforma el valor positivo (si) a *1* y el negativo (no) a *0*. Durante el cálculo de medias, aquellas instancias de valor 'Unknown' no serán consideradas, por lo que no afectarán negativamente (o positivamente) a los resultados.

Data

Data es la página del xslx creada por el programa que almacena las tablas de medias globales (general y superior) respecto a todas las queries o, en otras palabras, el programa genera una tabla de medias a partir de las tablas de medias ya creadas, solo que en este caso su implementación es mucho más sencilla debido a que los datos ya presentan un formato adecuado. También se calculará la desviación típica de las tablas superiores locales mediante la librería *NumPy*, la cual se utilizará posteriormente para ajustar los valores recomendados del informe.

Adicionalmente, en caso de que se seleccione comparar una URL con los resultados, la página también almacenará los parámetros SEO obtenidos de dicha URL (primarios y secundarios). La extracción de los parámetros para la URL en cuestión, se realiza justo después de completar todas las queries y entrenar al clasificador

Informe

Componente principal del archivo. Muestra el valor recomendado y el pesado (importancia) calculados de cada parámetro SEO (primario). En caso de seleccionar una URL a comparar, también se mostrará su nivel de error para cada parámetro (primario) así como dos puntuaciones (y estimaciones de clase) basado en los dos clasificadores anteriores.

A continuación, se explicará uno a uno cómo se han calculado los distintos elementos, así como sus características:

- **Valores recomendados:** Lisa de los valores (o rangos) recomendados para cada parámetro SEO.

  Para los parámetros binarios (uso de HTTPS, uso de datos estructurados...), su valor recomendado siempre será 'Si', a no ser que la media de las posiciones superiores sea menor a 0.5, en ese caso se considerará que su valor es 'Indiferente'.

  Por otro lado, los valores recomendados para los parámetros no binarios vendrán dados en forma de rango. El rango de valores que sea un 25 % superior o inferior a los valores de la tabla de medias superiores. Las únicas excepciones a esto serán el rendimiento móvil/sobremesa, donde el rango superior siempre será 100, y la velocidad de carga móvil/sobremesa, donde el rango inferior siempre será 0.

- **Nivel de error de la URL a comparar:** Listado de niveles de error para cada parámetro SEO. El error esta divido en 6 niveles: Error máximo (peor caso), Error alto, Error moderado, Error bajo, Adecuado (entra dentro de los valores recomendados) e indiferente (su valor es irrelevante).

  Si el parámetro es binario, en caso de que el parámetro de la URL a comparar tenga como valor 'Si', su nivel de error será 'adecuado' o 'indiferente'. En caso contrario, su nivel de error dependerá de cuanto se acerca la media general al valor '1', y de la diferencia entre la media general y la superior (solamente en caso de que el valor de la media superior sea mayor). Cuanto mayor sean estos valores, mayor será el nivel de error.

  En caso de los parámetros no binarios, su nivel de error dependerá de en qué porcentaje, el parámetro de a comparar, se aleja del valor de la tabla media superior En caso de que la varianza (previamente calculada) de dicho parámetro sea muy alta, se reducirá, proporcionalmente, el error máximo de dicho parámetro.

- **Peso:** Listado de pesos que suman 1 en total (uno por cada clasificador). Define la importancia de cada parámetro SEO extraído en base a que proporción le ayuda a los clasificadores a decidir a qué clase (rango de posiciones de búsqueda) pertenece una URL. Se obtienen al buscar el atributo *feature_importances_* dentro de los clasificadores Gradient Boosting.

- **Puntuaciones:** Una puntuación sobre 100 que estima como de buena es la URL a comparar (una por cada clasificador). Para su cálculo, primero se ha de multiplicar el peso de cada parámetro por una fracción de su valor dependiendo del nivel de

error[5], sumar cada una de ellas, y dividirlas entre el peso total. Tanto para el divisor como el dividendo, no se tendrán en cuenta los pesados con un nivel de error 'indiferente'.

- **Estimación del ranking:** Estima del rango de posiciones para la URL a comparar (Una por cada clasificador). Se obtiene aplicando la función *predict()* de la biblioteca *sklearn* en los clasificadores.

En caso de que se desee observar el código para la generación de valores recomendados y niveles de error, consultar la figura B.5 en los anexos.

[5]Multiplicación dependiendo del error:
Adecuado*1; Bajo*0.75; Moderado*0.5; Alto*0.25; Máximo*0; Indiferente*0

CAPÍTULO 6
Pruebas y resultados

En este capítulo, se podrá observar en qué medida el programa desarrollado ha conseguido alcanzar los objetivos propuestos al inicio del trabajo. Para ello se explicarán como han sido las pruebas realizadas para validar su funcionalidad y calidad, así como los resultados obtenidos a partir de ellas.

6.1 Validación de la extracción de parámetros

En este apartado se mostrará como se ha realizado la validación de los distintos parámetros SEO extraídos (Objetivo principal 1). Para ello, a lo largo de esta sección, se irá mostrando uno de los documentos (xslx) generados por el programa a modo ejemplo (Objetivo secundario 3).

Los datos introducidos en programa para generar este ejemplo son los siguientes:

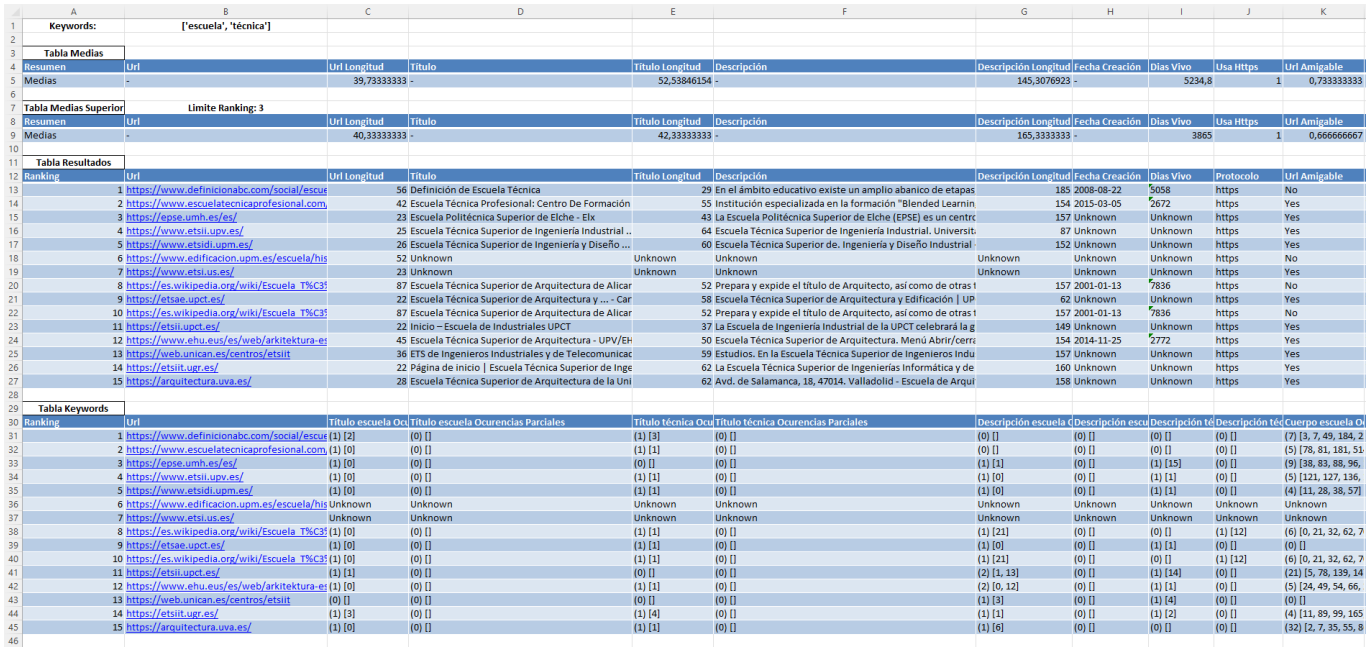
- **Número de queries:** 3
- **Número de búsquedas por query:** 15
- **Keywords=Query:** Sí (Las keywords buscadas corresponden a las palabras que componen la query)
- **Ranking inferior límite:** 3 (Los primeros 3 afectarán en mayor medida al análisis)
- **Comparar URL (informe):** Sí (será mostrado al final de la sección)

  -URL a comparar: http://www.upv.es/

  -Keywords de la URL: Universidad

- **Queries Buscadas:** Universidad, Universidades de informática, **Escuela Técnica**

Para ser exactos, será la página de la query *Escuela Técnica* la que se usará para ejemplificar la validación de extracción de parámetros. Para mostrar como son los resultados generados correspondientes al análisis de datos, al final de la sección también se mostrará un segmento de la página *informe* generada para este ejemplo, aunque las pruebas de calidad de dicho apartado serán realizadas en la siguiente sección.



	A	B	C	D	E	F	G	H	I	J	K
1	Keywords:	['escuela', 'técnica']									
2											
3	Tabla Medias										
4	Resumen	Url	Url Longitud	Titulo	Titulo Longitud	Descripción	Descripción Longitud	Fecha Creación	Dias Vivo	Usa Https	Url Amigable
5	Medias	-	39,73333333	-	52,53846154	-	145,3076923	-	5234,8	1	0,733333333
6											
7	Tabla Medias Superior	Limite Ranking: 3									
8	Resumen	Url	Url Longitud	Titulo	Titulo Longitud	Descripción	Descripción Longitud	Fecha Creación	Dias Vivo	Usa Https	Url Amigable
9	Medias	-	40,33333333	-	42,33333333	-	165,3333333	-	3865	1	0,666666667
10											
11	Tabla Resultados										
12	Ranking	Url	Url Longitud	Titulo	Titulo Longitud	Descripción	Descripción Longitud	Fecha Creación	Dias Vivo	Protocolo	Url Amigable
13	1	https://www.definicionabc.com/social/escue	56	Definición de Escuela Técnica	29	En el ámbito educativo existe un amplio abanico de etapas	185	2008-08-22	5058	https	No
14	2	https://www.escuelatecnicaprofesional.com/	42	Escuela Técnica Profesional: Centro De Formación	55	Institución especializada en la formación "Blended Learnin	154	2015-03-05	2672	https	Yes
15	3	https://epse.umh.es/es/	23	Escuela Politécnica Superior de Elche - Elx	43	La Escuela Politécnica Superior de Elche (EPSE) es un centro	157	Unknown	Unknown	https	Yes
16	4	https://www.etsii.upv.es/	25	Escuela Técnica Superior de Ingeniería Industrial ..	64	Escuela Técnica Superior de Ingeniería Industrial. Universit	87	Unknown	Unknown	https	Yes
17	5	https://www.etsidi.upm.es/	26	Escuela Técnica Superior de Ingeniería y Diseño ...	60	Escuela Técnica Superior de. Ingeniería y Diseño Industrial	152	Unknown	Unknown	https	Yes
18	6	https://www.edificacion.upm.es/escuela/his	52	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	https	No
19	7	https://www.etsi.us.es/	23	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	https	Yes
20	8	https://es.wikipedia.org/wiki/Escuela_T%C3%	87	Escuela Técnica Superior de Arquitectura de Alicar	52	Prepara y expide el título de Arquitecto, así como de otras t	157	2001-01-13	7836	https	No
21	9	https://etsae.upct.es/	22	Escuela Técnica Superior de Arquitectura y ... - Car	58	Escuela Técnica Superior de Arquitectura y Edificación | UP	62	Unknown	Unknown	https	Yes
22	10	https://es.wikipedia.org/wiki/Escuela_T%C3%	87	Escuela Técnica Superior de Arquitectura de Alicar	52	Prepara y expide el título de Arquitecto, así como de otras t	157	2001-01-13	7836	https	No
23	11	https://etsii.upct.es/	22	Inicio – Escuela de Industriales UPCT	37	La Escuela de Ingeniería Industrial de la UPCT celebrará la g	149	Unknown	Unknown	https	Yes
24	12	https://www.ehu.eus/es/web/arkitektura-et	45	Escuela Técnica Superior de Arquitectura - UPV/EH	50	Escuela Técnica Superior de Arquitectura. Menú Abrir/cerra	154	2014-11-25	2772	https	Yes
25	13	https://web.unican.es/centros/etsiit	36	ETS de Ingenieros Industriales y de Telecomunicac	59	Estudios. En la Escuela Técnica Superior de Ingenieros Indu	157	Unknown	Unknown	https	Yes
26	14	https://etsiit.ugr.es/	22	Página de inicio | Escuela Técnica Superior de Inge	62	La Escuela Técnica Superior de Ingenierías Informática y de	160	Unknown	Unknown	https	Yes
27	15	https://arquitectura.uva.es/	28	Escuela Técnica Superior de Arquitectura de la Uni	62	Avd. de Salamanca, 18, 47014. Valladolid - Escuela de Arqui	158	Unknown	Unknown	https	Yes
28											
29	Tabla Keywords										
30	Ranking	Url	Titulo escuela Oc	Titulo escuela Ocurencias Parciales	Titulo técnica Ocu	Titulo técnica Ocurencias Parciales	Descripción escuela C	Descripción escu	Descripción té	Descripción téc	Cuerpo escuela O
31	1	https://www.definicionabc.com/social/escue	(1) [2]	(0) []	(1) [3]	(0) []	(0) []	(0) []	(0) []	(0) []	(7) [3, 7, 49, 184, 2
32	2	https://www.escuelatecnicaprofesional.com/	(1) [0]	(0) []	(1) [1]	(0) []	(0) []	(0) []	(0) []	(0) []	(5) [78, 81, 181, 51
33	3	https://epse.umh.es/es/	(1) [0]	(0) []	(0) []	(0) []	(1) [1]	(0) []	(1) [15]	(0) []	(9) [38, 83, 88, 96,
34	4	https://www.etsii.upv.es/	(1) [0]	(0) []	(1) [1]	(0) []	(1) [0]	(0) []	(1) [1]	(0) []	(5) [121, 127, 136,
35	5	https://www.etsidi.upm.es/	(1) [0]	(0) []	(1) [1]	(0) []	(1) [0]	(0) []	(1) [1]	(0) []	(4) [11, 28, 38, 57]
36	6	https://www.edificacion.upm.es/escuela/his	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown
37	7	https://www.etsi.us.es/	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown
38	8	https://es.wikipedia.org/wiki/Escuela_T%C3%	(1) [0]	(0) []	(1) [1]	(0) []	(1) [21]	(0) []	(0) []	(1) [12]	(6) [0, 21, 32, 62, 7
39	9	https://etsae.upct.es/	(1) [0]	(0) []	(1) [1]	(0) []	(1) [0]	(0) []	(1) [1]	(0) []	(0) []
40	10	https://es.wikipedia.org/wiki/Escuela_T%C3%	(1) [0]	(0) []	(1) [1]	(0) []	(1) [21]	(0) []	(0) []	(1) [12]	(6) [0, 21, 32, 62, 7
41	11	https://etsii.upct.es/	(1) [1]	(0) []	(0) []	(0) []	(2) [1, 13]	(0) []	(1) [14]	(0) []	(21) [5, 78, 139, 14
42	12	https://www.ehu.eus/es/web/arkitektura-et	(1) [0]	(0) []	(1) [1]	(0) []	(2) [0, 12]	(0) []	(1) [1]	(0) []	(5) [24, 49, 54, 66,
43	13	https://web.unican.es/centros/etsiit	(0) []	(0) []	(0) []	(0) []	(1) [3]	(0) []	(1) [4]	(0) []	(0) []
44	14	https://etsiit.ugr.es/	(1) [3]	(0) []	(1) [4]	(0) []	(1) [1]	(0) []	(1) [2]	(0) []	(4) [11, 89, 99, 165
45	15	https://arquitectura.uva.es/	(1) [0]	(0) []	(1) [1]	(0) []	(1) [6]	(0) []	(0) []	(0) []	(32) [2, 7, 35, 55, 8
46											

Figura 6.1: Query (*Escuela Técnica*) - Fragmento

Fuente: Elaboración propia.

En esta ocasión, en la figura de arriba (6.1) se ha decidido mostrar un segmento de todas las tablas (y búsquedas) que componen los resultados de la query (*Escuela Técnica*). Esto se ha realizado únicamente con el objetivo de ilustrar como son los resultados generados para las queries, así que, por motivos prácticos, en futuros segmentos solo se mostrarán los resultados de las primeras 3 URLs de la tabla de resultados (tercera tabla de la figura). Se recomienda consultar al anexo A.1 para la visualización completa del documento de ejemplo.

Google Search

Para la validación de las URLs, los títulos y las descripciones obtenidas, se ha realizado, dentro del buscador de Google, la búsqueda de las queries utilizadas por el código para obtener dichos elementos. Por ejemplo, si compramos los datos mostrados en la tabla de resultados de la figura 6.1 con los mostrados en la figura 6.3, observaremos que, el ranking, la URL, el título y la descripción de los primeros tres resultados coinciden.

Una vez validados los elementos más misceláneos, es posible comprobar manualmente el valor los siguientes parámetros:

- **Longitud (URL, título y descripción):** Contando los caracteres de cada elemento.
- **Protocolo:** Visible dentro de la URL.
- **Amigabilidad de la URL:** Comprobando que no se utilizan caracteres extraños o extensiones innecesarias dentro de la URL.
- **Densidad, cantidad y posicionamiento de las keywords (URL, título y descripción):** Contando y midiendo manualmente.

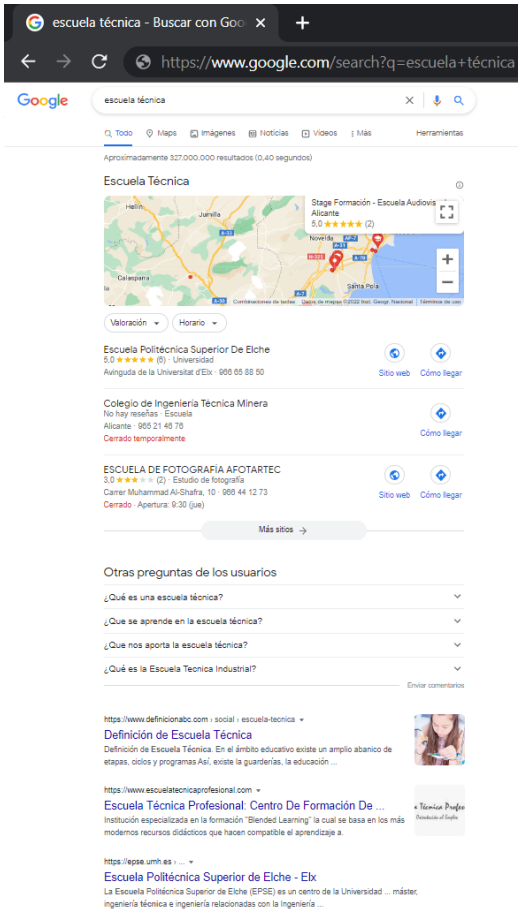


Figura 6.3: Búsqueda de *Escuela Técnica* en Google Search

Fuente: Elaboración propia a partir de los datos generados por Google Search.

Durante el periodo de validación del programa, se comprobó que los valores de los parámetros pertenecientes al listado anterior siempre coincidían con los extraídos por el programa, a excepción de aquellos casos en los que no se pudo establecer una conexión con la web (fallan longitudes y keywords), o en aquellos donde la descripción mostraba la última fecha modificación de la web (falla longitud de la descripción), extrayéndose dicha fecha en lugar que la descripción en sí.

APIs

Como se ha mencionado anteriormente, este programa interactúa directamente con dos APIs distintas, *Google APIs* y *WHOIS*. En el caso de la primera, debido a que es la misma Google quien nos ofrece el valor de los distintos parámetros (Compatibilidad móvil, rendimiento y velocidad de carga), su validación es considerada innecesaria, ya que la entidad detrás de la API es la misma que hay detrás del buscador analizado.

Cabe destacar que, a pesar de su fiabilidad, existe la posibilidad (baja) de que la API no consiga extraer con éxito alguno de los parámetros. Esto se puede observar en la tercera URL de la figura 6.5, la cual falla la extracción de la compatibilidad con móviles.

Ranking	Url	Fecha Creación	Dias Vivo	Compatibilidad Móvil	Rendimiento Móvil	Velocidad Carga Móvil	Rendimiento Sobremesa	Velocidad Carga Sobremesa
1	https://www.definicionabc.com/social/escuela	2008-08-22	5058	MOBILE_FRIENDLY	34	4,9 s	79	1,1 s
2	https://www.escuelatecnicaprofesional.com/	2015-03-05	2672	MOBILE_FRIENDLY	46	7,8 s	89	1,7 s
3	https://epse.umh.es/es/	Unknown	Unknown	Unknown	49	12,7 s	59	3,2 s

Figura 6.5: Parámetros obtenidos de APIs - Ejemplo

Fuente: Elaboración propia.

Por otro lado, para comprobar que la fecha de creación extraída mediante *WHOIS* sea correcta, se hizo uso de la herramienta online de *DomainTools*[1], que como se muestra en la figura 6.7, obtiene (entre otros) los datos relacionados con la edad de un dominio. Afortunadamente, exceptuando el caso de que falle la extracción, no se encontró ningún supuesto donde el valor obtenido fuera erróneo.

Figura 6.7: Comprobación de la fecha de creación de un dominio

Fuente: Elaboración propia a partir de DomainTools https://whois.domaintools.com/definicionabc.com

Buscador de errores

Para validar la correcta extracción de los errores, se utilizó *Dead link checker*[2], una herramienta que recolecta todos los errores encontrados en los enlaces de una web.

Como se observa en la figura 6.11, donde se analiza la primera URL de la figura 6.9, el motivo del error forma parte de los resultados generados por la herramienta, permitiendo validar tanto los errores 404 como los de conexión (errores diferentes al 404).

Ranking	Url	Errores 404	Errores Conexión
1	https://www.definicionabc.com/social/escuela	0	1
2	https://www.escuelatecnicaprofesional.com/	0	2
3	https://epse.umh.es/es/	0	6

Figura 6.9: Número de Errores - Ejemplo

Fuente: Elaboración propia.

Los resultados obtenidos por nuestro programa son, en general, muy similares a los obtenidos por la *Dead link checker*, aunque sí se encontraron fuertes discrepancias en aquellas webs donde el webmaster haya bloqueado el uso del protocolo HEAD (utilizado en nuestro código para comprobar el estado de los enlaces), recibiendo, en lugar del código de estado correcto, el error 405 (Método No Valido). Aunque es cierto que se podría utilizar el protocolo GET en su lugar, esto ralentizaría demasiado el tiempo de ejecución de nuestro programa y, teniendo en cuenta que el análisis de una web está limitado a 10 minutos, muchas de ellas no llegarían a terminar.

[1]Herramienta Online WHOIS: https://whois.domaintools.com/
[2]Herramienta Online DeadLinkChecker: https://www.deadlinkchecker.com/

Figura 6.11: Comprobación de errores

Fuente: Elaboración propia a partir de Dead Link Checker https://www.deadlinkchecker.com/website-dead-link-checker.asp

Link checker

La última herramienta utilizada para comparar y validar nuestros resultados fue el analizador de enlaces de *SEO Review Tools*[3]. En ella se muestran de manera directa, el número de enlaces totales, externos, duplicados (no se analizan en nuestro código) y NoFollow, y, de manera indirecta a partir de los anteriores, el número de enlaces internos y Follow. Aunque a diferencia de nuestro código, el analizador de SEO Review Tools no divide los Follow y NoFollow en externos e internos.

Ranking	Url	Links Totales	Links Internos	Links Externos	Follow Internos	NoFollow Internos	Follow Externos	NoFollow Externos
1	https://www.definicionabc.com/social/escuela	64	60	4	57	1	3	3
2	https://www.escuelatecnicaprofesional.com/	85	76	9	76	9	0	0
3	https://epse.umh.es/es/	127	119	8	119	8	0	0

Figura 6.13: Número de Enlaces - Ejemplo

Fuente: Elaboración propia.

En el ejemplo mostrado por la figura 6.15, podemos observar que sus resultados coinciden con los nuestros 6.13.

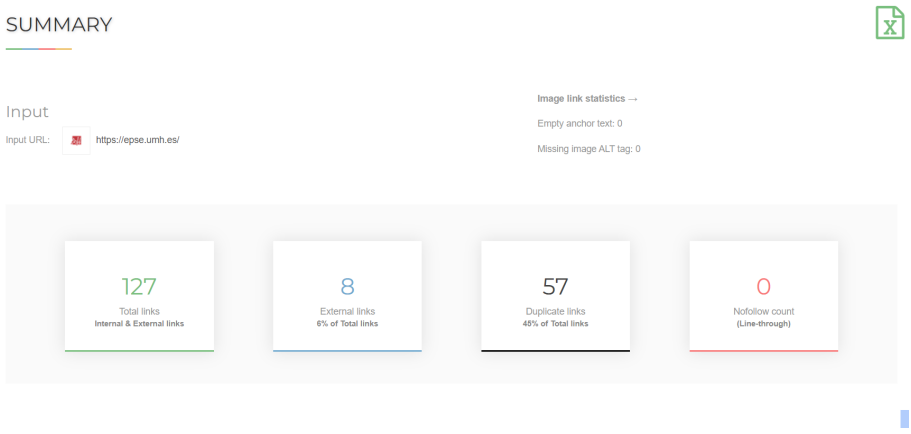


Figura 6.15: Comprobación de errores

Fuente: Elaboración propia a partir de SEO Review Tools https://www.seoreviewtools.com/internal-link-analyzer/

[3]Herramienta Online SEO Review Tools - Internal links analyzer: https://www.seoreviewtools.com/internal-link-analyzer/

Durante las pruebas realizadas con analizador de enlaces, se pudo observar que la clasificación de los enlaces (Follow/NoFollow y Externo/Interno) era idéntica a la de nuestro programa, pero en ocasiones, para las webs con mayor número de enlaces, podía darse la situación qué la herramienta encontrara uno o dos enlaces más (o menos). No se pudo encontrar el motivo a dicho comportamiento, pero tampoco se tuvo mucho en cuenta debido a que al tratarse de webs con cientos de enlaces, la diferencia resultaba trivial.

Búsqueda manual de robots.txt y sitemap.xml

En el caso de robots.txt, a no ser que se produzca un error de conexión, nuestro programa siempre lo obtendrá. Esto se puede comprar para cualquier URL con tan solo buscar el archivo 'robots.txt' en el directorio de la URL (figura 6.19).

Ranking	Url	Robots.txt	Sitemap.xml
1	https://www.definicionabc.com/social/escuela	Yes	No
2	https://www.escuelatecnicaprofesional.com/	Yes	Yes (1)
3	https://epse.umh.es/es/	Yes	Yes (1)

Figura 6.17: robots.txt y sitemap.xml - Ejemplo

Fuente: Elaboración propia.

En el caso de los sitemaps bastaría con comprobar que el número de instancias de ellos dentro del robots.txt es la misma que la obtenida por el programa (figuras 6.17 y 6.19). Aunque hay que tener en cuenta que el programa también busca un sitemap en el directorio raíz si no se encuentran resultados.

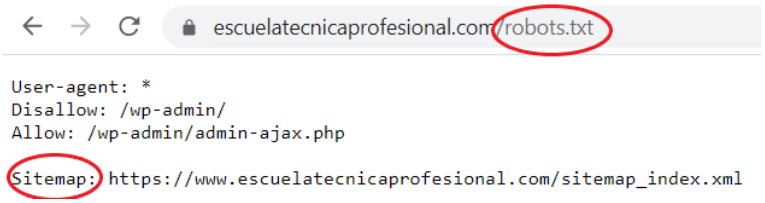


Figura 6.19: Búsqueda manual de sitemaps y robots.txt

Fuente: Elaboración propia a partir de Escuela Técnica Profesional https://www.escuelatecnicaprofesional.com/robots.txt

Búsqueda manual en el HTML

En el caso de los parámetros que solo precisan del recuento de etiquetas usadas (cabeceras, imágenes y vídeos) o de la presencia del algún vocabulario concreto (datos estructurados y marcado Schema), solo es necesario el uso de la herramienta de inspección propia de navegadores como Chrome y Firefox para asegurarse que el código actúa como esperamos.

6.1.1. Visualización del informe

Dentro de nuestros datos generados, el informe se utiliza para mostrar las conclusiones obtenidas tras la fase de extracción de parámetros. Teniendo en cuenta el significado que esto conlleva, se ha intentado ofrecer la mejor visualización posible dentro de los limites (estéticos) de un archivo xlsx,

En la figura 6.21 se puede observar un fragmento del informe generado a partir de los datos mencionados al principio de la sección. Para una visualización completa acuda al anexo A.1.

Leyenda			Conteo Errores		General (3 clases)	
	Máximo		4		Puntuación	Ranking Estimado
	Alto		4		66/100	Top 10
	Moderado		6			Precisión del clasificador: 0.417 (Desviación: 0.083)
	Bajo		10			
	Adecuado		13		Top 10 (2 clases)	
	Indiferente		2		Puntuación	En Top 10
					70/100	Si
						Precisión del clasificador: 0.553 (Desviación: 0.115)
Informe						
Web Analisis	Url	Url Longitud	Título	Título Longitud	Descripción	Descripción Longitud
Resultados	http://www.upv.es/	18	UPV Universitat Politècnic	39	Portal oficial de la Universitat Politècnica de València	143

Nivel de Error	Atributo	Valores Actuales	Valores Recomendados	Error Máximo	Peso (General)	Peso (Top 10)
	Url Longitud	18.0	[41-68]	3	0,11685624	0,006467064
	Título Longitud	39.0	[30-50]	5	0,002695342	0,025817139
	Descripción Longitud	143.0	[111-185]	5	0,152151241	0,10832658
	Dias Vivo	Unknown	[2178-3629]	3	1,62445E-08	1,62779E-06
	Uso Https	Yes	Yes	5	0,000430283	0,163277492
	Url Amigable	Yes	Yes	4	1,88807E-05	0,000207777
	Robots.txt	Yes	Yes	5	0,012328692	1,60219E-05
	Sitemap.xml	No	Yes	4	2,10326E-05	5,28213E-05
	Compatibilidad Móvil	Yes	Yes	5	0,003183769	0,007195284
	Rendimiento Móvil	43.0	[41-100]	5	0,057007285	0,063149507
	Velocidad Carga Móvil	15.7	[0-14.3]	3	0,12876763	0,09662823
	Rendimiento Sobremesa	83.0	[68-100]	5	0,018110064	0,038840888
	Velocidad Carga Sobremesa	2.4	[0-2.2]	5	0,084832944	0,020451922
	Cabeceras	3.0	[2-4]	5	0,012924485	0,021613551
	Datos Estructurados	Yes	Indiferente	0	4,26688E-16	4,09941E-07
	Schema Markup	Yes	Indiferente	0	3,81569E-06	1,58228E-17
	Imágenes Totales	21.0	[42-70]	2	0,050632103	0,094691862
	Videos Totales	2.0	[0-1]	5	0,011560952	0,005235359
	Links Totales	154.0	[219-365]	2	0,035468474	0,001073678
	Links Internos	129.0	[210-350]	2	0,01056422	0,008270435
	Links Externos	25.0	[9-15]	3	0,012344984	0,010054457
	Follow Internos	129.0	[209-349]	2	0,000979296	0,007493895
	Follow Externos	25.0	[9-15]	3	0,020453378	0,013437918
	NoFollow Internos	0.0	[0-1]	5	6,80234E-05	0,006130609
	NoFollow Externos	0.0	[0-1]	5	8,24394E-05	0,000402559
	Errores 404	0.0	[0-1]	5	0,00022409	0,000166445
	Errores Conexión	5.0	[0-32]	2	0,013556843	0,001649403
	Título Ocurrencias Totales (Keywords%)	0.0%	[15.81%-26.35%]	5	0,013404914	0,026717266
	Título Ocurrencias Parciales (Keywords%)	20.0%	[4.17%-6.95%]	2	3,58209E-06	2,25054E-07

Figura 6.21: Ejemplo de un informe - Fragmento

Fuente: Elaboración propia.

6.2 Valoración de parámetros

De entre todos los resultados generados por el programa, aquellos correspondientes al informe son, sin lugar a duda, los más esenciales de todos. En él, no solo se muestra la importancia relativa calculada para cada parámetro (Objetivo principal 2), sino que también permite la comparación directa de las queries realizadas con una URL particular (Objetivo principal 3) y, de manera más indirecta, también es posible extrapolar los resultados obtenidos de distintos informes para esbozar cuáles serían los parámetros más importantes de manera global (independiente de la query), así como sus valores recomendados (Objetivo secundario 2).

Para llegar a una conclusión sobre la capacidad del código para realizar dichas tareas, se han recolectado dos conjuntos de muestras generadas por el programa.

El primer conjunto, disponible en el anexo A.2, ha sido utilizado, a modo de prueba, para observar que tamaño de queries (número de resultados) ofrecen una clasificación más precisa, y así utilizar el valor óptimo dentro del conjunto final. El motivo por el que se decidió realizar dicha prueba fue para asegurarse que las clases correspondientes a las posiciones bajas del buscador (menor al Top 10) tuvieran un número de muestras (URLs) lo suficientemente grande, ya que, en caso contrario, las clases podrían llegar a ser demasiado similares entre si.

Por otro lado, el segundo conjunto (o conjunto final), disponible en el anexo A.3 ha sido utilizado para comprobar la calidad de los resultados generados en los informes, más concretamente, los valores recomendados y los pesados.

6.2.1. Pruebas de precisión y selección de muestras

Los datos correspondientes al primer conjunto están compuestos por un total de 3 pruebas distintas. Cada prueba está formada por 6 informes (para un total de 18) los cuales, a excepción del número de resultados, utilizan los mismos datos de inicialización:

- Cinco queries de temática similar.

- Keywords correspondientes a las palabras utilizadas en la query (menos stopwords), ya que es lo que más se asemeja a lo realizado por el buscador de Google.

- *Límite inferior* (clase 1 del clasificador) correspondiente a las tres primeras posiciones de cada query.

- Comparación de datos con una URL de temática similar a la queries analizadas.

En la siguiente tabla se muestra como la variación en el número de resultados ha afectado a la precisión (calculada mediante **k-fold cross-validation**) de los clasificadores en cada una de las pruebas:

	Precisión de los Clasificadores											
	Clasificador General (Resultados/Queries)						Clasificador Simple (Resultados/Queries)					
	20	30	40	50	100	200	20	30	40	50	100	200
Prueba 0.1	30,70%	27,00%	24,80%	29,90%	37,00%	49,50%	56,50%	63,20%	72,70%	76,30%	89,30%	93,90%
Prueba 0.2	33,60%	28,50%	29,70%	31,20%	37,60%	40,60%	62,20%	71,80%	77,10%	79,20%	89,80%	94,30%
Prueba 0.3	36,90%	27,70%	28,60%	32,70%	38,80%	42,50%	67,30%	66,50%	77,40%	82,90%	89,30%	94,70%
Medias	33,73%	27,73%	27,70%	31,27%	37,80%	44,20%	62,00%	67,17%	75,73%	79,47%	89,47%	94,30%

Tabla 6.1: Precisión de los clasificadores según el número de resultados por query

Fuente: Elaboración propia.

Los resultados muestran, para ambos clasificadores, una clara tendencia creciente de la precisión a medida que se aumenta el número de resultados por query.

Para comprobar que la precisión de los informes con un alto número de resultados no se debe a que únicamente se clasifican correctamente los datos en posiciones bajas (por pertenecer a clases con un número de muestras notablemente mayor), se pueden utilizar los ejemplos obtenidos al comparar las URLs en los informes de las pruebas (Tabla 6.3).

	Clasificación de las pruebas iniciales					
	Resultados/Queries	**URL**	**Query/Keyword**	**Estimación General**	**Estimación Top 10**	**Posición Real**
Prueba 0.1	**20**	http://www.upv.es/	Universidad	Top 3	Sí	48
	30	http://www.upv.es/	Universidad	Pos: 22+	Sí	48
	40	http://www.upv.es/	Universidad	Pos: [10-20]	No	48
	50	http://www.upv.es/	Universidad	Pos: [10-23]	No	48
	100	http://www.upv.es/	Universidad	Pos: [10-40]	No	48
	200	http://www.upv.es/	Universidad	Pos: [10-73]	No	48
Prueba 0.2	**20**	https://www.blogger.com/	Blog	Top 3	Sí	3
	30	https://www.blogger.com/	Blog	Top 10 (4-10)	Sí	3
	40	https://www.blogger.com/	Blog	Top 3	Sí	3
	50	https://www.blogger.com/	Blog	Pos: [23-36]	Sí	3
	100	https://www.blogger.com/	Blog	Pos: [10-40]	No	3
	200	https://www.blogger.com/	Blog	Top 3	No	3
Prueba 0.3	**20**	https://www.amazon.es/	Comprar	Top 3	No	1
	30	https://www.amazon.es/	Comprar	Top 3	Sí	1
	40	https://www.amazon.es/	Comprar	Pos: [10-20]	Sí	1
	50	https://www.amazon.es/	Comprar	Pos: [10-23]	Sí	1
	100	https://www.amazon.es/	Comprar	Top 3	No	1
	200	https://www.amazon.es/	Comprar	Top 3	Sí	1

Tabla 6.3: Clasificación de los informes de las muestras iniciales

Fuente: Elaboración propia.

Claramente se puede observar que, en ambos casos, utilizar 200 resultados por query parece ser lo más preciso. A pesar de esto, se ha decidido reducir moderadamente el número de resultados para las muestras finales debido a que, en ocasiones, el buscador de Google no llega a ofrecer 200 búsquedas, algo que puede modificar notablemente la precisión de aquellos informes con queries que sufran este problema (ya que sería el equivalente a bajar el número de resultados).

Para que sea justa la comparación de todas las muestras del conjunto final, se ha decidido reducir el número de resultados por query a 150 y, a excepción del número de queries (el cual será doblado para aumentar los datos de entrenamiento), se utilizarán los mismos datos de inicialización que en el caso anterior. En el caso de este conjunto, se realizarán un total de 6 pruebas de tan solo 1 informe cada una.

A continuación, se mostrará todos los elementos que componen conjunto final:

- **60 Queries** (10 por muestra)
- **9000 URLs** (150 por query)
- **351.000 Parámetros** (39 por URL) + **Posicionamiento de keywords**
- **12 listados de pesos** (2 por muestra)
- **6 listados de valores recomendados** (1 por muestra)
- **6 URLs a comparar** (1 por muestra)
    - Con resultados notablemente satisfactorios (Tabla 6.5).

	Clasificación de las pruebas finales				
	URL	Query/Keyword	Estimación General	Estimación Top 10	Posición Real
Prueba 1	http://www.upv.es/	Universidad	Pos: [10-56]	No	48
Prueba 2	https://www.amazon.es/	Comprar	Top 3	No	1
Prueba 3	https://www.ikea.com/es/es/	Muebles	Pos: [10-56]	No	31
Prueba 4	https://www.woorank.com/es	SEO	Pos: [56-102]	No	58
Prueba 5	https://es.wikipedia.org/wiki/Robot	Robot	Pos: [10-56]	No	1
Prueba 6	https://www.vacunacovid.gob.es/	COVID	Pos: Top 10	Sí	9

Tabla 6.5: Clasificación de los informes de las muestras finales

Fuente: Elaboración propia.

6.2.2. Pesados - Importancia de los parámetros

Ya recolectadas y analizadas las muestras del conjunto final, se han recopilado todos los pesos obtenidos en cada muestra para observar sí se obtiene unos resultados estables y robustos que pueden definir, en cierta medida, la importancia de cada uno de los parámetros SEO extraídos.

Afortunadamente para este trabajo, los datos recolectados por el clasificador general (Tabla 6.7) parecen ser capaces de definir notablemente muchos de los parámetros que, aunque a mitad de tabla tengan una desviación relativamente elevada (respecto a su media), si que parecen coincidir en los parámetros en los extremos de la tabla.

Los datos del clasificador general parecen sugerir que:

- La **longitud de las URLs** y la **densidad de keywords en el cuerpo del HTML** parecen ser los **parámetros más influyentes** para el posicionamiento, seguidos de cerca por la **antigüedad del dominio**.

- La **longitud de los meta elementos** (título y descripción), los **Core Web Vitals en dispositivos móviles** y el **número de imágenes** también son factores a **tener en cuenta**.

- El uso de **protocolo HTTPS**, **sitemaps**, **datos estructurados** (y **marcado Schema**), **robots.txt**, **videos** y una **URL amigable**, al igual que ofrecer una **buena compatibilidad con dispositivos móviles**, **no parecen tener efecto** dentro del posicionamiento.

A pesar de que los resultados obtenidos por el segundo clasificador (Tabla 6.9) presentan una desviación muy alta, las conclusiones que se pueden sacar de el se asemejan bastante al del clasificador anterior.

Los datos del clasificador Top 10 parecen sugerir que:

- La **longitud de las URLs** parece ser el parámetro que más **diferencia a los resultados de las primeras 10 posiciones**.

- La **antigüedad del dominio**, la **densidad de keywords en el cuerpo del HTML**, la **longitud de la meta descripción**, y el **número de imágenes** también son **importantes para alcanzar un posicionamiento alto**.

- El uso de **datos estructurados** (y **marcado Schema**), **robots.txt**, **videos** y una **URL amigable**, al igual que ofrecer una **buena compatibilidad con dispositivos móviles**, **no parecen ser elementos característicos de las web mejor clasificadas**.

El hecho de que los clasificadores coincidan en la valoración de tantos parámetros es algo muy positivo, ya que si el segundo clasificador permite distinguir de manera directa

(al ser solo dos clases) que parámetros definen a los resultados mejor posicionados del resto, pero estos presenten una gran varianza en sus elecciones, el que dichas elecciones coincidan con las de un clasificador más robusto (pero con peor capacidad de distinguir que parámetro afecta a que clase) aumentaría notablemente su valor.

De los resultados de ambos clasificadores podríamos llegar a distinguir dos grupos importantes para alcanzar posiciones altas en el buscador:

- **Parámetros muy importantes:** Longitud de la URL, antigüedad del dominio, densidad de keywords en el cuerpo del HTML, longitud de la meta descripción y número de imágenes.
- **Parámetros triviales:** Datos estructurados, marcado Schema, robots.txt, número de vídeos, URL amigable y compatibilidad móvil.

	Pesos generados por el Clasificador General (5 clases)							
	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5	Prueba 6	Media	Desviación
Url Longitud	8,92%	5,39%	6,68%	6,54%	7,84%	6,12%	6,91%	1,27%
Cuerpo Ocurrencias Totales (Keywords%)	6,49%	5,42%	5,86%	7,74%	7,53%	6,01%	6,51%	0,94%
Dias Vivo	5,21%	5,11%	6,35%	6,04%	4,88%	5,35%	5,49%	0,58%
Cuerpo Ocurrencias Parciales (Keywords%)	6,25%	6,59%	3,51%	4,53%	6,56%	4,39%	5,30%	1,33%
Velocidad Carga Móvil	5,84%	5,14%	4,68%	5,93%	5,04%	5,09%	5,29%	0,49%
Descripción Longitud	5,56%	3,51%	5,04%	3,24%	5,21%	5,85%	4,73%	1,09%
Rendimiento Móvil	3,68%	5,02%	4,62%	5,59%	3,39%	4,28%	4,43%	0,83%
Título Longitud	4,27%	4,77%	4,41%	3,30%	3,53%	5,04%	4,22%	0,68%
Imágenes Totales	3,53%	4,14%	4,07%	3,85%	4,11%	3,40%	3,85%	0,32%
Velocidad Carga Sobremesa	3,79%	4,14%	2,46%	2,69%	3,83%	4,49%	3,57%	0,81%
Links Totales	2,80%	3,71%	3,96%	4,30%	2,89%	3,51%	3,53%	0,59%
Rendimiento Sobremesa	3,02%	3,52%	2,66%	2,80%	4,82%	4,20%	3,50%	0,86%
Título Ocurrencias Totales (Keywords%)	4,69%	2,56%	4,30%	3,27%	2,76%	2,70%	3,38%	0,91%
Descripción Ocurrencias Totales (Keywords%)	3,05%	2,79%	3,57%	4,11%	2,88%	3,51%	3,32%	0,50%
Follow Internos	2,12%	2,76%	2,68%	3,11%	2,88%	3,09%	2,77%	0,36%
Alt Ocurrencias Totales (Keywords%)	2,53%	2,82%	2,44%	3,77%	2,44%	2,21%	2,70%	0,56%
Links Internos	2,79%	3,25%	2,53%	2,72%	2,35%	2,44%	2,68%	0,33%
Errores Conexión	1,83%	2,79%	2,19%	2,54%	2,93%	2,64%	2,49%	0,41%
Links Externos	1,64%	2,31%	1,84%	2,29%	3,21%	2,73%	2,34%	0,57%
Follow Externos	2,04%	1,43%	2,05%	2,14%	2,95%	2,82%	2,24%	0,56%
Alt Ocurrencias Parciales (Keywords%)	2,50%	2,46%	2,80%	1,12%	1,72%	2,55%	2,19%	0,64%
Descripción Ocurrencias Parciales (Keywords%)	2,01%	3,00%	1,84%	1,15%	3,41%	1,39%	2,13%	0,89%
Src Ocurrencias Totales (Keywords%)	1,66%	1,31%	1,80%	4,87%	1,10%	1,08%	1,97%	1,45%
H1 Ocurrencias Totales (Keywords%)	1,51%	1,72%	1,56%	2,11%	1,27%	2,82%	1,83%	0,56%
NoFollow Internos	1,37%	2,99%	2,12%	1,64%	1,47%	1,30%	1,81%	0,65%
Src Ocurrencias Parciales (Keywords%)	1,56%	2,11%	2,70%	1,34%	1,40%	1,02%	1,69%	0,61%
Cabeceras	1,78%	1,71%	1,45%	1,40%	1,19%	2,32%	1,64%	0,39%
NoFollow Externos	0,98%	1,98%	1,13%	1,79%	0,99%	1,70%	1,43%	0,45%
Título Ocurrencias Parciales (Keywords%)	1,27%	1,17%	1,16%	0,71%	1,75%	1,03%	1,18%	0,34%
Errores 404	1,34%	1,77%	1,00%	0,69%	0,24%	0,89%	0,99%	0,53%
H1 Ocurrencias Parciales (Keywords%)	1,12%	0,86%	1,46%	0,54%	1,27%	0,58%	0,97%	0,38%
Uso Https	0,10%	0,11%	3,09%	0,00%	0,08%	0,11%	0,58%	1,23%
Sitemap.xml	0,76%	0,32%	0,36%	0,23%	0,29%	1,30%	0,54%	0,42%
Datos Estructurados	0,54%	0,14%	0,19%	0,63%	0,50%	0,38%	0,40%	0,20%
Robots.txt	0,34%	0,38%	0,31%	0,69%	0,18%	0,29%	0,36%	0,17%
Videos Totales	0,14%	0,28%	0,70%	0,31%	0,27%	0,21%	0,32%	0,20%
Url Amigable	0,34%	0,23%	0,14%	0,12%	0,34%	0,71%	0,31%	0,22%
Compatibilidad Móvil	0,46%	0,15%	0,22%	0,06%	0,33%	0,23%	0,24%	0,14%
Schema Markup	0,17%	0,19%	0,06%	0,11%	0,15%	0,23%	0,15%	0,06%

Tabla 6.7: Tabla de pesos calculados para cada parámetro SEO - Clasificador General

Fuente: Elaboración propia.

	Pesos generados por el Clasificador Top 10 (2 clases)							
	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5	Prueba 6	Media	Desviación
Url Longitud	6,65%	8,82%	26,41%	8,28%	6,77%	9,49%	11,07%	7,60%
Dias Vivo	2,78%	5,87%	6,54%	9,18%	7,65%	4,41%	6,07%	2,28%
Cuerpo Ocurrencias Parciales (Keywords%)	7,20%	7,19%	3,68%	3,07%	10,43%	1,57%	5,52%	3,31%
Descripción Longitud	7,25%	1,41%	4,76%	3,41%	4,38%	9,78%	5,17%	2,95%
Cuerpo Ocurrencias Totales (Keywords%)	8,41%	3,08%	4,25%	5,93%	3,51%	3,74%	4,82%	2,02%
Imágenes Totales	1,09%	6,74%	3,84%	4,20%	4,33%	6,24%	4,41%	2,01%
Errores Conexión	2,61%	4,71%	1,94%	5,76%	3,90%	6,66%	4,26%	1,81%
Velocidad Carga Móvil	1,20%	8,02%	1,97%	4,73%	3,83%	3,48%	3,87%	2,40%
Uso Https	2,25%	0,00%	17,31%	0,00%	0,02%	2,26%	3,64%	6,79%
Rendimiento Móvil	4,60%	4,59%	2,02%	2,16%	4,18%	4,07%	3,60%	1,20%
Velocidad Carga Sobremesa	2,76%	4,16%	2,38%	4,57%	2,07%	4,63%	3,43%	1,15%
Links Totales	2,35%	2,43%	2,01%	3,85%	4,89%	4,05%	3,26%	1,16%
Rendimiento Sobremesa	4,83%	2,51%	0,47%	2,39%	3,25%	3,93%	2,90%	1,50%
Título Longitud	3,49%	2,50%	1,95%	3,15%	3,20%	1,92%	2,70%	0,68%
Descripción Ocurrencias Totales (Keywords%)	4,50%	2,45%	1,48%	4,45%	0,71%	1,30%	2,48%	1,64%
Alt Ocurrencias Parciales (Keywords%)	2,04%	4,50%	1,63%	0,85%	3,64%	1,84%	2,42%	1,37%
Follow Externos	2,66%	2,32%	1,04%	3,24%	3,13%	1,90%	2,38%	0,83%
Descripción Ocurrencias Parciales (Keywords%)	3,67%	6,25%	0,53%	0,52%	2,15%	1,17%	2,38%	2,24%
Src Ocurrencias Parciales (Keywords%)	2,41%	0,80%	0,41%	2,59%	5,85%	1,68%	2,29%	1,94%
Links Externos	0,90%	0,19%	1,80%	4,99%	4,21%	1,50%	2,26%	1,91%
NoFollow Externos	1,18%	1,12%	1,24%	1,63%	3,28%	4,46%	2,15%	1,39%
Follow Internos	3,44%	3,11%	0,95%	2,13%	0,54%	2,00%	2,03%	1,15%
H1 Ocurrencias Totales (Keywords%)	2,58%	1,65%	1,48%	0,99%	1,13%	4,21%	2,01%	1,21%
Src Ocurrencias Totales (Keywords%)	2,10%	0,42%	2,35%	3,43%	2,48%	0,77%	1,92%	1,13%
Alt Ocurrencias Totales (Keywords%)	0,97%	0,73%	1,30%	5,13%	1,12%	1,88%	1,85%	1,65%
Errores 404	1,64%	2,68%	0,02%	1,53%	1,98%	1,75%	1,60%	0,88%
Links Internos	2,68%	4,05%	0,64%	0,46%	0,12%	1,05%	1,50%	1,54%
Título Ocurrencias Totales (Keywords%)	2,69%	0,69%	1,49%	2,21%	1,01%	0,64%	1,45%	0,84%
NoFollow Internos	1,17%	2,60%	0,44%	2,13%	0,83%	0,14%	1,22%	0,96%
Sitemap.xml	1,74%	0,05%	0,08%	0,00%	0,18%	5,10%	1,19%	2,03%
Cabeceras	2,20%	0,00%	1,01%	2,22%	0,61%	0,87%	1,15%	0,89%
H1 Ocurrencias Parciales (Keywords%)	1,38%	0,22%	0,63%	0,15%	2,93%	0,62%	0,99%	1,05%
Título Ocurrencias Parciales (Keywords%)	0,81%	2,24%	1,49%	0,08%	0,74%	0,27%	0,94%	0,81%
Videos Totales	1,41%	0,70%	0,09%	0,02%	0,15%	0,02%	0,40%	0,56%
Robots.txt	0,29%	0,45%	0,00%	0,00%	0,42%	0,00%	0,19%	0,22%
Url Amigable	0,03%	0,70%	0,00%	0,10%	0,01%	0,15%	0,17%	0,27%
Schema Markup	0,00%	0,00%	0,17%	0,15%	0,40%	0,00%	0,12%	0,16%
Datos Estructurados	0,04%	0,02%	0,00%	0,24%	0,00%	0,35%	0,11%	0,15%
Compatibilidad Móvil	0,02%	0,03%	0,20%	0,07%	0,00%	0,09%	0,07%	0,07%

Tabla 6.9: Tabla de pesos calculados para cada parámetro SEO - Clasificador Top 10

Fuente: Elaboración propia.

6.2.3. Estimación de valores recomendados

Al contrario de los pesados, que hacen uso de un clasificador automático, la recomendación de valores es algo que se ha calculado siguiendo una serie de pautas personalizadas, basadas en las medias, desviaciones y diferencias obtenidas de los resultados extraídos (especialmente de los mejor posicionados). A pesar de esto, en la tabla 6.11 se pueden encontrar una gran cantidad de parámetros que parecen sugerir una tendencia común en la gran mayoría de las pruebas, lo cual puede ser utilizada para obtener recomendaciones más globales.

A partir de la tabla se han sacado las siguientes observaciones:

- Las distintas longitudes recomendadas parecen moverse en torno a unos valores muy cerrados. La longitud de la URL parece moverse entre los 30 y 70 caracteres (la prueba 6 parece desviarse ligeramente), el título entre los 30 y 65 caracteres, y la descripción entre los 100 y 180 caracteres.

- Aunque el rango de valores para los días vividos varían de manera considerable, es importante destacar que incluso para los resultados más bajos, el rango inferior no llega nunca a bajar de los 9 años. Además, los valores medios rondan los 13 y 18, unos resultados similares a los obtenidos por *Urosa Barreto* (Tabla 2.1) .

- En el caso de algunos de los parámetro menos recomendados en la sección anterior, como es el caso de uso de https, URL amigable, robots.txt y compatibilidad móvil, parece ser que su baja importancia en el pesado es debida a que la gran mayoría de las webs, independientemente de su ranking, hacía uso de ellas, por lo que es posible que en realidad su uso sea indispensable. Por otro lado, los sitemaps, datos estructurados y marcado Schema, se podrían considerar de menor importancia, ya que sus recomendaciones como 'indiferente' se debe a que la gran mayoría de webs con un alto posicionamiento no hacen uso de ellas (en esa selección).

- Para el rendimiento y velocidad de carga también se obtienen unos resultados bastantes sólidos. En el caso del móvil, su velocidad de carga parece rondar los 10 segundos, y su rendimiento apenas parece llegar al aprobado. Por otro lado, los dispositivos sobremesa parecen tener un rendimiento notablemente mayor el cual ronda los 80 puntos, y una velocidad de carga menor que ronda los 2.5 segundos (también se asemeja a lo obtenido por *Urosa Barreto*).

- Las webs mejor posicionadas parece incorporar 2 o 3 cabeceras.

- El uso de vídeos en webs bien posicionadas no parece ser común. En el caso de las imágenes, aunque siempre presentes, no se ha podido sacar conclusiones globales respecto a su cantidad.

- En el caso de uso de los links, aunque no se puede sacar un valor recomendado concreto para la cantidad de cada uno de ellos, si que es cierto que las webs con buen posicionamiento tienden a utilizar centenares de enlaces, siendo, en su gran mayoría, enlaces Follow internos.

- Aun con un gran número de enlaces, hay que hacer todo lo posible para mantener el número de errores (404 y de conexión) al mínimo. El motivo de que los resultados de la prueba 2 sean tan diferentes se debe a que una de las URLs posicionada en el segundo lugar de una query, generó 1499 errores de conexión, algo que también demuestra que el criterio de selección de parámetros recomendados utilizado es débil ante valores extremos.

- La aparición/densidad de las keywords en la etiqueta Src es, prácticamente, nula. La aparición en el resto de las etiquetas será comentada en más detalle en la siguiente sección.

	Valores recomendados obtenidos					
	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5	Prueba 6
Url Longitud	[40-67]	[32-52]	[33-55]	[43-72]	[41-68]	[49-82]
Título Longitud	[33-55]	[38-63]	[32-54]	[39-64]	[36-60]	[33-54]
Descripción Longitud	[113-188]	[105-175]	[91-151]	[111-185]	[108-179]	[109-182]
Dias Vivo	[3768-6280]	[3792-6319]	[5214-8690]	[3399-5665]	[5073-8455]	[5307-8844]
Uso Https	Sí	Sí	Sí	Sí	Sí	Sí
Url Amigable	Sí	Sí	Sí	Sí	Sí	Sí
Robots.txt	Sí	Sí	Sí	Sí	Sí	Sí
Sitemap.xml	Indiferente	Sí	Sí	Sí	Indiferente	Indiferente
Compatibilidad Móvil	Sí	Sí	Sí	Sí	Sí	Sí
Rendimiento Móvil	[47-100]	[49-100]	[41-100]	[53-100]	[57-100]	[45-100]
Velocidad Carga Móvil	[0-12.8]	[0-9.7]	[0-10.6]	[0-7.0]	[0-7.6]	[0-13.7]
Rendimiento Sobremesa	[74-100]	[77-100]	[71-100]	[79-100]	[84-100]	[75-100]
Velocidad Carga Sobremesa	[0-2.4]	[0-2.1]	[0-3.0]	[0-1.9]	[0-1.5]	[0-2.8]
Cabeceras	[2-3]	[2-4]	[2-4]	[3-4]	[3-4]	[2-3]
Datos Estructurados	Indiferente	Indiferente	Sí	Sí	Sí	Sí
Schema Markup	Indiferente	Indiferente	Sí	Sí	Sí	Sí
Imágenes Totales	[31-52]	[56-93]	[40-67]	[24-40]	[22-36]	[15-25]
Videos Totales	[0-1]	[0-1]	[0-1]	[0-1]	[0-1]	[0-1]
Links Totales	[287-478]	[283-472]	[177-295]	[85-142]	[177-294]	[252-419]
Links Internos	[220-367]	[271-451]	[166-277]	[69-116]	[153-255]	[216-360]
Links Externos	[66-111]	[12-21]	[11-18]	[16-27]	[23-39]	[36-59]
Follow Internos	[220-366]	[255-425]	[157-262]	[68-114]	[153-255]	[216-360]
Follow Externos	[14-24]	[11-19]	[9-16]	[14-23]	[15-25]	[15-25]
NoFollow Internos	[0-1]	[16-27]	[9-15]	[0-1]	[0-1]	[0-1]
NoFollow Externos	[52-87]	[0-1]	[0-1]	[2-4]	[9-14]	[20-34]
Errores 404	[0-5]	[0-6]	[0-1]	[0-1]	[0-3]	[0-6]
Errores Conexión	[0-39]	[0-296]	[0-28]	[0-6]	[0-6]	[0-9]
Título Ocurrencias Totales (Keywords%)	[15.73%-26.21%]	[13.54%-22.56%]	[20.8%-34.66%]	[15.51%-25.86%]	[14.54%-24.23%]	[10.14%-16.9%]
Título Ocurrencias Parciales (Keywords%)	[6.83%-11.39%]	[4.34%-7.23%]	[3.18%-5.3%]	[0.73%-1.21%]	[4.74%-7.9%]	[2.15%-3.59%]
Descripción Ocurrencias Totales (Keywords%)	[3.73%-6.22%]	[4.93%-8.22%]	[7.2%-12.0%]	[5.67%-9.45%]	[3.65%-6.09%]	[3.15%-5.25%]
Descripción Ocurrencias Parciales (Keywords%)	[4.9%-8.17%]	[2.47%-4.12%]	[1.51%-2.52%]	[0.6%-0.99%]	[2.88%-4.8%]	[0%-1%]
H1 Ocurrencias Totales (Keywords%)	[17.39%-28.99%]	[19.39%-32.31%]	[31.58%-52.64%]	[20.67%-34.45%]	[33.07%-55.12%]	[27.23%-45.38%]
H1 Ocurrencias Parciales (Keywords%)	[5.77%-9.62%]	[2.94%-4.9%]	[3.61%-6.02%]	[2.57%-4.28%]	[9.12%-15.19%]	[5.55%-9.24%]
Alt Ocurrencias Totales (Keywords%)	[2.26%-3.76%]	[1.14%-1.9%]	[1.1%-1.84%]	[6.72%-11.2%]	[2.01%-3.35%]	[2.78%-4.64%]
Alt Ocurrencias Parciales (Keywords%)	[1.58%-2.63%]	[0.96%-1.6%]	[3.12%-5.2%]	[0.9%-1.5%]	[0.11%-0.18%]	[0.45%-0.76%]
Src Ocurrencias Totales (Keywords%)	[0.52%-0.87%]	[0.21%-0.35%]	[0.09%-0.15%]	[0.7%-1.17%]	[0.15%-0.25%]	[0%-1%]
Src Ocurrencias Parciales (Keywords%)	[0.52%-0.86%]	[0.68%-1.13%]	[0.67%-1.11%]	[0.35%-0.58%]	[0.37%-0.62%]	[0.32%-0.53%]
Cuerpo Ocurrencias Totales (Keywords%)	[1.38%-2.3%]	[1.1%-1.83%]	[1.28%-2.13%]	[1.74%-2.9%]	[1.23%-2.05%]	[0.63%-1.05%]
Cuerpo Ocurrencias Parciales (Keywords%)	[3.11%-5.19%]	[1.74%-2.89%]	[1.05%-1.75%]	[0.42%-0.7%]	[0.98%-1.63%]	[0.48%-0.8%]

Tabla 6.11: Tabla de valores recomendados para cada parámetro SEO

Fuente: Elaboración propia.

6.3 Análisis de aparición de keywords

La última prueba realizada en este TFG es el análisis de importancia del uso de las keywords (Objetivo secundario 1). Para ello se compararán los resultados obtenidos sobre la densidad y probabilidad de aparición de keywords en las muestras finales (Anexo A.3), con los obtenidos en trabajos más enfocados al estudio de keywords, como es el caso de las tesis de *Urosa Barreto* y *Carreras Lario* mencionadas en el capítulo 2.

Para calcular la probabilidad de aparición de las keywords se han seguido las siguientes pautas:

- De manera similar al de las tesis, se han utilizado como datos representativos las primeras tres posiciones de cada query (180 URLs).

- En el caso de las queries compuestas, se ha tenido en cuenta la aparición de cada keyword de manera separada.

- Al contrario de lo que se representa en las tablas, la densidad y la probabilidad de aparición parcial de una keyword también tomará en cuenta la aparición total de la misma. (parcial=parcial+total).

- Si una URL en el Top 3 no ha podido ser analizada, se usará la más próxima en su lugar.

Esto ha dado lugar a los siguientes resultados:

	Ariel Martínez	**Urosa Barreto**	**Carreras Lario**
Densidad de keywords	1,35 % (3.07 % parcial)	1,38 %	2,35 %
Probabilidad de aparición keyword (Título)	72 % (88 % parcial)	81 %	93,3 %
Probabilidad de aparición keyword (Descripción)	64 % (81 % parcial)	73 %	-
Probabilidad de aparición keyword (H1)	64 % (72 % parcial)	75 %	76,6 % (h1 y/o h2)
Probabilidad de aparición keyword (Alt)	29 % (37 % parcial)	11 %	34,8 %

Tabla 6.13: Comparativa de aparición de keywords obtenida con las conclusiones de *Urosa Barreto* y *Carreras Lario*

Fuente: Elaboración propia.

Como se observa en la taba, los resultados obtenidos de nuestras muestras parecen ser bastante similares a los obtenidos en los otros estudios, especialmente en el caso de la densidad de keywords, donde el resultado de este trabajo apenas dista de unas milésimas al obtenido por *Urosa*.

Aunque se observe una ligera diferencia en las probabilidades de aparición, estas podrían ser fruto del uso de queries compuestas, donde, a mayor número de keywords, menor será la probabilidad que aparezcan todas ellas debido a la escasa longitud de las etiquetas analizadas. También es probable que la selección de keywords haya influenciado, ya que es posible que algunas de las palabras (keywords) utilizadas no sean las más

comunes dentro de su familia léxica. Por ejemplo, si se tiene en cuenta la probabilidad de aparición parcial, nuestros datos, alcanzan, o superan, el de los otros estudios, algo a tener en cuenta para los títulos y las cabeceras h1 que, debido su corta longitud, no suelen incorporar dos palabras de la misma familia léxica.

Independientemente de las diferencias, es evidente en todos los estudios que la aparición de las keywords en el título, la descripción y las cabeceras h1, de las páginas mejor posicionadas es extremadamente común, haciendo visible su importancia en el posicionamiento. En el caso de las etiquetas Alt ningún estudio parece haber obtenido indicios de que su uso sea importante.

Desgraciadamente, aunque también se ha intentado sacar conclusiones respecto al posicionamiento de las keywords, la varianza observada en los resultados de las muestras finales parece indicar que, en un principio, el posicionamiento de las keywords dentro del cuerpo de una web es indiferente. A pesar de esto, hay que tener en cuenta que esto solo se aplica si se mantiene una densidad de keywords moderada y el texto escrito tiene alguna lógica, ya que estos parámetros sí que son tomados en cuenta por el buscador y tienen una fuerte relación con el posicionamiento de las keywords.

CAPÍTULO 7
Conclusiones

En el presente trabajo nos hemos marcado como objetivo la elaboración de un programa en Python capaz de extraer y analizar de manera automática los parámetros que influyen en el posicionamiento del buscador de Google. Tras validar la funcionalidad del programa, realizar el análisis de 39 parámetros de posicionamiento para 9000 URLs (para un total de 351.000 parámetros, menos los que produjeron error durante la extracción), y analizar, de manera aislada, la tasa de aparición de keywords, se han obtenido las siguientes conclusiones:

- Se ha completado satisfactoriamente la elaboración de un código capaz de extraer y analizar, para un número arbitrario de resultados y queries, un total de 39 parámetros que afectan al posicionamiento de páginas web. A pesar de esto, el programa no es ni mucho menos perfecto, y se podría mejorar la extracción de algunos elementos como las descripciones (de las cuales no siempre se obtiene el resultado esperado) y las fechas de creación (el cual es propenso a no obtener resultados). Aunque ha sido imposible incluirlos debido a la complejidad y tiempo que supondría, también se podría mejorar la presencia de una mayor cantidad de parámetros off-page como los backlinks.
- Se ha conseguido realizar, mediante el uso de dos clasificadores Gradient Boosting, las estimaciones de importancia de los distintos parámetros extraído tanto de manera local (una ejecución), como de manera global, llegándose incluso a recolectar el grupo de parámetros que más afectan al posicionamiento. Cabe destacar que para lograr una buena precisión de dichas estimaciones, es recomendable utilizar un número alto de queries y de resultados.
- Los valores recomendados que genera el programa cumplen con las expectativas creadas al inicio del trabajo, pero eso es solo en el caso de que se utilice un número considerable de queries y en el caso de que no haya valores extremos, los cuales podrían arruinar completamente las recomendaciones de algún parámetro. La necesidad de diversas queries es algo razonable teniendo en cuenta que con solo una o dos sería imposible obtener resultados objetivos, pero en el caso de los valores extremos sí que sería necesario, en un futuro, buscar la forma de eliminar su influencia dentro de los valores recomendados.
- El programa elaborado permite la comparación de los datos extraídos de una selección de queries, con los de una URL especifica, usando el pesado y los valores recomendados obtenidos para generar una puntuación final. Desgraciadamente, esto también significa que si obtiene algún error en los pesados o los valores recomendados (como los valores extremos), esto afectará negativamente a la puntuación generada.

- El estudio independiente del uso de keywords, aunque reducido en el número de parámetros investigados, ha conseguido mostrar que el uso de keywords (parcial o total) en el título, la descripción, y la cabecera h1 tienden a estar presentes en las páginas web mejor clasificadas. También se han llegado a conclusiones sobre la densidad de keywords en el cuerpo de una web, muy similares a las obtenidas en estudios de otros autores.
- El uso de la biblioteca *xlsxwriter* ha conseguido plasmar los resultados extraídos y analizados por el programa de una manera sencilla y organizada.

7.1 Relación del trabajo con los estudios cursados

Sin lugar a dudas, este trabajo ha sido una prueba de fe de mis habilidades como informático, en el cual se ha intentado coger todo lo aprendido durante mi formación y utilizarlo en un entorno, en el que personalmente, soy prácticamente un desconocido.

Debido a la lenta curva de aprendizaje característica de trabajar en un campo totalmente nuevo, como lo es el SEO para mí, las horas invertidas en este trabajo no fueron pocas. Durante el desarrollo era común que me surgían nuevas ideas que, o acababan en la nada, o multiplicaban el tiempo de investigación y desarrollo necesario.

Afortunadamente, entre todas las dificultades que conlleva la escasez de conocimiento previo, también se pueden sacar cosas positivas como la detección y desarrollo de carencias que previamente desconocía, o el aprendizaje de una gran variedad de nuevos recursos y herramientas, los cuales no habría obtenido trabajando en un entorno más familiar.

Cabe mencionar que, aunque el área de investigación de este trabajo me sea novedosa y se acerque más a las Tecnologías de la Información que a la Computación, eso es solo de manera superficial. En este trabajo además de las técnicas más esenciales para cualquier informático, como el paralelismo o el uso de interfaces gráficas, también se han utilizado técnicas y herramientas propias de la computación como son: la búsqueda (automática) de palabras clave, algoritmos para la distancia de edición, técnicas de aprendizaje automático (como el clasificador), y el análisis automático de datos en general.

CAPÍTULO 8
Trabajos futuros

Mediante la implementación de los algoritmos descritos en el capítulo 5, se ha logrado creador un programa capaz de extraer y analizar 39 factores SEO de manera directa. Desafortunadamente, dicho programa no es ni mucho menos perfecto, especialmente en lo que se refiere a la precisión del clasificador.

Para alcanzar un mayor nivel de calidad, se han planteado una serie de ampliaciones posibles que podrían ser incluidas en un futuro. Dichas mejoras son:

- **Ampliación de parámetros SEO:** Como se menciona al principio del documento, se estima que el algoritmo de posicionamiento de Google cuenta con más de un centenar de parámetros. En este trabajo solo se han cubierto algunos de los que se consideran más importantes, pero la inclusión de parámetros de menor importancia podría ayudar ligeramente a optimizar los resultados del programa. También, aunque mucho más complejo, se podría intentar elaborar una base de datos que incluya información de parámetros SEO off-page de gran importancia los cuales no han podido ser incluidos, como por ejemplo los backlinks.
- **Ocultar de la IP:** Uno de los mayores defectos del programa es su alto coste temporal para búsquedas de tamaño considerable. Esto se debe a la imposibilidad de aumentar el paralelismo debido a la aparición del error 429, la cual podría ser evitada si se incluye algún elemento, simulando una VPN, que pudiera cambiar el identificador del ordenador usado para lanzar el programa.

La implementación de ambas mejoras cubriría los dos grandes flecos que presenta el código actual: la precisión del clasificador y el tiempo de ejecución.

Lógicamente existen un mayor número de mejoras posibles con relación a otros temas como la visualización (mejora de GUI) o la implantación del programa (de ejecutable a aplicación web), pero debido a que este trabajo se centra en la funcionalidad del programa, no se han considerado lo suficientemente importantes como para ser incluidos en más detalle.

Bibliografía

[1] Juan Carlos Mancha Ramos. (2021, diciembre 16). Evolución del posicionamiento SEO. Consultar en: https://www.disegnowebologna.com/marketing/evolucion-del-posicionamiento-seo/.

[2] Matt G. Southern. (2020, octubre 14). Conoce los principales 172 factores de posicionamiento en Google y cómo influyen en el SEO de las páginas web. Consultar en: https://rockcontent.com/es/blog/factores-de-posicionamiento-en-google/.

[3] André Mousinho. (2020, julio 14). Over 25 % of People Click the First Google Search Result. Consultar en: https://www.searchenginejournal.com/google-first-page-clicks/374516/#close.

[4] Greg Snow-Wasserman. On-Page vs. Off-Page SEO: What's the Difference? Consultar en: https://www.woorank.com/en/blog/on-page-vs-off-page-seo-whats-the-difference#:~:text=On%2Dpage%20SEO%20refers%20to,or%20brand%20around%20the%20web.

[5] Google developers Team. Cómo funcionan los datos estructurados Consultar en: https://developers.google.com/search/docs/advanced/structured-data/intro-structured-data.

[6] Nate Drake, Brian Turner, Desire Athow, March. (2022, marzo 03). Website intelligence for Sales Professionals and Market Analysts. Consultar en: https://www.techradar.com/news/best-seo-tool.

[7] Miguel Florido. (2015, octubre 07). 20 Herramientas SEO para mejorar tu posicionamiento. Consultar en: https://es.semrush.com/blog/herramientas-posicionamiento-seo/.

[8] Website intelligence for Sales Professionals and Market Analysts. Consultar en: https://www.woorank.com/en/solutions/data-services.

[9] Jason Brownie. (2019, agosto 21). Comparing 13 Algorithms on 165 Datasets, de Machine Learning Mastery. Consultar en: https://machinelearningmastery.com/start-with-gradient-boosting/.

[10] Jason Brownie. (2020, mayo 04). How to Develop a Gradient Boosting Machine Ensemble in Python, de Machine Learning Mastery. Consultar en: https://machinelearningmastery.com/gradient-boosting-machine-ensemble-in-python/.

[11] Jason Brownie. (2020, diciembre 28). Histogram-Based Gradient Boosting Ensembles in Python, de Machine Learning Mastery. Consultar en: https://machinelearningmastery.com/histogram-based-gradient-boosting-ensembles/.

[12] Jason Brownie. (2016, septiembre 09). Histogram-Based Gradient Boosting Ensembles in Python, de Machine Learning Mastery. Consultar en: https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/.

[13] Jerome H. Friedman. (1999, marzo 26). Stochastic Gradient Boosting. Consultar en: https://jerryfriedman.su.domains/ftp/stobst.pdf.

[14] Jerome H. Friedman. (2001, abril 19). Greedy Function Approximation: A Gradient Boosting Machine. Consultar en: https://jerryfriedman.su.domains/ftp/trebst.pdf.

[15] D. Félix Urosa Barreto. (2020). Posicionamiento orgánico en buscadores (SEO): Estudio sobre el ranking de resultados en Google.es del sector educativo superior. Consultar en: https://eprints.ucm.es/id/eprint/64419/1/T42152.pdf.

[16] D. Ricardo Carreras Lario. (2012). CÓMO CLASIFICA GOOGLE LOS RESULTADOS DE LAS BÚSQUEDAS: FACTORES DE POSICIONAMIENTO ORGÁNICO. Consultar en: https://eprints.ucm.es/id/eprint/17450/1/T34083.pdf.

[17] Alejandro Alós Moya. (2011). GUÍA PARA EL POSICIONAMIENTO WEB. Consultar en: https://riunet.upv.es/bitstream/handle/10251/12063/Memoria.pdf?sequence.

[18] Jet Brains. (2021). The State of Developer Ecosystem 2021. Consultar en: https://www.jetbrains.com/lp/devecosystem-2021/.

[19] Google developers Team. Simplificar la estructura de las URLs. Consultar en: https://developers.google.com/search/docs/advanced/guidelines/url-structure?hl=es.

[20] Álvaro Fontela. (2020, julio 20). Core Web Vitals / Google Search Console – LCP, CLS, FID. Consultar en: https://alvarofontela.com/core-web-vitals-google-search-console-lcp-cls-fid/.

[21] Naciones Unidas. de Desarrollo Sostenible. Consultar en: https://www.un.org/sustainabledevelopment/es/

APÉNDICE A
Muestras de resultados generados

Debido al tamaño de los resultados generados, ha sido imposible incluirlos de manera orgánica en el apéndice. Por ello, se han recolectado los resultados utilizados en las pruebas del programa dentro de carpetas de Google Drive.

A continuación, se mostrará los enlaces a cada grupo de resultados.

A.1 Muestra de ejemplo completa

Muestra utilizada como ejemplo en la prueba de funcionalidad.

URL: https://drive.google.com/drive/folders/1gsLWPtARsCCufOVO1mOVI4Z11Bk9GyKb?usp=sharing

A.2 Muestras iniciales - número de búsquedas

Muestras utilizadas para seleccionar el número de búsquedas para la pruebas de análisis.

URL: https://drive.google.com/drive/folders/1wkuxNx_lqSzbE-fRPs2H5RwOz6ZWl9lc?usp=sharing

A.3 Muestras Finales - análisis de resultados

Muestras utilizadas para las pruebas de análisis.

URL: https://drive.google.com/drive/folders/1MuOrGkyDWmrsDFXr-mWD9ROn5FjtDgbJ?usp=sharing

APÉNDICE B
Código y algoritmos adicionales

A continuación, se mostraran algunos segmentos secundarios del código pero que son vitales para la ejecución del mismo.

- **Extracción de URLs:** (Figura B.1) Segmento de código utilizado para extraer las URLs a analizar de una query concreta.

  q = query donde buscar los resultados.

  num = número de búsquedas por query.

- **Entrenamiento del clasificador:** (Figura B.3) Moduló el cual es utilizado para el entrenamiento de los dos clasificadores. Las variables utilizadas vienen definidas en la figura.

- **Cálculo de valores recomendados y nivel de erro:** (Figura B.5) Segmento de código utilizado para calcular los valores recomendados de cada figura y, en caso de que se seleccione comparar una URL, calcular el nivel de error de sus parámetros respecto a los recomendados.

  rVal[cu] = Valor recomendado *cu*.

  prioridad[cu] = nivel de error para el parámetro en posición *cu*.

```
#Creamos los Headers
    q = q.replace(' ', '+')
    URL = "https://google.com/search?q="+q  + "&num=10"
    USER_AGENT = "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:65.0) Gecko/20100101 Firefox/65.0"
    headers = {"user-agent" : USER_AGENT, "Cache-Control": "no-cache", "Pragma": "no-cache"}

    #Primera p gina de google search
    prevURL=""
    numg=0
    nplus=0
    ListGdes=[]
    ListGdes1=[]
    ListNumg=[]

    try:
        #Sacamos el HTML de google search
        resp = requests.get(URL, headers=headers, timeout=60)
        if resp.status_code == 200:
            soupg = BeautifulSoup(resp.content, "html.parser")
        else:
            num=0

        #Busqueda URLs
        URL2=URL
        while(numg<num):
            #Buqueda de URLs
            count=0
            for gdes in soupg.find_all("div", {"class":["tF2Cxc","jtfYYd"]}):
                if(numg==num):
                    break
                gdes1=gdes.find_all("a")
                gdes1=gdes1[0].get("href")

                #Evitar Sitelinks
                if(gdes1==prevURL or urlparse(gdes1).netloc == urlparse(prevURL).netloc):
                    continue
                prevURL=gdes1

                ListGdes.append(gdes)
                ListGdes1.append(gdes1)
                ListNumg.append(numg)

                numg=numg+1
                count=count+1

            if count==0:
                num=numg
            #Pasamos de Pagina en Google Search si es necesario
            if(numg<num):
                nplus=nplus+10
                URL2=URL + "&start=" + str(nplus)
                resp = requests.get(URL2, headers=headers, timeout=60)
                if resp.status_code == 200:
                    soupg = BeautifulSoup(resp.content, "html.parser")
                else:
                    numg=num

    except Exception:
        #Some URLs Not Found
        pass
```

Figura B.1: Extracción de URLs

Fuente: Elaboración propia.

```
def clasificador(numM,DT,DY):

    """
    Num ejecuciones: 2

    Funci n para crear el clasificador gradientBoosting que estimar  los pesados de la variables y la URL del
      informe.

    param:  "numM": Array que musetra el numero de etiquetas de cada clase. Clase i de DT y DY = a clase i-1.
            "DT": Tabla de datos.
            "DY": Tabla de etiquetas.

             return: "clf": Clasificador obtenido.
                     "cvStr2": Precisi n del clasificador (Str).
    """

    #Clasificador
    auxCero=[0]*39
    auxCero=[auxCero]
    numMinF=numM[0]
    if(numMinF==0):numMinF=numM[1]
    for i in numM:
        if(i>0 and i<numMinF): numMinF=i

    if(numMinF>=5):
        cv = RepeatedStratifiedKFold(n_splits=5, n_repeats=3, random_state=1)
        for _ in range(5):
            DT = np.concatenate((DT, auxCero))
            DY = np.concatenate((DY, [0]))
    else:
        cv = RepeatedStratifiedKFold(n_splits=numMinF, n_repeats=3, random_state=1)
        for _ in range(numMinF):
            DT = np.concatenate((DT, auxCero))
            DY = np.concatenate((DY, [0]))


    learnR=0
    n_est=0

    if(numMinF==1):
        if(idioma): cvStr2 = "Classifier Accuracy: Unavilable"
        else: cvStr2 = "Precisi n del Clasificado: No Disponible"
        clf2 = GradientBoostingClassifier()

    else:
        crossVS = np.array([[0], [0]])
        crossi=[0.1,0.25,0.5,0.75,1]
        crossj=[100,250,500,750,1000]

        for i in crossi:
            for j in crossj:

                clf3 = GradientBoostingClassifier(learning_rate=i,n_estimators=j)
                crossVS2 = cross_val_score(clf3, DT, DY, scoring='accuracy',cv=cv)
                if(np.mean(crossVS2)>np.mean(crossVS)):
                    crossVS=crossVS2
                    clf2=clf3
                    learnR=i
                    n_est=j


        if(idioma):
            cvStr2= "Classifier Accuracy: " + str(round(np.mean(crossVS),3)) + " (Deviation: " + str(round(np.
     std(crossVS),3)) + ")"
        else:
            cvStr2= "Precisi n del clasificador: " + str(round(np.mean(crossVS),3)) + " (Desviaci n: " + str(
     round(np.std(crossVS),3)) + ")"

    return clf2.fit(DT,DY), cvStr2
```

Figura B.3: Entrenamiento del clasificador

Fuente: Elaboración propia.

```
def informe(cvStr,clf,cvStr10,clf10,numClases,numClases10,clfList,workbook,worksheetI,mediaS,mediaT,aTable,
    a2Table,desviaci nT,idioma):

    """
    Num ejecuciones: 1

    Funci n para crear el apartado informe del xslx para la URL/Query principal a analizar.

    param:  "workbook": Woorkbook xlsx en el que mostrar los datos.
            "worksheetI": Woorksheet para el Informe dentro del Workbook.
            "mediaS": Medias Total de todas las queries limitado por el Ranking inferior l mite (Mejores
    resultados).
            "mediaT": Media Total de todas las queries.
            "aTable": Variables SEO de la URL/Query principal (acondicionado para ilustrarlo en xslx).
            "a2Table": Variables SEO de la URL/Query principal (acondicionado para calculos).
            "idioma": Idioma seleccionado (Ingles/Espa ol).
            "desviaci nT": Array de desviaci n  t pica.

    """


    prioridad = [0]*44
    prioridadMax = [0]*44
    rVal = [0]*44

    #0-1
    ceroUno=[9,10,11,12,13,19,20]
    for cu in ceroUno:
        rVal[cu]="Yes"
        if(mediaS[cu]>0.9 or mediaS[cu]-mediaT[cu]>0.3):
            prioridad[cu]=5
        elif(mediaS[cu]>0.75 or mediaS[cu]-mediaT[cu]>0.2):
            prioridad[cu]=4
        elif(mediaS[cu]>0.6 or mediaS[cu]-mediaT[cu]>0.1):
            prioridad[cu]=3
        elif(mediaS[cu]>0.5):
            prioridad[cu]=2
        else:
            prioridad[cu]=0
            if(idioma):
                rVal[cu]="Indifferent"
            else:
                rVal[cu]="Indiferente"

        prioridadMax[cu]=prioridad[cu]
        if a2Table[cu] == 1 and (not (prioridad[cu] == 0)):
            prioridad[cu]=1


    #int
    numer=[2,4,6,8,14,15,16,17,18,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43]
    for cu in numer:

        roundaux=0
        if cu in [15,17]:
            roundaux=1
        elif cu in [32,33,34,35,36,37,38,39,40,41,42,43]:
            roundaux=2
        difMed=round(mediaS[cu],roundaux)
        dif=round(abs(a2Table[cu]-mediaS[cu]),roundaux)


        if(dif<= difMed*0.25):
            prioridad[cu]=1
        elif(dif<= difMed*0.45):
            prioridad[cu]=2
        elif(dif<= difMed*0.65):
            prioridad[cu]=3
        elif(dif<= difMed*0.85):
            prioridad[cu]=4
        else:
            prioridad[cu]=5

        dt=round(desviaci nT[cu],roundaux)
        if(dt<= difMed*0.45):
            dtp=5
        elif(dt<= difMed*0.65):
            dtp=4
        elif(dt<= difMed*0.85):
            dtp=3
        elif(dt<= difMed*0.95):
            dtp=2
        else:
            if cu in [26,27,30,31,32,33,34,35,36,37,38,39,40,41,42,43]:
                dtp=2
            else:
                dtp=0

        prioridadMax[cu]=dtp

        if(dtp<prioridad[cu]): prioridad[cu] = dtp




        raux = mediaS[cu]-difMed*0.25
        raux2 = mediaS[cu]+difMed*0.25


```

```
        if difMed == 1 or difMed == 0:
            if dif <= 1 and not (prioridad[cu]==0):
                prioridad[cu]=1
            raux =0
            raux2=1

        if cu in [14,16]:
            if a2Table[cu] > mediaS[cu]-difMed*0.05:
                prioridad[cu]=1
            elif(a2Table[cu] > mediaS[cu]-difMed*0.10):
                prioridad[cu]=2
            elif(a2Table[cu] > mediaS[cu]-difMed*0.20):
                prioridad[cu]=3
            elif(a2Table[cu] > mediaS[cu]-difMed*0.30):
                prioridad[cu]=4
            else:
                prioridad[cu]=5
            raux=round(mediaS[cu]-difMed*0.05,0)
            raux=int(raux)
            if(raux <= a2Table[cu] <= 100): prioridad[cu]=1
            rVal[cu]= "[" + str(raux) + "-100]"
        elif cu in [15,17]:
            if a2Table[cu] < mediaS[cu]:
                prioridad[cu]=1
            raux2=round(raux2,1)
            if(0 <= a2Table[cu] <= raux2): prioridad[cu]=1
            rVal[cu]= "[0-" + str(raux2) + "]"
        elif cu in [30,31]:
            if a2Table[cu] < mediaS[cu]:
                prioridad[cu]=1
            raux=round(raux,0)
            raux2=round(raux2,0)
            raux=int(raux)
            raux2=int(raux2)
            if(0 <= a2Table[cu] <= raux2): prioridad[cu]=1
            rVal[cu]= "[0-" + str(raux2) + "]"
        elif cu in [32,33,34,35,36,37,38,39,40,41,42,43]:
            raux=round(raux,2)
            raux2=round(raux2,2)
            if(raux <= a2Table[cu] <= raux2 and not (prioridad[cu]==0)): prioridad[cu]=1
            rVal[cu]= "[" + str(raux) + "%-" + str(raux2) + "%]"
        else:
            raux=round(raux,0)
            raux2=round(raux2,0)
            raux=int(raux)
            raux2=int(raux2)
            if(raux <= a2Table[cu] <= raux2 and not (prioridad[cu]==0)): prioridad[cu]=1
            rVal[cu]= "[" + str(raux) + "-" + str(raux2) + "]"

    for cu in numer:
        if "Unknown"==str(aTable[0][cu]) :
            prioridad[cu]=prioridadMax[cu]



    ...
    #El m dulo continua pero aqui finaliza la valoraci n de par metros
    ...
```

Figura B.5: Cálculo de valores recomendados y nivel de error

Fuente: Elaboración propia.

APÉNDICE C
Glosario de términos

C.1 Definiciones

API: Conjunto de funciones y métodos ofrecidos por alguna biblioteca que puede ser utilizada dentro de por otro software.

SEO: Conjunto de técnicas y estrategias usadas para mejorar la visibilidad de una página web dentro de un motor de búsqueda como Google y Bing.

SERPs: Resultados más destacados mostrados por un buscador como Google y Bing.

HTML: Código utilizado para mostrar y estructurar el contenido de una página web.

URL: Dirección que recibe un recurso web específico. Sirve como método de identificación para un recurso en internet.

GUI: Conjunto de objetos gráficos que permiten al usuario a interactuar con el programa.

API: Protocolos y definiciones que permiten la comunicación entre dos aplicaciones de software, en la una de ellas ofrece sus servicios software a la otra.

Enlace: Referencias de un recurso web a otro Comúnmente conocido como link.

World Wide Web: Colección de paginas web almacenadas en servidores y que son capaces de conectarse con dispositivos locales a través de internet.

Python: Lenguaje de alto nivel de programación en el que se destaca su legibilidad y su variedad de librerías.

Webmasters: Es el administrado de una pagina web. Se suele encargar del mantenimiento y de la programación de la página.

Crawler: Programa informático que inspecciona de manera automatizada y metódiga las paginas de la World Wide Web.

Web Scraper: Software que permite la indexación y almacenamiento de información procedente de la web.

Query: o consulta, es aquella búsqueda específica realizada en un motor de búsqueda.

Keywords: Palabras usadas en una query que facilita la ubicación de una URL a través de los motores de búsqueda.

PageRank: Serie de algoritmos desarrollados por Google y son usados para optimizar los resultados de su buscador. Definen los resultados del posicionamiento de las webs y actualmente no son visibles.

Autoridad de dominio: Serie de algoritmos creados por MOZ (consultoría SEO) como sustituto al PageRank de Google, en el que trata de estimar la probabilidad de obtener un buen posicionamiento de una web determinada.

Protocolo: Método establecido para el intercambio y transferencia de datos a través de una red informática y que conforma la primera parte de una URL (ej: https).

Backlinks: Enlaces de paginas externas que referencian a una página en específico.

Sitelinks: Enlace que en algunas ocasiones aparece justo debajo de la descripción de una web en una Google Search y que ayuda al usuario a navegar rápidamente a enlaces de interés del mismo dominio.

Sitemap: Archivo dentro de un dominio que proporciona información sobre las paginas y archivos de la web, así como sus relaciones.

Google My Business: Herramienta de Google que permite a las empresas y organizaciones administrar su presencia en línea en Google.

Core Web Vitals: Parámetros básicos utilizados en la valoración del rendimiento de una web y de la experiencia del usuario al acceder a ella.

Aprendizaje automático: Una serie de métricas usadas en cada web de Google y que van centradas a la experiencia del usuario.

Stopword: Serie de palabras que no se consideran importantes y se eliminan para utilizar solo las que los son.

xslx: Extensión usada por archivos en forma de hoja de cálculo, como los creados por Microsoft Excel.

C.2 Acrónimos

TFG: Trabajo de Fin de Grado.

API: Application Programming Interface (Interfaz de Programación de Aplicaciones).

SEO: Search Engine Optimization (Optimización para Motores de Búsqueda).

SERP: Search Engine Results Page (Página de Resultados del Buscador).

KW: Keyword (Palabra clave).

URL: Uniform Resource Locator (Localizador de Recursos Uniforme).

HTML: HyperText Markup Language (Lenguaje de Marcado de Hipertexto).

GUI: Grafical User Interface (Interfaz Gráfica de Usuario).

API: Application Programming Interface (Interfaz de Programación de Aplicaciones).

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

APÉNDICE D
Objetivos de Desarrollo Sostenible

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenible	**Alto**	**Medio**	**Bajo**	**No procede**
ODS 1. **Fin de la pobreza.**				X
ODS 2. **Hambre cero.**				X
ODS 3. **Salud y bienestar.**				X
ODS 4. **Educación de calidad.**				X
ODS 5. **Igualdad de género.**				X
ODS 6. **Agua limpia y saneamiento.**				X
ODS 7. **Energía asequible y no contaminante.**				X
ODS 8. **Trabajo decente y crecimiento económico.**			X	
ODS 9. **Industria, innovación e infraestructuras.**			X	
ODS 10. **Reducción de las desigualdades.**				X
ODS 11. **Ciudades y comunidades sostenibles.**				X
ODS 12. **Producción y consumo responsables.**				X
ODS 13. **Acción por el clima.**				X
ODS 14. **Vida submarina.**				X
ODS 15. **Vida de ecosistemas terrestres.**				X
ODS 16. **Paz, justicia e instituciones sólidas.**				X
ODS 17. **Alianzas para lograr objetivos.**				X

Reflexión sobre la relación del TFG/TFM con los ODS y con el/los ODS más relacionados:

Dada la naturaleza de este trabajo, establecer una relación con los Objetivos de Desarrollo Sostenible es algo bastante complicado. El programa desarrollado no tiene como objetivo proveer ningún tipo de servicio que beneficie directamente a los más necesitados, a la igualdad social o a la sostenibilidad del planeta, sino que simplemente trata de entender el funcionamiento de lo que se podría considerar el sistema informático más utilizado en el mundo, el motor de búsqueda de Google.

A pesar de esto, si evitamos entrar muy en detalle, sí que se podría establecer una ligera conexión con dos de los objetivos, más concretamente, el objetivo 8, "Trabajo decente y crecimiento económico", y el objetivo 9, "Industria, innovación e infraestructuras".

Una de las posibles aplicaciones del programa y de las conclusiones elaboradas en nuestro trabajo, es la posibilidad de permitir a entidades, o empresas pequeñas, mejorar su presencia en internet. Esto es evidente cuando se piensa que los parámetros de posicionamiento estudiados en este trabajo no precisan de la intervención de agentes externos, por lo que cualquier empresa, independientemente de su tamaño, podría hacer uso de las conclusiones obtenidas para facilitar su digitalización. Una vez alcanzado este punto, las posibilidades respecto a la modernización tecnológica e innovación dentro de una empresa serían mucho más amplias, promoviendo, por tanto, el crecimiento económico que se intenta impulsar dentro del octavo objetivo de las ODS.

Por otro lado, a diferencia del anterior, establecer una conexión general con el noveno objetivo no es algo posible, ya que nuestro trabajo no presenta ningún elemento que permita promover la industrialización o la construcción de infraestructuras resilientes. Pero si que es posible relacionarlo con una de las metas específicas propias de dicho objetivo. Más concretamente la meta 9.c, la cual, como se describe en la Agenda 2030 sobre el Desarrollo Sostenible, consiste en "Aumentar significativamente el acceso a la tecnología de la información y las comunicaciones y esforzarse por proporcionar acceso universal y asequible a Internet en los países menos adelantados de aquí a 2020".

Aunque, como se observa en la cita anterior, la fecha del objetivo 9.c ya haya sido alcanzada, es obvio que el acceso a internet y a las diferentes tecnologías de la información no está, ni mucho menos, normalizada dentro de los países menos desarrollados. Nuestro trabajo, por motivos muy similares al octavo objetivo, podría ayudar al cumplimiento de dicha meta. En este caso, la información recolectada en el trabajo sobre el posicionamiento web, facilitaría a cualquier persona o entidad, a mejorar su visibilidad dentro de internet, más concretamente, dentro de los resultados del buscador de Google, el portal de información número uno del mundo. Esto podría ayudar a los países menos habituados a las tecnologías de la información y las comunicaciones a establecer su propia presencia dentro de internet.

Dicho todo esto, nos guisaría pensar que el trabajo tiene una fuerte relación con alguno de los objetivos, pero ese, desafortunadamente, no es el caso. Aunque ha sido posible relacionarlo con algunos objetivos, los casos de uso propuestos son claramente idílicos y, en el mejor de los casos, este trabajo solo podría llegar a ser un pequeño complemento para el avance tecnológico y digital.

ETS Enginyeria Informàtica
Camí de Vera, s/n, 46022, València
T +34 963 877 210
F +34 963 877 219
etsinf@upvnet.upv.es - www.inf.upv.es