SISTEMA DE ALERTAS PARA NOTICIAS

Miembros del grupo
Álvaro Hidalgo López
Eugenio Palacios París
Raúl Sánchez de Saboya
Luis Sánchez Asunción

Contenido

1. Introducción.

La propuesta elegida es:

- **Sistema de Alerta de Noticias Personalizado**: Implementar un sistema que recopile noticias basadas en intereses específicos del usuario y las envíe de forma periódica.

Se ha elegido esta propuesta porque para este grupo era una idea interesante poder recoger noticias de gente de interés o acontecimientos sin tener que buscar entre el resto de las noticias usando palabras clave. De esta manera, se ahorra tiempo y se evitan distracciones con noticias que no son relevantes.

1.1. Herramientas, librerías y lógica usadas.

Dado que se va a obtener información de páginas de noticias que normalmente son estáticas, y esto quiere decir que no se va a tener que hacer captchas o algo parecido.

Pero, en este caso, no se elige la librería **BeautifulSoap** para realizar consultas de información en sus HTML, sino que, simplemente se recurre a **Requests**.

Esta hace las consultas mucho más simples y es ideal para acceder a información mediante la **NEWSAPI [2]** y saca formato texto o **JSON**, siendo este último este caso.

Para ello se coge la **NEWSAPI_KEY** que se carga de las variables de entorno.

La biblioteca **Transformers** nos sirve para coger el **pipeline** para usar modelos, cuya finalidad es hacer resúmenes de textos de las noticias.

Por otro lado, login sirve para que no salten ciertos avisos sobre los límites de texto (todo esto se aplica en la función **dynamic_summarize**).

Después se obtienen las noticias de dos maneras:

1. **A través de la NEWSAPI**:

   Se definen los campos que queremos en una variable de la siguiente manera:

   Primero, se coge la url = **"https://newsapi.org/v2/everything".**

   Después, se define lo siguiente:

```
params = {
    "q": topic,
    "language": language,
    "pageSize": page_size,
    "sortBy": "publishedAt",
}
```

   También se define el **Header** con el que se hace la **request** usando la clave:

```
headers = {
    "Authorization": f"Bearer {NEWSAPI_KEY}"
}
```

   Por último, se coge el **title**, **description** y **url** del artículo de la respuesta en JSON.

2. **A través de una lista:**
   La única diferencia con la anterior es que se establen dominios de diferentes páginas de noticias para consultas.

   Tras esto definimos una función personalizada para determinar el tipo de artículo y así filtrarlos para que solo queden los de este tipo (**filter_by_article_type**).

   Durante el proceso también definimos una función para pasarle los argumentos o parámetros del programa definiendo el tipo de noticia, que palabras clave debe tener, etc.

   La librería Time sirve para ponerle un límite a la consulta de noticias dado que podemos sobrecargar las páginas, además de que pueden tomar medidas como bloquear la IP.

   La librería Schedule sirve para ejecutar de forma periódica el programa a diario.

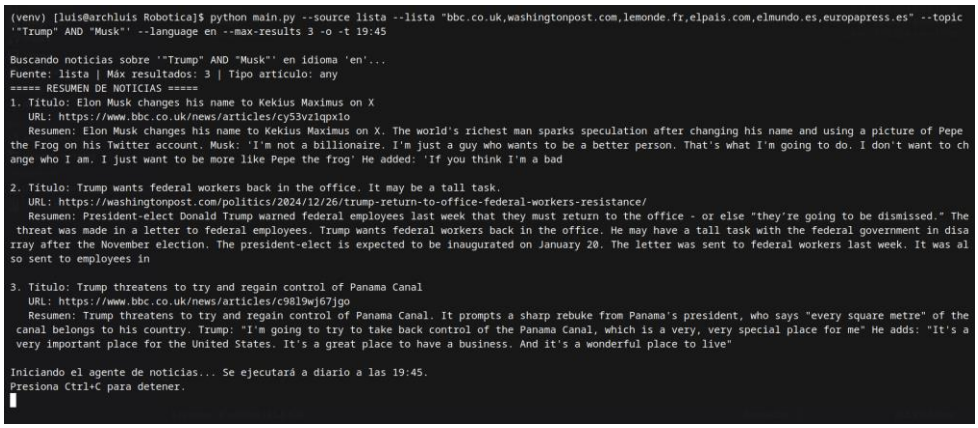

Ilustración 1: Ejecución del programa por lista.

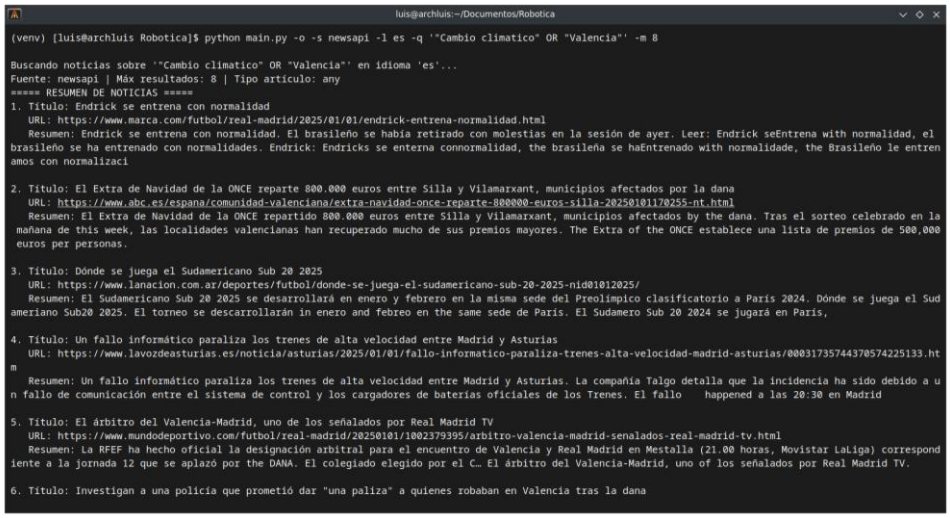

Ilustración 2: Ejecución con newsapi.

2. Conclusiones.

Esta propuesta, en comparación con otras, no tiene que tratar bibliotecas como **Selenium** para pasar sistemas **antibot** como captchas, ya que no se tratan de webs dinámicas.

Sin embargo, sí que se han añadido cosas que pueden ayudar a obtener noticias un poco más a **gusto del consumidor**, como, por ejemplo, el **idioma** en que se buscan o de que **tipo** queremos la noticia.

La ventaja principal, es que al utilizar la API **no estamos infringiendo términos de estas páginas**, pero no se profundiza mucho más al no utilizar estas otras librerías.

A diferencia de otras propuestas como la de **Steam**, que necesitaría **Selenium**, lo que quiere decir que se necesitaría saber que **navegador** se va a usar y que **driver** coger, también se tiene que añadir la **lógica** para tratar sistemas antibot, además que esto también **inflige sus normas**, lo cual puede derivar en bloqueos de IP o cosas más **graves**.

3. Ideas de futuro y posibles mejoras.

1. **Añadir notificación por Telegram o email**. El objetivo es que cada día te mande un resumen con las noticias como si fuera una Newsletter, pero con agentes de manera automática de medios variados.
2. **Hacer GUI de forma que sea más fácil y accesible el uso a cualquier usuario**, además de permitir la lectura en la propia aplicación donde puedas elegir cosas como fuente y tamaño de la letra.
3. Guardar **historico** y permitir **búsqueda posterior**.
4. Implementar un **clasificador** entrenado más preciso.

4. Referencias.

[1] Lo visto en la asignatura: https://github.com/etsisi/Robotica/tree/main/Slides

[2] Documentación de Newsapi: https://newsapi.org/docs

[3] Modelo de resumen: https://github.com/huggingface/transformers

[4] Librería de schedule: https://pypi.org/project/schedule/

5. Anexos.

Código del proyecto: Repositorio GitHub con el código del proyecto.