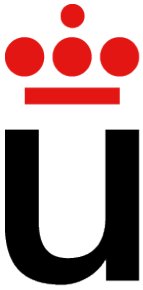

Universidad
Rey Juan Carlos

GRADO EN INGENIERÍA EN SISTEMAS AUDIOVISUALES Y MULTIMEDIA

Curso Académico 2021/2022

Trabajo Fin de Grado

AMPLIACIÓN DEL ANÁLISIS DE CÓDIGO PYTHON CON CEFRL

Autor : Javier Jiménez Ramiro

Tutor : Dr. Gregorio Robles Martínez

Trabajo Fin de Grado

Ampliación del Análisis de Código Python con CEFRL

Autor : Javier Jiménez Ramiro

Tutor : Dr. Gregorio Robles Martínez

La defensa del presente Proyecto Fin de Carrera se realizó el día de de 2022, siendo calificada por el siguiente tribunal:

Presidente:

Secretario:

Vocal:

y habiendo obtenido la siguiente calificación:

Calificación:

Fuenlabrada, a de de 2022

Un alma saludable

habita en un cuerpo y una mente saludables.

Maka Albarn

II

Agradecimientos

Con este proyecto doy final a una maravillosa etapa universitaria, un camino recorrido junto a mucha gente, gente que vino y se quedo, gente que siempre estuvo ahí y gente que se acabó yendo pero que aprendí algo de ellos.

Por ello quiero dar las gracias a esas personas que siempre estuvieron ahí y que creyeron en que yo podría conseguirlo. La primera de estas personas soy yo mismo, quiero agradecer el no rendirme nunca, el seguir luchando por muy difícil que se pusiera todo y el hecho de que si caía derrotado o cometía algún error me levantaba y lo solucionaba asumiendo las consecuencias necesarias.

Por supuesto agradecer a mis padres, porque ellos lucharon día a día para permitirme cumplir este sueño, me ayudaron y me apoyaron y a día de hoy lo siguen haciendo. Siempre recordaré los abrazos de mi madre y la frase que me dijo mi padre el primer día de esta etapa: ”si tienes algún problema estoy solo a tres horas de coche”.

También me gustaría agradecer a mis dos hermanos, las dos personas que más me han escuchado y ayudado en los malos momentos, mi hermano de sangre y mi hermana de espíritu. Mi hermano me enseño que la etapa universitaria debes tomarla con calma, no amargarte y seguir luchando, me enseñó que si quieres algo debes luchar que nadie te lo regala y me enseñó que ser el mejor es cuestión de actitud y creer que puedes. Mi hermana me enseñó la importancia de la tranquilidad, me enseñó que puedo superar mis miedos y que soy mucho más de lo que creo, me escuchó siempre y siempre estuvo ahí cuando la necesité, lloró conmigo y rió conmigo y me enseñó que la paz vive en nuestro interior, que no eres ni más ni menos que nadie y que todos merecemos un abrazo, una sonrisa y mucho respeto. Y que si transmites alegría, recibes alegría.

Por último y no menos importante agradecerle a mi mano izquierda, el regalo que me dio la universidad y uno de mis mejores amigos, el hizo que estar a tres horas de mi hogar fuera más

ameno. Darle las gracias a ellos y a mis profesores en especial a Susana que hizo que el inglés fuese divertido y a Gregorio que me ayudo a que este proyecto siguiera, me transmitió su pasión por Python y ayudó a que todo fuese más fácil. Gracias a todos.

Resumen

El objetivo de este proyecto es ampliar la herramienta ya existente de análisis de código Python, inspirada en los niveles del CEFRL, con dos nuevas opciones de análisis como son fragmentos de respuestas en la web Stack Overflow y aportaciones a proyectos mediante Pull Request en GitHub, añadiendo filtros y adecuando los fragmentos de código al método de análisis ya existente.

El uso de esta herramienta está enfocado, ya no solo en la educación y evaluación del alumnado, en ser capaces de evaluar código externo para poder añadirlo a nuestros proyectos cerciorandonos de que el código es de buena calidad o está acorde a nuestro nivel de comprensión.

Para ello se han utilizado varias tecnologías. El lenguaje Python para la extracción del código como para el desarrollo de los filtros, JSON para la muestra de resultados y tanto GitHub como Stack Overflow para la obtención de ejemplos y pruebas y para la base de desarrollo de este proyecto.

Summary

The objective of this project is to extend the existing Python code analysis tool, inspired by the CEFRL levels, with two new analysis options such as response snippets on the Stack Overflow website and contributions to projects via Pull Request on GitHub, adding filters and adapting the code snippets to the existing analysis method.

The use of this tool is focused, not only in the education and evaluation of students, in being able to evaluate external code to be able to add it to our projects making sure that the code is of good quality or is according to our level of understanding.

Several technologies have been used for this purpose. The Python language for code extraction and filter development, JSON for sample results and both GitHub and Stack Overflow for obtaining examples and tests and for the development base of this project.

Índice general

Índice de figuras

Capítulo 1

Introducción

En el mundo de auge tecnológico en el que vivimos, en el que cada año muchas de las tecnologías se quedan obsoletas por el nacimiento o desarrollo de otras, la forma más rápida y fácil de crecer y ampliar el conocimiento es compartiéndolo con otras personas, es decir, intercambiando ideas y trabajando en equipo.

Como desarrollador software tienes dos maneras principales de hacer esto, o bien preguntando a una comunidad de miles de personas con más experiencia y conocimientos que puedan aportarte una solución práctica a tu problema, o bien permitiendo el hecho de que otros desarrolladores puedan darte consejos y mejoras a tu proyecto dejando todo tu código fuente a disposición de todo el mundo que quiera aportar altruistamente su granito de arena.

Esto se puede realizar por medio de páginas web con comunidades de dimensiones mundiales como github[1] o stackoverflow[2].

Pero, al tener acceso todo el mundo, ¿Cómo podemos saber si una respuesta o una aportación es correcta? Y profundizando más aún, ¿Cómo podemos saber si esa misma respuesta o aportación tiene el nivel que nosotros deseamos para nuestro proyecto?.

Pues así nació la idea de este proyecto, el poder analizar las respuestas dadas en stackoverflow y las aportaciones ofrecidas en github por medio del uso del software de un proyecto anterior centrándonos en el lenguaje de programación de código abierto Python.

Este proyecto mencionado en el párrafo anterior ya era útil para conocer cual es tu nivel en python, ver tu propia mejoría o incluso en el ámbito de la educación poder analizar y evaluar los

[1] https://github.com/

[2] https://stackoverflow.com/

proyectos del alumnado haciendo más fácil y rápida esta tarea, pero gracias a esta ampliación ahora también podrás ver que aportaciones o que respuestas son las más adecuadas para cada uno de tus proyectos mediante el marco e análisis, ya utilizado en muchos campos, CEFRL.

Este marco nos permite el dividir en tres categorías con dos subcategorías cada una, estas categorías son: usuario básico (A), usuario independiente (B) y usuario competente (C).

Así por este método, aunque el lema de python sea ”solo hay una manera de hacer las cosas, podremos reescribir este lema diciendo ”hay maneras más correctas que otras de hacer la cosasz escoger por nosotros mismos las que más se adecuen a nuestro proyecto y entendimiento.

1.1. Estructura de la memoria

En cuanto a la estructura, se divide está memoria en 7 grandes capítulos abordando cada una de las cuestiones más importante de este proyecto.

Los capítulos son:

- **Capítulo 1:** En este capítulo se muestra la introducción del proyecto y la estructura que llevará la memoria al completo.
- **Capítulo 2:** En este capítulo se abordan los objetivos tanto generales como específicos que se han seguido para el desarrollo del proyecto así como una planificación temporal el mismo.
- **Capítulo 3:** En este capítulo se desarrolla una breve explicación de todas las tecnologías usadas en la realización del proyecto.
- **Capítulo 4:** En este capítulo se muestra todo el proceso que se ha llevado a cabo para completar la ampliación del análisis de código Python de manera detallada.
- **Capítulo 5:**
- **Capítulo 6:**
- **Capítulo 7:** En este capítulo se abordan todos los problemas y éxitos encontrados a la hora de realizar los objetivos mencionados en el capítulo 2, todo lo aplicado y aprendido

durante el desarrollo del grado de manera personal y posibles trabajos futuros que ayuden a la mejora constante de esta herramienta.

Capítulo 2

Objetivos

2.1. Objetivo general

Mi trabajo de fin de grado consiste en ampliar las opciones de análisis de la herramienta software de evaluación de código python, version 3, inspirada en el marco CEFRL con el análisis de las respuestas dadas en stackoverflow y las aportaciones dadas en github a través de pull requests.

Con esto seremos capaces de introducir esta ayuda externa en nuestro código sabiendo exactamente cómo de buenas son estas aportaciones.

2.2. Objetivos específicos

Para el objetivo general descrito anteriormente se han de abordar los siguientes objetivos específicos:

- Estudiar y entender el funcionamiento de análisis ya existente dentro del programa de pycefrl y comprobar la funcionalidad ya instalada siguiendo los pasos dentro del código.

- Modificar y añadir las funcionalidades nuevas descritas anteriormente acondicionandolas e integrandolas para su correcto paso por el análisis del programa.

- Extraer y guardar los fragmentos de código útiles para su futuro análisis.

- estudiar y aplicar los diferentes filtros para cada una de las nuevas funcionalidades y poder transformar el fragmento de código en un objeto útil para su análisis.

- probar las nuevas funcionalidades con cien URLs de interés para cada opción.

- Mejorar dentro de lo posible aquellos fallos que muestren los distintos filtros después de la primera batería de pruebas.

- Probar de nuevo y anotar todos los resultados para cada opción para poder cerciorarnos de la eficacia del análisis y de las condiciones para que el mismo sea correcto.

2.3. Planificación temporal

Este proyecto comenzó en febrero de 2022, después de un intento fallido de TFG que comenzó en septiembre del año anterior.

Después de trabajar en el primer intento ocurrió un error en enero de 2022 que dejó mi ordenador inservible y supuso la perdida de todos los datos y de todo el trabajo de esos meses, lo que causo que, tras mucho valorar, decidí que necesitaba un cambio a otro proyecto que llamase mucho más mi interés.

Y así, teniendo claro que yo quería que tratase sobre programación y aún más importante sobre Python, comencé con este proyecto con muchas más fuerzas e ilusión.

En los primeros dos meses de trabajo, febrero y marzo, se consiguió la correcta extracción de los fragmentos de código de interés pero sin tener éxito en el resultado final de su análisis, creyendo que era por culpa del análisis se decidió abordar la parte analítica y comprobar que podía estar fallando, esto costó mucho tiempo ya que a la vez que trabajaba en este proyecto la mitad de la tarde de lunes a viernes, al rededor de dos horas y media al día, también trabajaba en las prácticas por las mañanas y estudiaba para obtener el titulo B1 de inglés asistiendo al final de la tarde a una academia.

Tras estos dos meses a mediados de abril, después de una reunión de dudas con mi tutor, me di cuenta que realmente el fallo no se encontraba en el análisis si no en los filtros utilizados para el acondicionamiento del fragmento de código a analizar, por lo tanto, pasé el resto abril dedicando la tarde completa y las primeras semanas de mayo dedicando toda la mañana ya que

había finalizado mis prácticas de empresa, al rededor de 5 horas por día de lunes a jueves, a mejorar y añadir los filtros que consideraba necesarios y útiles.

El resto del mes de mayo y parte de la primera semana de junio durante toda la mañana de lunes a jueves, habiendo terminado todos los filtros, me dediqué a probar mi herramienta con cien URLs por opción, doscientas en total, y anotando los resultados y porcentajes.

Por último la segunda semana de Junio se comenzó con la memoria con vistas a entregarla en el mismo mes y realizar su defensa a principios del mes de julio.

Capítulo 3

Estado del arte

En este capitulo se explican las tecnologías más importantes utilizadas en el desarrollo de este proyecto.

3.1. Python

Python[4] es un lenguaje de programación de alto nivel, interpretado y de propósito general. Su filosofía de diseño enfatiza la legibilidad del código con el uso de una sangría significativa. A menudo se describe como un lenguaje çon pilas incluidas"debido a su completa biblioteca estándar.

En cuanto a su historia, Python fue concebido a finales de la década de 1980 por Guido van Rossum en el Centrum Wiskunde Informatica de los Países Bajos como sucesor del lenguaje de programación ABC, inspirado en SETL, capaz de manejar excepciones y de interactuar con el sistema operativo Amoeba. Su implementación comenzó en diciembre de 1989. Python 3.0, publicado el 3 de diciembre de 2008, con muchas de sus principales características retrocedidas a Python 2.6.x y 2.7.x. Las versiones de Python 3 incluyen la utilidad 2to3, que automatiza la traducción del código de Python 2 a Python 3. Posteriormente, también se dejó de dar soporte a la versión 3.6. En 2021, Python 3.9.2 y 3.8.8 fueron acelerados ya que todas las versiones de Python tenían problemas de seguridad que conducían a una posible ejecución remota de código y envenenamiento de la caché web.

En su diseño, Python es un lenguaje de programación multiparadigma. La programación orientada a objetos y la programación estructurada están totalmente soportadas, y muchas de

sus características soportan la programación funcional y la programación orientada a aspectos. Muchos otros paradigmas están soportados a través de extensiones, incluyendo el diseño por contrato y la programación lógica. Su diseño ofrece cierto apoyo a la programación funcional en la tradición Lisp.

Las normas de su filosofía mas destacadas son:

- Bonito mejor que feo.
- Explicito es mejor que implícito.
- Lo simple mejor que lo complejo.
- Lo complejo es mejor que lo complicado.
- La legibilidad cuenta.

En lugar de construir toda su funcionalidad en su núcleo, Python fue diseñado para ser altamente extensible a través de módulos. Python se esfuerza por conseguir una sintaxis y una gramática más sencillas y menos recargadas, al tiempo que ofrece a los desarrolladores la posibilidad de elegir su metodología de codificación. Los desarrolladores de Python se esfuerzan por evitar la optimización prematura y rechazan los parches en partes no críticas de la implementación de referencia de CPython que ofrecerían incrementos marginales de velocidad a costa de la claridad. Un neologismo común en la comunidad de Python es pitónico, que tiene una amplia gama de significados relacionados con el estilo del programa.

3.2. JSON

JSON[3] es un formato de texto sencillo para el intercambio de datos. Una de las supuestas ventajas de JSON sobre XML como formato de intercambio de datos es que resulta mucho más sencillo escribir un analizador sintáctico para él. Por esa razón, JSON se emplea habitualmente en entornos donde el tamaño del flujo de datos entre cliente y servidor es de vital importancia cuando la fuente de datos es explícitamente de fiar y donde no es importante el hecho de no disponer de procesamiento XSLT para manipular los datos en el cliente.

Un ejemplo de datos JSON podría ser:

```
{
  "departamento":8,
  "nombredepto":"Ventas",
  "director": "Juan Rodríguez",
  "empleados":[
    {
      "nombre":"Pedro",
      "apellido":"Fernández"
    },{
      "nombre":"Jacinto",
      "apellido":"Benavente"
    }
  ]
}
```

3.3. Github

GitHub[1] es una forja(plataforma de desarrollo colaborativo) para alojar proyectos utilizando el sistema de control de versiones Git. El software que opera GitHub fue escrito en Ruby on Rails. El código de los proyectos alojados en GitHub se almacena generalmente de forma pública.

3.3.1. Repositorios

Los repositorios son, como su propio nombre indica, lugares virtuales alojados en la nube en donde los usuarios pueden almacenar cualquier tipo de archivo. Para ser identificados, los repositorios muestran información relevante, como la descripción del proyecto, los lenguajes de programación usados o las etiquetas de búsqueda. Un aspecto muy importante de la información de los repositorios es el archivo «README», el cual los desarrolladores pueden crear para describir su proyecto a fondo.

3.3.2. Issues

En GitHub, los Issues son abiertos por usuarios que tienen un fallo con el código del repositorio y quieren recibir ayuda para solucionarlo, pos usuarios que quieren plantear una compleja

mejora del programa, entre otros usos. Se pueden enlazar pull requests a los issues, así como asignar usuarios y etiquestas.

3.3.3. Pull Requests

Las pull requests son solicitudes de admisión de código. Los usuarios pueden editar cualquier archivo de código de un repositorio, ya sea a través del editor integrado de GitHub o desde su ambiente local. Adicionalmente, pueden revisar el código y enviar comentarios a modo de retroalimentación. Este proceso se conoce como revisión de código".

3.3.4. Debates

Son parecidos a los Issues ya que pueden crearse sobre temas específicos y almacenar comentarios de otros usuarios a modo de foro. Para una mejor organización, los debates pueden clasificarse en diferentes categorías que el dueño del repositorio puede crear. A diferencia de los issues, hay un sistema de votos para los comentarios y un usuario con permisos en el repositorio puede marcar una respuesta como 'correcta'. Pueden convertirse en issues.

3.3.5. Acciones

Es un servicio que permite al usuario crear flujos de trabajo con los cuales automatizar ciertas acciones, logrando así una integración continua y una entrega continua. Las acciones pueden ser personalizadas gracias al uso de archivos en un formato concreto.

3.3.6. Wiki

Una función muy útil para algunos repositorios es la de la creación de páginas de wiki. Gracias a estas, los creadores de los proyectos pueden escribir artículos que expliquen más cosas acerca del proyecto o de su funcionamiento.

3.4. Stack Overflow

Stack Overflow[5] es un sitio de preguntas y respuestas para programadores profesionales y aficionados. Es el sitio emblemático de la red Stack Exchange, creado en 2008 por Jeff Atwood

y Joel Spolsky. Contiene preguntas y respuestas sobre una amplia gama de temas de programación. Se creó para ser una alternativa más abierta a sitios previos de preguntas y respuestas como Experts-Exchange.

El funcionamiento de su web es bastante sencillo:

1. El usuario se registra en la web
2. El usuario realiza su pregunta y la hace pública
3. El usuario comienza a recibir respuestas de la comunidad

Las respuestas son publicadas por los miembros de una comunidad determinada o por otros usuarios con las mismas experiencias que encontraron solución al problema planteado.

Se ha convertido en una web muy fiable por diversas razones de las cuales se pueden destacar:

- Confianza de los usuarios
- Habilidades de comunicación
- Calidad y relevancia en preguntas y respuestas
- Manejo de los temas
- Nivel de experiencia y participación

3.5. Proyecto de programación del CEFRL

Este proyecto proporciona una tabla la cual evalúa las habilidades de programación de un usuario en tres clases, escritura, entendimiento e interacción, y a su vez en varias subclases como pueden ser reutilización del código o autoaprendizaje dentro del lenguaje.

Dependiendo del nivel adquirido en cada una de estas clases podemos catalogar a los programadores en tres grandes niveles divididos a su vez en dos niveles cada uno:

- A (A1, A2): el primer nivel sería para usuarios básicos capaces de realizar las tareas con guías o supervisión de programadores con niveles más altos.

- B (B1, B2): el segundo nivel sería para aquellos programadores prácticamente independientes que solo necesitarían de pequeñas orientaciones o indicaciones par realizar la tarea.

- C (C1, C2): el último nivel está reservado para todos aquellos programadores con nivel suficiente para no necesitar ninguna indicación y ser totalmente independientes.

Esta clasificación por niveles nos da mucha libertad a la hora de usarla ya que con ella podemos clasificar la dificultad de las diferentes tareas, la calidad de los proyectos e incluso evaluar las habilidades de un grupo de alumnos en cada clase y así también ser más eficientes a la hora de asignar trabajos y manejar equipos de la forma más correcta.

Capítulo 4

Diseño e implementación

4.1. Arquitectura general

Ya se ha visto en la sección 2.2 los pasos necesarios para conseguir el objetivo principal de este proyecto. Para mostrarlo más gráficamente centrémonos en la figura 4.1 y en la figura 4.2, la primera nos muestra como funcionaba la herramienta anteriormente y en la segunda se puede observar su funcionamiento actual.

La herramienta anteriormente, explicada con detalle en la tesis de mi compañera Ana Poveda García Herrero, contaba con un programa principal Python al cuál se le añadían parámetros por la shell indicando si se quería analizar un directorio, un repositorio o un usuario en GitHub y en el caso de que fuera un parámetro de GitHub se utilizaba su API para extraer los ficheros necesarios a analizar. En la parte izquierda de la figura 4.1 se destaca la clase IterTree la cual, junto con el módulo ast de árboles de sintaxis abstracta y el fichero de configuración, extraía los niveles y clases necesarias para su análisis.

Por último, el programa principal nos devuelve los resultados por la shell además de en un formato CSV y JSON el cual se utiliza como una base de datos para poder formar la página de manera visual con HTML, JavaScript y CSS.

En el caso de la nueva herramienta mejorada, el análisis y la muestra de resultados permanece de la misma manera, el gran cambio donde se ha centrado toda la atención es en añadir dos opciones más a la lista de parámetros de objetos que se pueden analizar, en este caso, respuestas planteadas a las preguntas de Stack Overflow, página mencionada anteriormente en el capítulo 3, y código añadido a los pull request de GitHub, mencionados a su vez en el capítulo 3. Como

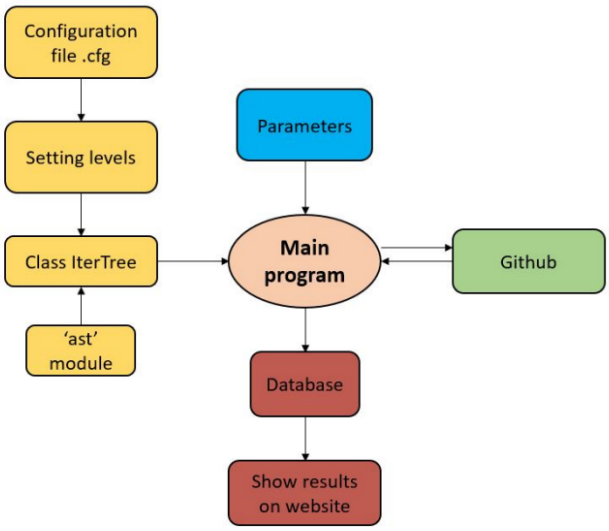

Figura 4.1: Esquema de funcionamiento principal.

observamos en la parte derecha de la figura 4.2 vemos que si se recibe por parámetros las opciones de stack overflow o de un pull request, antes de proceder con su análisis, entra en juego un nuevo fichero Python que mediante una de las librerías de Python extrae todos los fragmentos de código de interés y posteriormente los somete a una serie de filtros y limpieza para adecuar este tipo de fragmentos al análisis que ya estaba creado con anterioridad escribiéndolos uno a uno dentro de un nuevo directorio y prescindir de aquellos que no pasen los filtros o que no sean de relevancia a la hora de analizarlos correctamente.

Finalmente, y utilizando la manera de análisis de directorios que ya había, se le devuelve al programa principal el directorio creado con todos los fragmentos de código útiles para su analisis y muestra de resultados por CSV y JSON.

4.2. Obtención de ficheros en las diferentes opciones

En este punto y una vez ejecutado el fichero de configuración (explicado en el capitulo 5) podemos enviar al programa principal distintas parejas de opción y valor según lo que queramos analizar.

las distintas opciones son:

- **Directory:** En este caso el valor que debe ir con esta opción tiene que ser una ruta a una

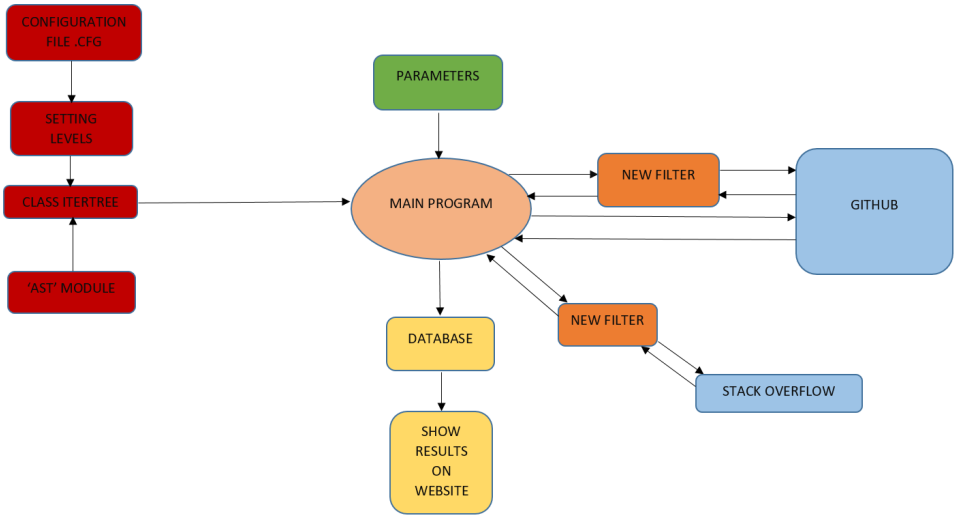

Figura 4.2: Esquema de funcionamiento mejorado.

carpeta en específico. Ya que, como la propia definición de directorio especifica, debe ser una carpeta contenedora de archivos y otros subdirectorios. Un ejemplo de esto podría ser:

```
Users\jjime\Escritorio\tfg
```

- **repo-url:** En este caso el valor que lo acompaña debe ser una URL asociada a un repositorio alojado en GitHub. Para conseguir esto debemos ir a cualquier repositorio que queramos analizar en GitHub, pulsar en el botón 'clone' y copiar la URL de https. Un ejemplo de esto podría ser:

```
https://github.com/jjramiro/pycefr.git
```

- **user:** En este caso el valor que acompaña a esta opción debe ser un usuario de GitHub cualquiera. Con esto analizara todos los repositorios de ese usuario, un ejemplo podría ser:

```
jjramiro
```

- **stack:** Este es uno de los nuevos casos añadidos. El valor que debe de acompañar a esta opción es una URL de la web de Stack Overflow mencionada en capítulos anteriores. Para conseguir un valor valido debemos irnos a web de Stack Overflow y pulsar en el apartado 'questions', una vez ahí aplicaremos el filtro de Python 3.X y entrar en la pregunta que más nos interese y copiar la URL completa. Un ejemplo de esto podría ser:

  ```
  https://stackoverflow.com/questions/606191/
  convert-bytes-to-a-string
  ```

- **pull:** Este el segundo de los nuevos casos añadidos. El valor que acompaña a este caso es la URL en github de un pull-request mencionado en el capítulo 3. Para conseguir un valor correcto debemos irnos a GitHub y buscar un proyecto escrito en su mayoría en Python y entrar en él, una vez ahí pulsamos en el apartado de 'pull requests', pulsamos en cualquiera de los que nos aparecen y copiamos la URL completa. Un ejemplo de esto podría ser:

  ```
  https://github.com/TheAlgorithms/Python/pull/6209
  ```

En el primer caso comienza el análisis del directorio sin problemas indicando el proceso por la shell, en el segundo y tercer caso son algo más complejos ya que se hace uso de la API de GitHub para hacer peticiones GET y así clonar el repositorio únicamente si supera la mitad de código en Python. Estas tres opciones ya estaban anteriormente explicadas y hechas en la tesis de mi compañera Ana así que en los siguientes puntos nos vamos a centrar en profundidad en la obtención de los ficheros a analizar en las últimas dos opciones.

4.3. Lectura y extracción de los fragmentos de interés

En este punto nos vamos a centrar en como se extraen los fragmentos de código útiles para su análisis de las últimas dos opciones comentadas anteriormente.

Ejecutando o bien la opción de 'stack' o bien la opción de 'pull' con una URL correcta nos llevará al nuevo fichero Python de análisis añadido. las formas de extracción dependiendo de la opción serán las siguientes.

4.3.1. Lectura y extracción de Stack Overflow

En el caso de Stack Overflow comenzamos haciendo una pequeña comprobación de que la URL es correcta, comprobando si la URL contiene el nombre de la web, el apartado de preguntas, denominado 'questions', y el protocolo https.

Si la URL no tiene ningún fallo podemos pasar a la lectura de la web. Para ello utilizamos la librería de Python 'urllib'[6] que nos permite obtener el HTML completo de la página en cuestión, además, como esta organizado por etiquetas podemos aislar cada uno de los fragmentos de código, en este caso de las respuestas, que sean útiles para analizar.

Una vez realizado este aislamiento de todos los fragmentos de código mediante las etiquetas de HTML nos queda una lista con todos los fragementos o 'snippets' de las respuestas. Con esto podemos proceder a su limpieza y análisis.

4.3.2. Lectura y extracción de Pull requests

Igual que en el caso anterior lo primero que debemos hacer es una pequeña comprobación de que la URL es correcta, comprobando que la URL contenga el nombre de la web de GitHub y que el protocolo sea https.

Teniendo la URL correcta podemos pasar a la lectuar de la web igual que hacimaos en la parte anterior pero creando una nueva web de mayor interés mediante la web de github proporcianada, por lo tanto, si la web proporcionada es por ejemplo:

```
https://github.com/TheAlgorithms/Python/pull/6209
```

La nueva web creada para la extracción de ficheros sería tal que:

```
https://patch-diff.githubusercontent.com/raw/
TheAlgorithms/Python/pull/6209.diff
```

Esta nueva web nos da acceso a los archivos diff[1] dentro del pull request seleccionado, estos archivos son los fragmentos de código que han sido añadidos, eliminados o cambiados comparándolos con la rama principal del proyecto, así nos es mucho más sencillo aislar los fragmentos de interés para el análisis.

[1] https://docs.kde.org/stable5/es/kompare/kompare/working-with-diff-files.html

Teniendo esta web la manera de proceder a su lectura y extracción es muy sencilla, extraemos el HTML de la web con la misma librería mencionada anteriormente y después solo nos queda eliminar las cabeceras de cada uno de los fragmentos.

Con esto conseguimos una lista de los fragmentos que nos interesan preparados para su limpieza y análisis.

4.4. limpieza y análisis de los fragmentos obtenidos

En este punto de la extracción tenemos una lista completa con todos los fragmentos de código que pueden ser interesantes para el análisis, pero aun no están perfilados por completo ya que tenemos estos fragmentos en 'crudo'.

Por lo tanto para que el análisis sea lo más correcto posible hay que cambiar algunas partes dentro de estos fragmentos como símbolos incompatibles, por ejemplo el inicio de la shell de Python[2], partes aún en html o caracteres especiales[3].

Para solucionar este problema se han añadido varios filtros por los que tiene que pasar el fragmento para permitirle ser analizado correctamente.

4.4.1. limpieza de los fragmentos Stack Overflow

Como se comentaba en el capítulo 4.3.1 ya tenemos los fragmentos necesarios y podemos comenzar a su limpieza mediante los filtros, para esta tarea y sabiendo que en una web como stack overflow cada usuario puede escribir los fragmentos de código de la manera que quiera e incluso poner fragmentos incompletos o en otro tipo de lenguajes que no sean Python3 como en texto plano, se han ideado los filtros pensando en las complicaciones más comunes y queriendo abarcar los problemas con mayor probabilidad.

los distintos filtros que se han ideado son:

1. Para el primer filtro se hace uso de una de las librerías de Python llamada html[2]. Con esta librería transformamos todos los caracteres especiales en formato html a su formato en ascii[4] que es el que nos interesa.

[2]https://pythondiario.com/2013/04/el-interprete-de-python-o-shell.html
[3]https://ascii.cl/es/codigos-html.htm
[4]https://elcodigoascii.com.ar/

2. El siguiente filtro que entra en acción es para intentar eliminar todos aquellos fragmentos que se basen en un error mostrado por la línea de comandos y los que se basen en comandos en sí, si aparecen una de estas dos cosas ignoramos el fragmento ya que no tendrá nada de utilidad en el análisis. además en este mismo momento también se ignoran todos aquellos fragmentos que sean solo una palabra ya que no tendría sentido el analizarla.

Todos aquellos fragmentos que hayan llegado hasta aquí merecen ser guardados ya que podrían tener utilidad, pero aun quedan algunos filtros de limpieza que aplicar. Antes de seguir con la limpieza se crean archivos .py para guardar, con un nombre distintivo, todas las respuestas que podrían ser útiles. A cada fragmento se le nombra siguiendo el siguiente patrón:

```
answer_ + número de respuesta + _file_ + número del fragmento + .py
```

Con los fragmentos guardados podemos proseguir con los siguientes filtros:

1. Para el siguiente filtro pasamos el fragmento por una función que se encarga de dividir este fragmento por líneas para eliminar todos los símbolos y caracteres que tengan que ver con la estructura de la shell de Python para así obtener únicamente el código útil.

2. El siguiente filtro se encarga de los saltos de línea, estos se indican con las barras invertidas y la letra 'n', pero al extraer el código la barra invertida se duplica por lo tanto debemos eliminar esa barra extra para no causar ningún error en el análisis.

3. Para terminar aplicamos el último filtro que consta de una función que se encarga de eliminar todos aquellos fragmentos que solo posean una línea de código ya que aunque podrían llegar a ser correctos al pasar por el análisis no influye en el nivel CEFRL de la respuesta por lo tanto solo aplican un mayor tiempo de ejecución y no son eficientes.

Una vez hecho esto dentro del directorio solo quedan las respuestas útiles y se puede proceder con su análisis.

4.4.2. limpieza de los fragmentos Pull-Requests

Igual que en el caso de Stack Overflow y como se comentaba en el capítulo 4.3.2, ya tenemos una lista con todos los fragmentos listos para su limpieza y análisis, pero al contrario que en el

punto anterior en esta parte solo nos tenemos que preocupar de algunos caracteres de control (comienzan con la barra invertida) o algunos mensajes de aviso que se 'cuelan' en el fragmento seleccionado.

Para abordar los problemas más comunes ideamos los siguientes filtros:

1. El primer filtro nos permite ignorar todos los fragmentos de menos de 50 caracteres que, aunque puedan tener un análisis correcto, son demasiado pequeños y no interesa su análisis ya que no variará mucho el resultado y solo molestarían a la ejecución del análisis.

Los fragmentos que pasen de este primer filtro merecen ser guardados ya que podrían tener utilidad, pero igual que antes aun quedan algunos filtros de limpieza que aplicar. Antes de seguir con la limpieza se crean archivos .py al igual que en el punto anterior para guardar, con un nombre distintivo, todos los fragmentos que nos sean de utilidad. A cada fragmento se le nombra siguiendo el siguiente patrón:

```
pull_ + número del fragmento + .py
```

Una vez guardados podemos proseguir con los filtros de limpieza que nos quedan:

1. Para el siguiente filtro debemos eliminar algunos caracteres que aparecen en todos los pull requests cuando hay cambios o cuando se añaden y eliminan líneas de código. Estos caracteres suelen ser el signo + y el signo -, así que para eliminarlos del comienzo de todas las líneas pasamos el fragmento por una función que elimina el primer carácter de todas las líneas dejando el código mucho más limpio.

2. En el siguiente filtro de limpieza nos encargamos de eliminar tanto los simbolos de inicio de la shell de Python, al igual que en el apartado anterior, como algunos caracteres invisibles formados por la barra invertida y una letra. Con los caracteres invisibles llamados caracteres de controlo solo nos encargamos de eliminar una de las barras invertidas por estar duplicada. Estos caracteres son:

    - El primero de ellos es la barra invertida seguida de una 'n' que representa el salto de línea.
    - el segundo es la misma barra invertida seguida de una 't' que equivale a el tabulador del teclado.

- el tercero equivale a esta misma barra invertida seguida de una 'r' que equivale al retorno de carro (tecla enter).

3. En el último de estos filtros solamente eliminamos un mensaje de aviso que aparece al final de los pull-requests precedido de una doble barra invertida que se produce cuando el fragmento no tiene una línea en blanco al final del mismo.

Una vez habiendo terminado con los filtros ya solo quedan en el directorio los fragmentos listos para su análisis.

4.4.3. Análisis y resultados

Por último y acabando con este proceso, ya tenemos todos los fragmentos reunidos en un directorio y separados en ficheros .py listos para su análisis. Por ello, le devolvemos la ruta donde se encuentran estos fragmentos al programa principal, y aprovechando la opción de análisis de directorios, procedemos a el análisis de todos los fragmentos.

Aun habiendo pasado por tantos filtros, y sabiendo que Stack Overflow y la parte de Pull-Requests de GitHub son tan extensas, aun pueden ocurrir errores durante el análisis por pequeños fallo que no se han tomado en cuenta por la dificultad que supone solucionarlos y porque a grandes rasgos no son significativos.

Después del análisis completo se generan tanto un pequeño resumen mostrado por la línea de comandos, como una 'base de datos' en JSON, y otra en CSV que nos serán útiles si queremos mostrar los resultados finales de forma mas gráfica y específica mediante la creación de una página web ya disponible en la versión anterior de esta herramienta y creada por mi compañera Ana.

Capítulo 5

Manual de Usuario

En este capitulo abordaremos una correcta utilización y configuración, desde el inicio hasta los resultados, de la herramienta desarrollada mediante un manual de usuario centrándose en las dos nuevas opciones añadidas.

Lo primero que debemos hacer es obtener la herramienta de forma local desde GitHub. Para ello primero debemos hacer un 'fork'[1], basicamente copiar el repositorio de la rama principal a nuestro perfil de usuario. Después, se debe clonar este repositorio de forma local para poder utilizarlo y editarlo[2].

Una vez que lo tenemos en local empezaremos con la creación de un fichero de configuración. Este fichero sirve para determinar y fijar los distintos tipos de niveles CEFRL en Python. en este caso y aunque se pueda personalizar completamente vamos a utilizar el que esta hecho por defecto.

Para ello ejecutaremos en nuestra shell dentro del directorio local de nuestra herramienta el siguiente comando:

```
python3 dict.py
```

Ejecutando este comando ya tenemos el fichero de configuración listo y ahora podemos pasar a ejecutar el comando para el análisis de cada una de las opciones disponibles.

- Para el análisis de un repositorio Github:

[1] https://desarrolloweb.com/articulos/fork-git

[2] https://docs.github.com/es/repositories/creating-and-managing-repositories/cloning-a-repository

```
python3 pycerfl.py repo <name_urlclone>
```

- Para el análisis de un usuario Github:

```
python3 pycerfl.py user <name_user>
```

- Para el análisis de un directorio local:

```
python3 pycerfl.py directory <name_path>
```

Pero nos centraremos como he mencionado anteriormente en el análisis de Stack Overflow y en el análisis de los Pull-Requests.

5.1. Manual de análisis Stack Overflow

Para comenzar con este análisis entraremos en la página web de Stack Overflow:

```
https://stackoverflow.com/
```

Una vez dentro pulsaremos en las tres líneas de la parte superior izquierda que vemos en la figura 5.1 y pulsaremos en el apartado 'questions'.

En el apartado de las preguntas veremos algo como lo que aparece en la figura 5.2. En esta parte pulsaremos en el botón 'filter' y buscaremos, dentro del apartado de 'the following tags', la etiqueta clave 'python3'. Una vez hecho esto pulsamos en 'apply filter'.

Con esto nos aseguraremos de que todas las preguntas que aparecen son sobre Python y podremos analizarlas. Ahora podemos pulsar en cualquiera de las preguntas y nos aparecerá algo como la figura 5.3.

Antes de continuar con el comando final crearemos, dentro del directorio de nuestra herramienta, el directorio donde se guardaran los fragmentos de código analizados de Stack Overflow con el comando:

```
mkdir /test_directory
```

Con esto ya solo nos queda copiar la URL y ejecutar el comando dentro del directorio local de nuestra herramienta de esta manera:

```
python3 pycerfl.py stack <url_stack>
```

Al ejecutarlo nos da una respuesta como la de la figura 5.4.

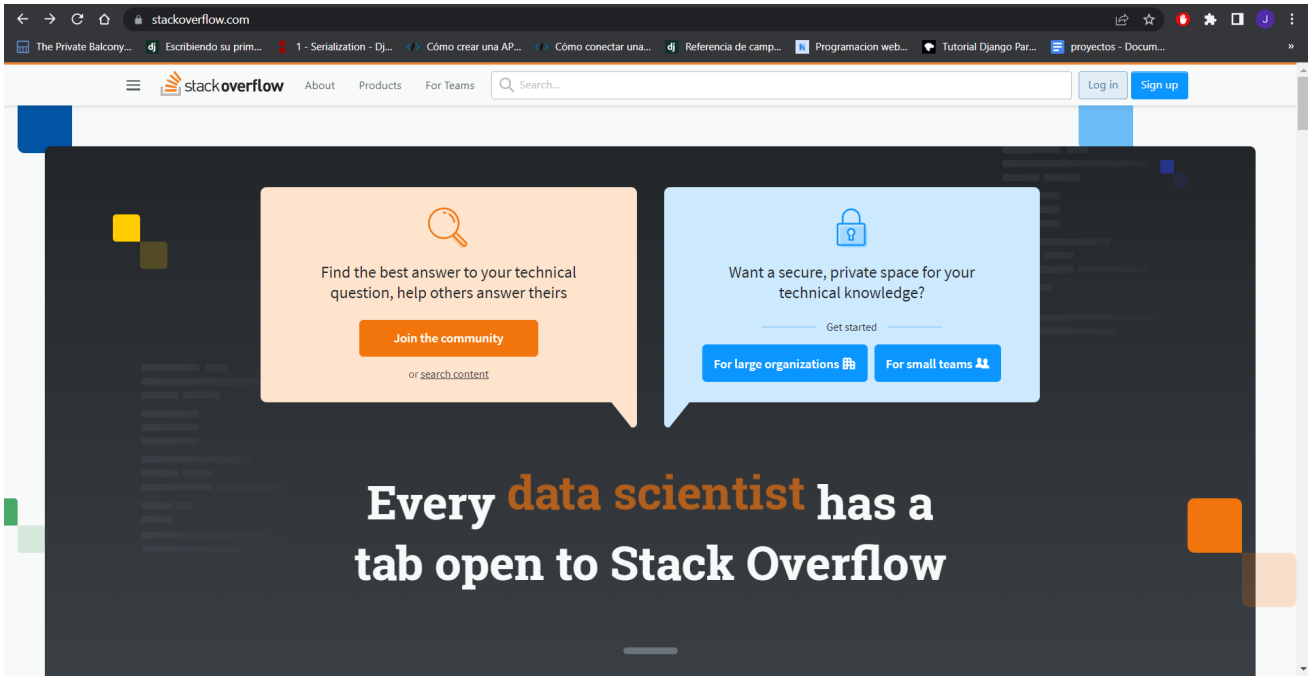

Figura 5.1: Pagina principal de Stack Overflow

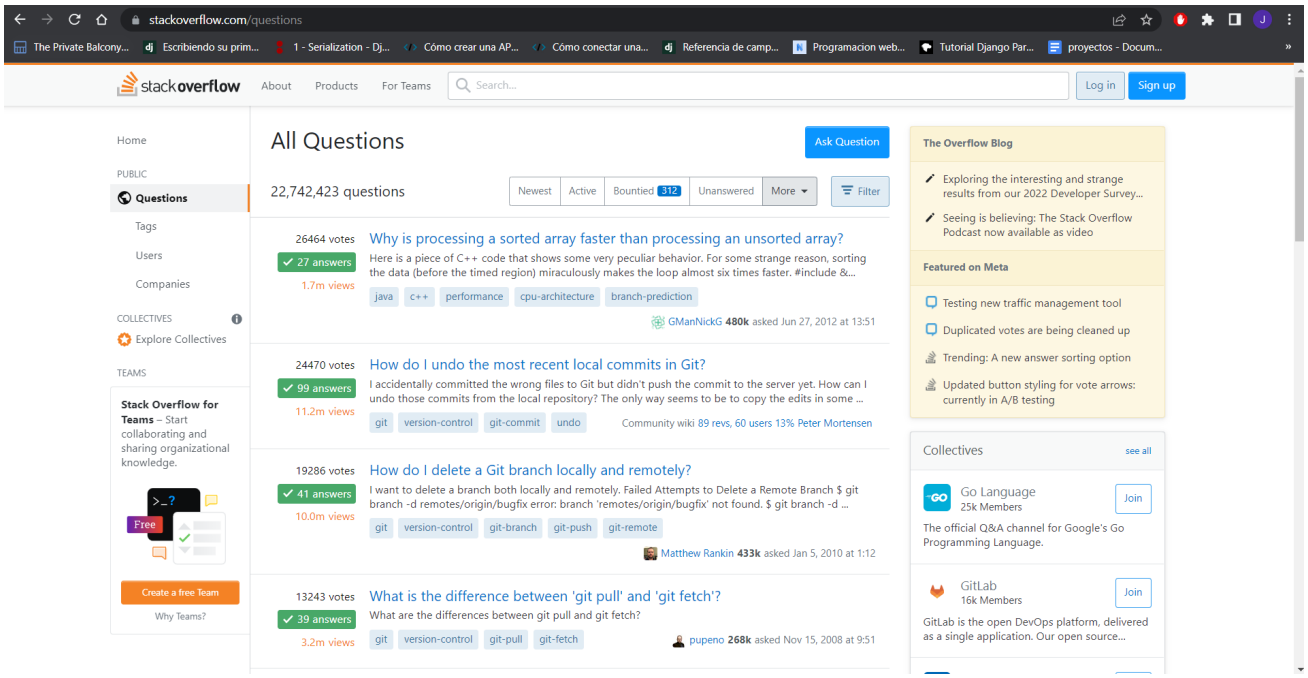

Figura 5.2: Pagina de preguntas de Stack Overflow

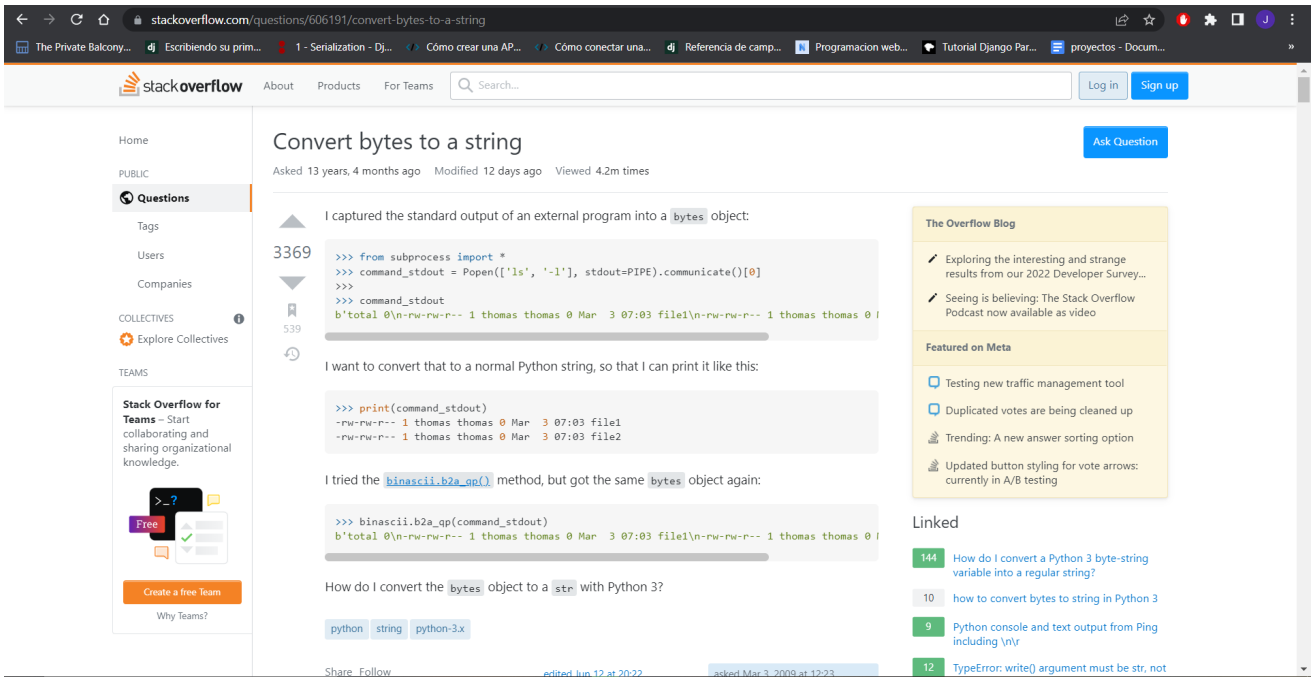

Figura 5.3: Pregunta de ejemplo de Stack Overflow

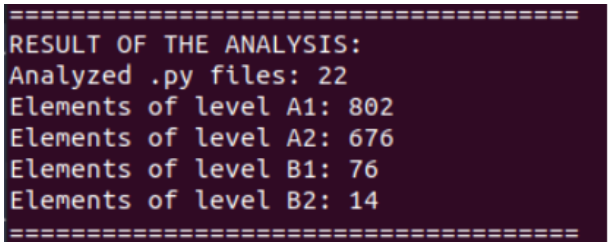

Figura 5.4: Resultado de análisis de Stack Overflow

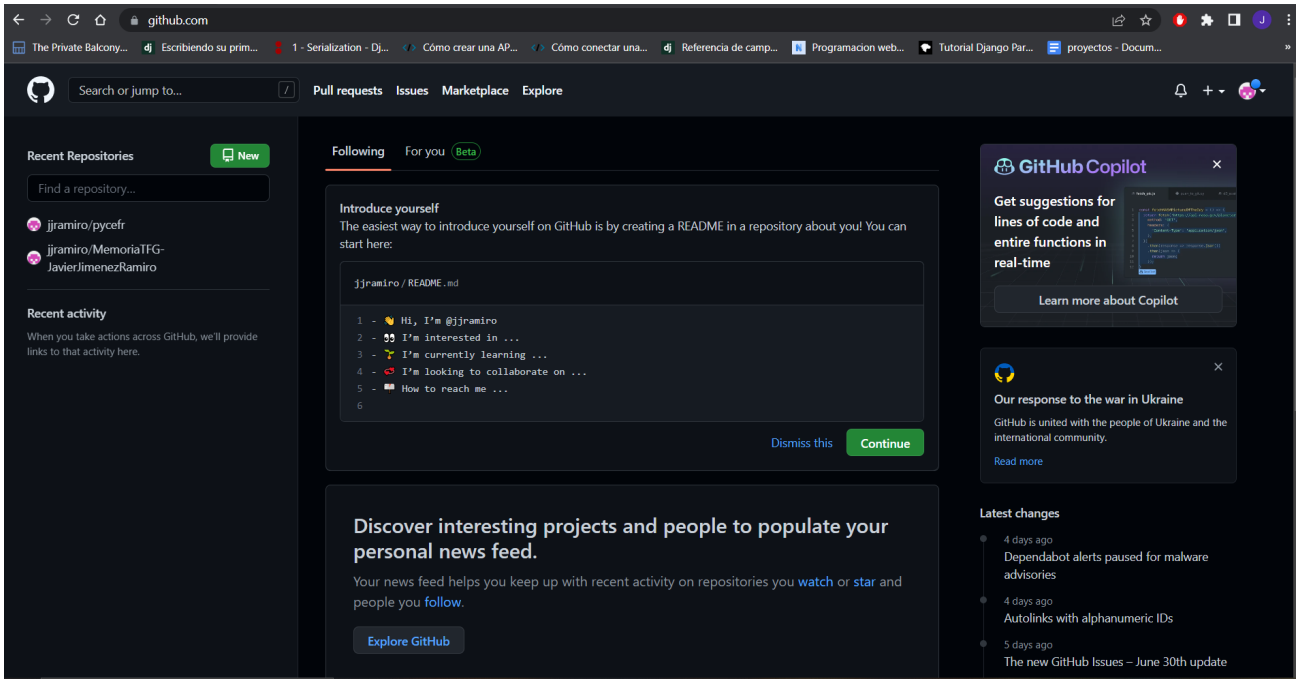

Figura 5.5: Página principal de GitHub

5.2. Manual de análisis Pull Request

Al igual que en la sección anterior con stack Overflow, ahora, pasamos a explicar el procedimiento a seguir para el análisis de los Pull Request.

Para ello lo primero que se debe hacer es entrar en la página web principal de GitHub y acceder a ella con nuestra cuenta:

`https://github.com/`

Una vez que entremos nos aparecerá una página como la de la figura 5.5, donde pincharemos en el botón de arriba y ligeramente a la izquierda llamado 'Explore'.

Aquí se pueden ver varios apartados dentro de 'Explore'. En estos apartados pincharemos en 'topic' y nos aparecerá algo parecido a la figura 5.6. Dentro de topic buscaremos el tema que nos interesa para el buen funcionamiento de nuestra herramienta que en nuestro caso es 'Python' y pincharemos en el.

Ahora, como podemos ver en la figura 5.7, pincharemos en la parte de 'language' y seleccionaremos Python, esto lo hacemos para asegurarnos de que los Pull Request son de este lenguaje y así nuestro análisis funcione correctamente.

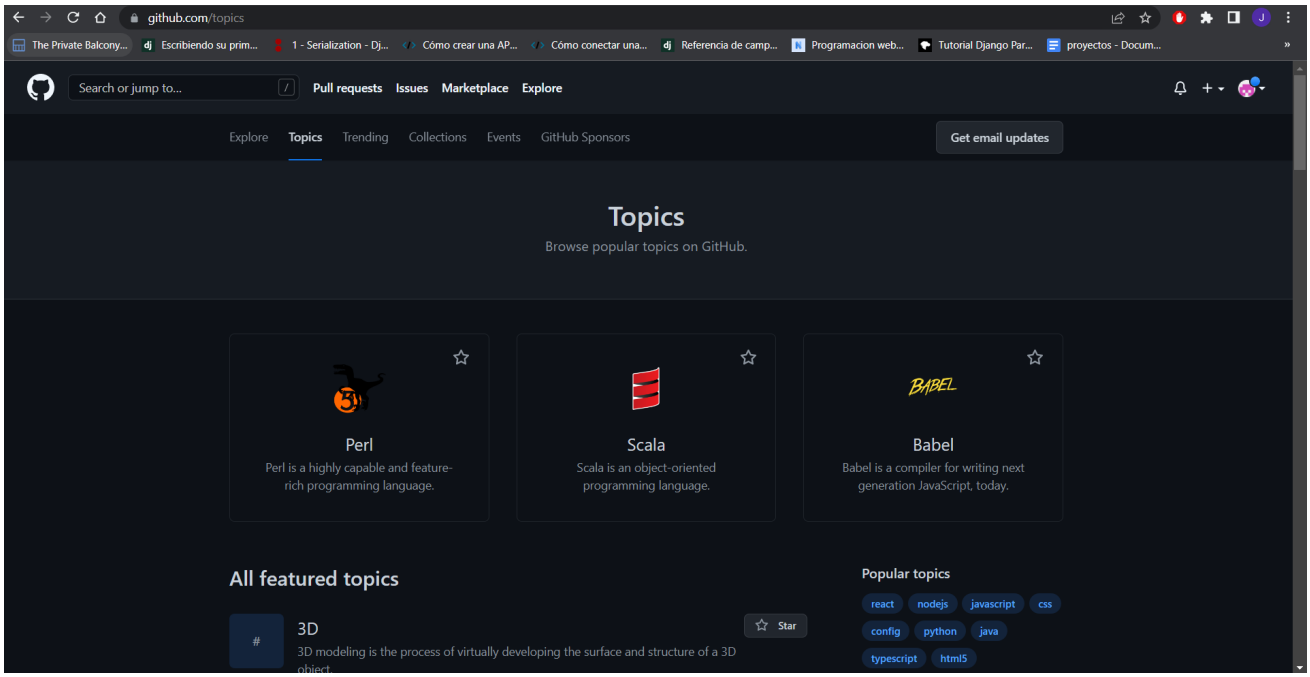

Figura 5.6: Página 'topic' de GitHub

Como resultado, se puede ver que aparecen una lista de repositorios de usuarios de GitHub y si nos fijamos bien en uno cualquiera podemos ver que tiene un botón donde se lee 'pull request'.

Eligiendo cualquiera de estos proyectos y pinchando en el botón mencionado en el párrafo anterior nos aparece lo que vemos en la figura 5.8.

Antes de continuar con el comando final crearemos, dentro del directorio de nuestra herramienta, el directorio donde se guardaran los fragmentos de código analizados de los Pull Request con el comando:

```
mkdir /test_directory
```

Por último, solo quedaría elegir cualquiera de estos 'pull', pinchar en el y copiar la URL resultante para poder ejecutar dentro del directorio de Pycefrl el siguiente comando:

```
python3 pycefrl.py pull <url_pull_request>
```

Al ejecutarlo nos debe dar una respuesta, al igual que con Stack Overflow, parecida a la figura 5.4.

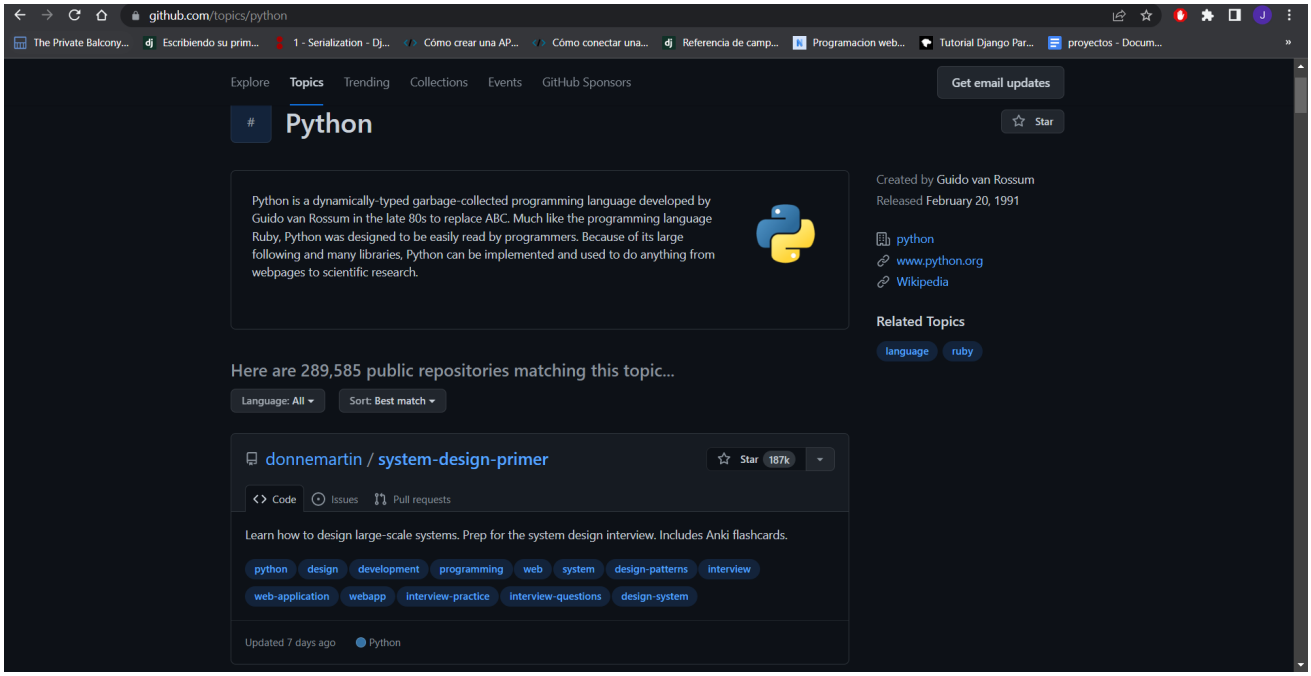

Figura 5.7: Página 'topic' de Python en GitHub

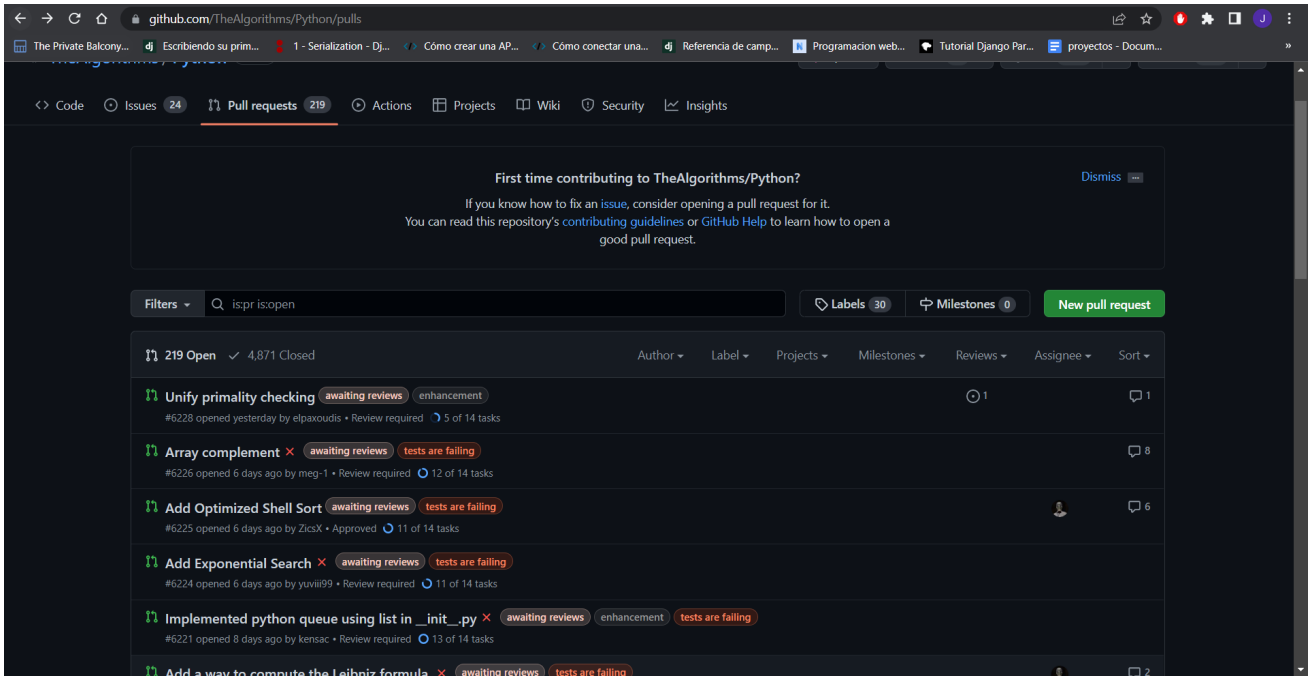

Figura 5.8: Página 'pull request' de un repositorio en GitHub

Capítulo 6

Experimentos y validación

En este punto ya sabemos como esta desarrollada la herramienta y el correcto modo de uso ejemplificado para conseguir un resultado favorable. Por lo tanto en este capítulo se recoge todas las pruebas y cálculos que se han hecho para aproximar el porcentaje de acierto de la herramienta en los casos de Stack Overflow y Pull Request.

Para estos dos casos se han probado, siguiendo el método explicado en el capítulo 5, cien URLs distintas para cada uno, es decir, doscientas URLs en total.

En el caso de Stack Overflow, cada una de las URL trata de una pregunta distinta, se han intentado evitar todas las preguntas que no tuvieran respuestas o bien que, aun filtrando en la página de preguntas por lenguaje Python, no contuvieran este lenguaje.

Por otro lado en los Pull Requests, se han intentado evitar aquellos que únicamente tenían fragmentos de código sin acabar, como por ejemplo un trozo de una función, o aquellos que directamente estuvieran mal escritos. También se han evitado todos aquellos que no contuvieran, aun filtrando por Python, este lenguaje.

En cuanto a los resultados, en el caso de Stack Overflow, se ha calculado el porcentaje de acierto de cada una de las URL y después el porcentaje total juntando todos los individuales sobre cien haciendo la media. Este calculo se ha hecho dividiendo todos aquellos fragmentos analizados correctamente frente a todos los fragmentos que han pasado el filtro correctamente, después, este resultado se ha multiplicado por 100 para sacar el porcentaje. En el caso de los Pull Request, se han sumado todas aquellas URL que sus fragmentos se han analizado correctamente, que al ser 100 las totales, nos daba directamente el porcentaje de acierto.

A continuación muestro en las figuras 6.1, 6.2, 6.3, 6.4, 6.5, 6.6, 6.7, 6.8, 6.9, 6.10, 6.11,

todas las URL de Stack Overflow seguidas de sus resultados y su porcentaje:

Por último, teniendo en cuenta todos los resultados mostrados y haciendo el porcentaje final, nos da como porcentaje de acierto completo un 70.4.

A continuación muestro en las figuras 6.12, 6.13, 6.14, 6.15, 6.16, 6.17, 6.18, todas las URL de los Pull Request seguidas de si se han analizado (y) correctamente o no(n):

Juntando todos los aciertos dentro de los Pull Request llegamos a un resultado de aciertos en total de ochenta y seis sobre cien, es decir, un 86 por ciento.

https://stackoverflow.com/questions/606191/convert-bytes-to-a-string 17/25=68%

https://stackoverflow.com/questions/510348/how-can-i-make-a-time-delay
22/28=78.5%

https://stackoverflow.com/questions/30081275/why-is-1000000000000000-in-range1000000000000001-so-fast-in-python-3 8/10=80%

https://stackoverflow.com/questions/7943751/what-is-the-python-3-equivalent-of-python-m-simplehttpserver 0/0=100%

https://stackoverflow.com/questions/230751/how-can-i-flush-the-output-of-the-print-function-unbuffer-python-output 9/14=64.3%

https://stackoverflow.com/questions/3289601/referring-to-the-null-object-in-python
19/30=63.3%

https://stackoverflow.com/questions/1471994/what-is-setup-py 5/8=62.2%

https://stackoverflow.com/questions/7585435/best-way-to-convert-string-to-bytes-in-python-3 5/6=83.3%

https://stackoverflow.com/questions/16981921/relative-imports-in-python-3
29/44=66%

https://stackoverflow.com/questions/36932/how-can-i-represent-an-enum-in-python
48/53=90.6%

https://stackoverflow.com/questions/16819222/how-to-return-dictionary-keys-as-a-list-in-python 10/14=71.4%

https://stackoverflow.com/questions/6908143/should-i-put-shebang-in-python-scripts-and-what-form-should-it-take 0/0=100%

https://stackoverflow.com/questions/33533148/how-do-i-type-hint-a-method-with-the-type-of-the-enclosing-class 18/18=100%

Figura 6.1: Resultados Stack Overflow (1)

https://stackoverflow.com/questions/16869024/what-is-pycache 0/2=0%

https://stackoverflow.com/questions/9233027/unicodedecodeerror-charmap-codec-cant-decode-byte-x-in-position-y-character 2/2=100%

https://stackoverflow.com/questions/14379753/what-does-mean-in-python-function-definitions 10/15=66.7%

https://stackoverflow.com/questions/33054527/typeerror-a-bytes-like-object-is-required-not-str-when-writing-to-a-file-in 9/12=75%

https://stackoverflow.com/questions/45310254/fixed-digits-after-decimal-with-f-strings 10/13=76.9%

https://stackoverflow.com/questions/517923/what-is-the-best-way-to-remove-accents-normalize-in-a-python-unicode-string 14/18=77.8%

https://stackoverflow.com/questions/954834/how-do-i-use-raw-input-in-python-3 6/7=85.7%

Figura 6.2: Resultados Stack Overflow (2)

https://stackoverflow.com/questions/1303347/getting-a-map-to-return-a-list-in-python-3-x 7/8=87.5%

https://stackoverflow.com/questions/11914472/how-to-use-stringio-in-python3 4/10=40%

https://stackoverflow.com/questions/30418481/error-dict-object-has-no-attribute-iteritems 3/4=75%

https://stackoverflow.com/questions/2792650/import-error-no-module-name-urllib2 10/12=83.3%

https://stackoverflow.com/questions/5105517/deep-copy-of-a-dict-in-python 6/7=85.7%

https://stackoverflow.com/questions/33945261/how-to-specify-multiple-return-types-using-type-hints 3/6=50%

https://stackoverflow.com/questions/2675028/list-attributes-of-an-object 15/18=83.3%

https://stackoverflow.com/questions/32370281/how-to-embed-image-or-picture-in-jupyter-notebook-either-from-a-local-machine-o 5/9=55.6%

https://stackoverflow.com/questions/23944657/typeerror-method-takes-1-positional-argument-but-2-were-given 15/16=93.8%

https://stackoverflow.com/questions/25445439/what-does-syntaxerror-missing-parentheses-in-call-to-print-mean-in-python 4/7=57.1%

https://stackoverflow.com/questions/7585307/how-to-correct-typeerror-unicode-objects-must-be-encoded-before-hashing 3/8=37.5%

https://stackoverflow.com/questions/436198/what-is-an-alternative-to-execfile-in-python-3 9/11=81.8%

Figura 6.3: Resultados Stack Overflow (3)

https://stackoverflow.com/questions/12169839/which-is-the-preferred-way-to-concatenate-a-string-in-python 10/17=58.8%

https://stackoverflow.com/questions/7243750/download-file-from-web-in-python-3 15/15=100%

https://stackoverflow.com/questions/14301967/bare-asterisk-in-function-arguments 3/6=50%

https://stackoverflow.com/questions/4915361/whats-the-difference-between-raw-input-and-input-in-python-3 2/3=66.7%

https://stackoverflow.com/questions/1347791/unicode-error-unicodeescape-codec-cant-decode-bytes-cannot-open-text-file 1/2=50%

https://stackoverflow.com/questions/14087598/python-3-importerror-no-module-named-configparser 2/4=50%

https://stackoverflow.com/questions/17192158/nameerror-global-name-xrange-is-not-defined-in-python-3 2/2=100%

Figura 6.4: Resultados Stack Overflow (4)

https://stackoverflow.com/questions/15286401/print-multiple-arguments-in-python 3/6=50%

https://stackoverflow.com/questions/17615414/how-to-convert-binary-string-to-normal-string-in-python3 0/0=100%

https://stackoverflow.com/questions/17534345/typeerror-missing-1-required-positional-argument-self 3/4=75%

https://stackoverflow.com/questions/13638898/how-to-use-filter-map-and-reduce-in-python-3 14/16=87.5%

https://stackoverflow.com/questions/9452108/how-to-use-string-replace-in-python-3-x 6/9=66.7%

https://stackoverflow.com/questions/37835179/how-can-i-specify-the-function-type-in-my-type-hints 4/5=80%

https://stackoverflow.com/questions/43728431/relative-imports-modulenotfounderror-no-module-named-x 14/20=70%

https://stackoverflow.com/questions/6624453/whats-the-correct-way-to-convert-bytes-to-a-hex-string-in-python-3 6/8=75%

https://stackoverflow.com/questions/24752395/python-raise-from-usage 2/6=33.3%

https://stackoverflow.com/questions/32557920/what-are-type-hints-in-python-3-5 7/10=70%

https://stackoverflow.com/questions/37139786/is-init-py-not-required-for-packages-in-python-3-3 0/1=0%

https://stackoverflow.com/questions/6930982/how-to-use-a-variable-inside-a-regular-expression 6/12=50%

https://stackoverflow.com/questions/13906623/using-pickle-dump-typeerror-must-be-str-not-bytes 1/1=100%

Figura 6.5: Resultados Stack Overflow (5)

https://stackoverflow.com/questions/39429526/how-to-specify-nullable-return-type-with-type-hints 1/4=25%

https://stackoverflow.com/questions/8908287/why-do-i-need-b-to-encode-a-string-with-base64 6/8=75%

https://stackoverflow.com/questions/19699367/for-line-in-results-in-unicodedecodeerror-utf-8-codec-cant-decode-byte 3/5=60%

https://stackoverflow.com/questions/18272160/access-multiple-elements-of-list-knowing-their-index 9/11=81.8%

https://stackoverflow.com/questions/10851906/python-3-unboundlocalerror-local-variable-referenced-before-assignment 3/6=50%

https://stackoverflow.com/questions/15014310/why-is-there-no-xrange-function-in-python3 3/9=33.3%

https://stackoverflow.com/questions/17140886/how-to-search-and-replace-text-in-a-file 20/22=90.9%

Figura 6.6: Resultados Stack Overflow (6)

https://stackoverflow.com/questions/28583565/str-object-has-no-attribute-decode-python-3-error 1/5=20%

https://stackoverflow.com/questions/2395160/what-is-the-correct-syntax-for-else-if 5/7=71.4%

https://stackoverflow.com/questions/1073396/is-generator-next-visible-in-python-3 3/5=60%

https://stackoverflow.com/questions/7818811/import-error-no-module-named-numpy 1/1=100%

https://stackoverflow.com/questions/28885132/why-is-x-in-x-faster-than-x-x 1/9=11.1%

https://stackoverflow.com/questions/826948/syntax-error-on-print-with-python-3 2/18=11.1%

https://stackoverflow.com/questions/5512811/builtins-typeerror-must-be-str-not-bytes 0/1=0%

https://stackoverflow.com/questions/4912852/how-do-i-change-the-string-representation-of-a-python-class 1/1=100%

https://stackoverflow.com/questions/21764770/typeerror-got-multiple-values-for-argument 2/3=66.7%

https://stackoverflow.com/questions/1282945/why-does-integer-division-yield-a-float-instead-of-another-integer 7/9=77.8%

https://stackoverflow.com/questions/20449427/how-can-i-read-inputs-as-numbers 0/2=0%

https://stackoverflow.com/questions/21017698/converting-int-to-bytes-in-python-3 12/13=92.3%

Figura 6.7: Resultados Stack Overflow (7)

https://stackoverflow.com/questions/40181344/how-to-annotate-types-of-multiple-return-values 12/19=63.2%

https://stackoverflow.com/questions/9282967/how-to-open-a-file-using-the-open-with-statement0/0=100%

https://stackoverflow.com/questions/9282967/how-to-open-a-file-using-the-open-with-statement 0/4=0%

https://stackoverflow.com/questions/37372603/how-to-remove-specific-substrings-from-a-set-of-strings-in-python 10/11=90.9%

https://stackoverflow.com/questions/5952344/how-do-i-format-a-string-using-a-dictionary-in-python-3-x 9/9=100%

https://stackoverflow.com/questions/11480042/python-3-turn-range-to-a-list 4/5=80%

https://stackoverflow.com/questions/21884271/warning-about-too-many-open-figures 6/7=85.7%

Figura 6.8: Resultados Stack Overflow (8)

https://stackoverflow.com/questions/27435284/multiprocessing-vs-multithreading-vs-asyncio-in-python-3 7/7=100%

https://stackoverflow.com/questions/24853923/type-hinting-a-collection-of-a-specified-type 5/5=100%

https://stackoverflow.com/questions/18169965/how-to-delete-last-item-in-list 3/4=75%

https://stackoverflow.com/questions/51710037/how-should-i-use-the-optional-type-hint 2/7=28.6%

https://stackoverflow.com/questions/13905741/accessing-class-variables-from-a-list-comprehension-in-the-class-definition 20/25=80%

https://stackoverflow.com/questions/27385633/what-is-the-symbol-for-in-python 1/4=25%

https://stackoverflow.com/questions/34283178/typeerror-a-bytes-like-object-is-required-not-str-in-python-and-csv 3/4=75%

https://stackoverflow.com/questions/21285380/find-column-whose-name-contains-a-specific-string 6/7=85.7%

https://stackoverflow.com/questions/18982610/difference-between-except-and-except-exception-as-e 1/4=25%

https://stackoverflow.com/questions/24487405/getting-value-of-enum-on-string-conversion 7/7=100%

https://stackoverflow.com/questions/19201290/how-to-save-a-dictionary-to-a-file 14/17=82.4%

https://stackoverflow.com/questions/39327032/how-to-get-the-latest-file-in-a-folder 9/12=75%

Figura 6.9: Resultados Stack Overflow (9)

https://stackoverflow.com/questions/33727149/dict-object-has-no-attribute-has-key 4/5=80%

https://stackoverflow.com/questions/6862770/let-json-object-accept-bytes-or-let-urlopen-output-strings 8/8=100%

https://stackoverflow.com/questions/2823316/generate-a-random-letter-in-python 14/16=87.5%

https://stackoverflow.com/questions/18552001/accessing-dict-keys-element-by-index-in-python3 5/5=100%

https://stackoverflow.com/questions/13998492/when-should-iteritems-be-used-instead-of-items 4/4=100%

https://stackoverflow.com/questions/19877306/nameerror-global-name-unicode-is-not-defined-in-python-3 6/7=85.7%

https://stackoverflow.com/questions/19507808/python-3-integer-division 1/1=100%

https://stackoverflow.com/questions/68152730/understand-python-swapping-why-is-a-b-b-a-not-always-equivalent-to-b-a-a 11/14=78.6%

Figura 6.10: Resultados Stack Overflow (10)

https://stackoverflow.com/questions/10058140/accessing-items-in-an-collections-ordereddict-by-index 13/18=72.2%

https://stackoverflow.com/questions/17322668/typeerror-dict-keys-object-does-not-support-indexing 2/2=100%

Figura 6.11: Resultados Stack Overflow (11)

https://github.com/TheAlgorithms/Python/pull/6169 Y

https://github.com/TheAlgorithms/Python/pull/6167 Y

https://github.com/TheAlgorithms/Python/pull/6165 N

https://github.com/TheAlgorithms/Python/pull/6143 N

https://github.com/TheAlgorithms/Python/pull/6137 Y

https://github.com/TheAlgorithms/Python/pull/6121 Y

https://github.com/TheAlgorithms/Python/pull/6119 N

https://github.com/TheAlgorithms/Python/pull/6117 N

https://github.com/TheAlgorithms/Python/pull/6105 Y

https://github.com/TheAlgorithms/Python/pull/6086 Y

https://github.com/TheAlgorithms/Python/pull/6061 Y

https://github.com/TheAlgorithms/Python/pull/6057 N

https://github.com/TheAlgorithms/Python/pull/6044 N

https://github.com/TheAlgorithms/Python/pull/6042 Y

https://github.com/TheAlgorithms/Python/pull/6036 Y

https://github.com/TheAlgorithms/Python/pull/6025 N

https://github.com/TheAlgorithms/Python/pull/5978 Y

https://github.com/TheAlgorithms/Python/pull/5969 Y

https://github.com/TheAlgorithms/Python/pull/5966 Y

Figura 6.12: Resultados Pull Request (1)

https://github.com/TheAlgorithms/Python/pull/5962 Y

https://github.com/TheAlgorithms/Python/pull/5961 Y

https://github.com/TheAlgorithms/Python/pull/5955 N

Figura 6.13: Resultados Pull Request (2)

https://github.com/TheAlgorithms/Python/pull/5953	Y
https://github.com/TheAlgorithms/Python/pull/5948	Y
https://github.com/TheAlgorithms/Python/pull/5942	Y
https://github.com/TheAlgorithms/Python/pull/5937	Y
https://github.com/TheAlgorithms/Python/pull/5926	Y
https://github.com/TheAlgorithms/Python/pull/5903	Y
https://github.com/TheAlgorithms/Python/pull/5885	N
https://github.com/TheAlgorithms/Python/pull/5875	Y
https://github.com/TheAlgorithms/Python/pull/5873	Y
https://github.com/TheAlgorithms/Python/pull/5872	Y
https://github.com/TheAlgorithms/Python/pull/5870	Y
https://github.com/TheAlgorithms/Python/pull/5864	Y
https://github.com/TheAlgorithms/Python/pull/5852	N
https://github.com/TheAlgorithms/Python/pull/5849	Y
https://github.com/TheAlgorithms/Python/pull/5842	Y
https://github.com/TheAlgorithms/Python/pull/5839	Y
https://github.com/TheAlgorithms/Python/pull/5823	Y
https://github.com/TheAlgorithms/Python/pull/5776	Y
https://github.com/TheAlgorithms/Python/pull/5729	Y
https://github.com/TheAlgorithms/Python/pull/5727	Y

Figura 6.14: Resultados Pull Request (3)

https://github.com/TheAlgorithms/Python/pull/5720 Y

https://github.com/TheAlgorithms/Python/pull/5693 Y

https://github.com/TheAlgorithms/Python/pull/5692 Y

https://github.com/TheAlgorithms/Python/pull/5687 Y

https://github.com/TheAlgorithms/Python/pull/5673 Y

https://github.com/TheAlgorithms/Python/pull/5668 Y

https://github.com/TheAlgorithms/Python/pull/5664 Y

https://github.com/TheAlgorithms/Python/pull/5646 Y

https://github.com/TheAlgorithms/Python/pull/5637 Y

https://github.com/TheAlgorithms/Python/pull/5602 Y

https://github.com/TheAlgorithms/Python/pull/5568 Y

Figura 6.15: Resultados Pull Request (4)

https://github.com/TheAlgorithms/Python/pull/5562 Y

https://github.com/TheAlgorithms/Python/pull/5531 N

https://github.com/TheAlgorithms/Python/pull/5529 Y

https://github.com/TheAlgorithms/Python/pull/5522 Y

https://github.com/TheAlgorithms/Python/pull/5515 Y

https://github.com/TheAlgorithms/Python/pull/5505 Y

https://github.com/TheAlgorithms/Python/pull/5487 Y

https://github.com/TheAlgorithms/Python/pull/5484 Y

https://github.com/TheAlgorithms/Python/pull/5467 Y

https://github.com/TheAlgorithms/Python/pull/5457 Y

https://github.com/TheAlgorithms/Python/pull/5454 Y

https://github.com/TheAlgorithms/Python/pull/5432 Y

https://github.com/TheAlgorithms/Python/pull/5428 Y

https://github.com/TheAlgorithms/Python/pull/5427 Y

https://github.com/TheAlgorithms/Python/pull/5422 Y

https://github.com/TheAlgorithms/Python/pull/5418 Y

https://github.com/TheAlgorithms/Python/pull/5413 Y

https://github.com/TheAlgorithms/Python/pull/5403 Y

https://github.com/TheAlgorithms/Python/pull/5397 Y

https://github.com/TheAlgorithms/Python/pull/5395 Y

Figura 6.16: Resultados Pull Request (5)

https://github.com/TheAlgorithms/Python/pull/5382 Y

https://github.com/TheAlgorithms/Python/pull/5376 Y

https://github.com/TheAlgorithms/Python/pull/5365 Y

https://github.com/TheAlgorithms/Python/pull/5364 Y

https://github.com/TheAlgorithms/Python/pull/5359 N

https://github.com/TheAlgorithms/Python/pull/5356 Y

https://github.com/TheAlgorithms/Python/pull/5351 Y

https://github.com/TheAlgorithms/Python/pull/5336 Y

https://github.com/TheAlgorithms/Python/pull/5329 N

https://github.com/TheAlgorithms/Python/pull/5328 Y

https://github.com/TheAlgorithms/Python/pull/5325 Y

Figura 6.17: Resultados Pull Request (6)

https://github.com/TheAlgorithms/Python/pull/5314 Y

https://github.com/TheAlgorithms/Python/pull/5313 Y

https://github.com/TheAlgorithms/Python/pull/5312 Y

https://github.com/TheAlgorithms/Python/pull/5286 Y

https://github.com/TheAlgorithms/Python/pull/5281 Y

https://github.com/TheAlgorithms/Python/pull/5269 Y

https://github.com/TheAlgorithms/Python/pull/5262 Y

https://github.com/TheAlgorithms/Python/pull/5255 Y

https://github.com/TheAlgorithms/Python/pull/5254 Y

https://github.com/TheAlgorithms/Python/pull/5252 Y

https://github.com/TheAlgorithms/Python/pull/5235 Y

https://github.com/TheAlgorithms/Python/pull/5232 Y

https://github.com/TheAlgorithms/Python/pull/5229 Y

https://github.com/TheAlgorithms/Python/pull/5226 Y

https://github.com/TheAlgorithms/Python/pull/5210 Y

Figura 6.18: Resultados Pull Request (7)

Capítulo 7

Conclusiones

7.1. Consecución de objetivos

Como se comentó en el capítulo 2, el objetivo de este proyecto era el de ampliar la herramienta ya existente y creada por mi compañera Ana Poveda García Herrero con dos nuevas opciones de análisis, Stack overflow y Pull Request de GitHub, para poder cerciorarnos de como de bueno es un código y como de útil puede ser implementarlo en nuestros proyectos dependiendo de su nivel. Por lo tanto el objetivo general se ha cumplido.

En cuánto a los objetivos específicos, podemos observar que también han sido cumplidos aunque esta vez con complicaciones y no tan bien perfeccionados como se podría esperar.

Esto ocurre por las limitaciones propias que nos presentan estas dos opciones y que tienen una alta dificultad para solucionarlas y se podría definir como la ”libertad”que nos dan estas opciones. Esta libertad trata de que tanto en Stack Overflow como en los Pull Request de GitHub cualquier usuario puede aportar cualquier cosa, pudiendo añadir, dentro del filtrado de Python, otros lenguajes, texto plano o incluso fragmentos de código mal escritos o sin acabar.

Para este tipo de problemas lo único que tenemos como ’defensa’ es una excepción ya creada en la anterior versión de la herramienta.

Y aunque es cierto que, como vemos en el capítulo 6, tenemos unos resultados muy favorables y con un buen porcentaje de acierto, hay que tener en cuenta que tienen que darse las condiciones de que el lenguaje sea Python3 y que el código este bien escrito y acabado para que el análisis sea correcto. Quitando estas dificultades bastante complicadas de abordar podemos asegurar que los objetivos se han cumplido satisfactoriamente.

7.2. Aplicación de lo aprendido

En cuanto a la aplicación de lo aprendido, considero que todas las asignaturas del grado y todos los profesores con los que he tenido oportunidad de encontrarme han aportado, directa o indirectamente, su granito de arena para hacerme llegar hasta aquí, bien sea por los conocimientos aprendidos o por la manera de enseñarme a gestionar cierto tipo de problemas con los que me he encontrado en este último año, pero sí me gustaría destacar algunas asignaturas que han sobresalido:

1. **Informática 1:** Esta asignatura fue mi primer contacto con la programación y me ayudó a saber que había escogido el camino correcto y que todos los problemas son divisibles.

2. **Informática 2:** Esta asignatura me planteo nuevos retos, me enseñó la importancia de los paquetes y de las librerías y me enseñó que el mundo de la programación es inmenso y repleto de buenas ideas.

3. **Protocolos para la transmisión de audio y vídeo por Internet (ptavi):** Esta asignatura me ayudó a descubrir el que es mi lenguaje favorito, Python, y sus profesores me transmitieron aun más pasión por el mundo de la programación. Por ello tenía claro que mi proyecto tenía que estar orientado a Python.

4. **Prácticas externas:** Aunque no es una asignatura como tal, el haber hecho mis prácticas con Python me ayudó a consolidar más mis conocimientos sobre el lenguaje y me hizo mucho más fácil el hecho de poder depurar errores y dejar el código todo lo limpio que supe.

7.3. Lecciones aprendidas

Además de todo lo aprendido en mi paso por el grado, he aprendido lecciones muy importantes de este proyecto, las cuales son:

- Este proyecto me ha ayudado a consolidar aun más los conocimientos sobre Python, sobre sus librerías y a ser algo más independiente a la hora de programar en el. Y aunque aun me queda camino por recorrer, sé que estoy en el sendero correcto.

- También me ha ayudado a entender mejor el funcionamiento tanto de Stack overflow como de GitHub, más concretamente los Pull Request. Ya que al tener que analizarlos y utilizarlos me he dado cuenta de todas las posibilidades que ofrecen cada una de estas webs.

- Otra lección aprendida es el uso de LaTeX. Me ha hecho darme cuenta de lo útil que puede llegar a ser este lenguaje para la creación y edición de proyectos y lo cómodo que resulta trabajar con el.

- Como última lección destacable añadiría la gestión y organización de un proyecto algo más grande. La realización de este proyecto me ha ayudado a saber organizar mejor este tipo de proyectos más prolongados en el tiempo.

7.4. Trabajos futuros

La ampliación de esta herramienta y todo lo desarrollado en este proyecto es solo una pequeña mejora de todo lo que se podría añadir. Hay un gran abanico de mejoras y funcionalidades que se podrían implementar por lo que aquí dejo algunas ideas:

- Se podría mejorar con un analizador propio la cantidad de lenguajes a analizar añadiendo por ejemplo Python2 u otro tipo de lenguajes.

- Se pueden añadir más filtros y aun más controles de errores para mejorar el porcentaje de acierto de las nuevas opciones.

- Se pueden añadir nuevas funcionalidades para que en el resultado no solo aparezca tu nivel si no también como mejorarlo y que elementos implementar para elevar el nivel de tu proyecto.

Bibliografía

[1] Github.
https://docs.github.com/es.

[2] Html python.
https://docs.python.org/3/library/html.html.

[3] Json.
https://abrirarchivos.info/extension/json.

[4] Python.
https://docs.python.org.

[5] Stack overflow.
https://stackoverflow.com/documentation.

[6] Urllib.request.
https://docs.python.org/3/library/urllib.request.html.