Máster Universitario de Ciencia de Datos

Tipología y ciclo de vida de los datos

Práctica 1: ¿Cómo podemos capturar los datos de la web?

Autores:

Carlos Díez Domínguez
Jacobo Osorio Ríos

1. Contexto

En esta práctica, se ha creado un conjunto de datos mediante técnicas de scraping empleadas en la página web *Reddit* (en concreto, en el hilo WallStreetBets: https://www.reddit.com/r/wallstreetbets/new/). Tal y como la definen sus creadores, *Reddit* es "*una red de comunidades donde la gente puede indagar en sus intereses, hobbies o pasiones*". Debido a su gran popularidad, actualmente existen miles de comunidades activas en las que usuarios de todo el mundo comparten información de todo tipo de temas. Es por esto que Reddit es, en términos generales, una fuente muy interesante de extracción de datos, particularmente si estamos interesados en análisis de datos mediante técnicas de *Natural Language Processing*.

En particular, una de las comunidades con más tráfico en los últimos meses ha sido *WallStreetBets*, un hilo donde la gente comparte sus opiniones sobre distintos activos, como acciones de empresas que cotizan en Wall Street o criptomonedas. Como veremos en los próximos apartados, esta alta actividad permite extraer datos de valor y con alta frecuencia con utilidad en distintos ámbitos, como la predicción de valores de acciones o la toma de decisiones por parte de una empresa en relación a comentarios de sus inversores.

2. Título

Hemos decidido asignarle al conjunto de datos el título ***Reddit's WallStreetBets Posts*****,** ya que creemos que define adecuadamente de dónde se han extraído los datos (el subreddit/foro *WallStreetBets* de Reddit), y qué es lo que se ha extraído (datos asociados a los posts publicados en este subreddit).

3. Descripción del dataset

El dataset está constituido por datos *scrapeados* exclusivamente del hilo *WallStreetBets*, que forma parte de la página web *Reddit.* En términos generales, el conjunto de datos consta de información relacionada con *posts* y los comentarios que han hecho los usuarios de la plataforma sobre estos.

En particular, el conjunto de datos es un único archivo *csv* de **3329 filas** (la primera el encabezado), donde las 216 primeras corresponden a 216 *posts* y las 3112 restantes a comentarios sobre estos posts. En cuanto a las **columnas** (que describiremos en el apartado 5), dentro del archivo podemos encontrar 7 distintas: ***id, ID_padre, Usuario, Fecha, Votos, Titulo*** y ***Texto***.

Con respecto al diseño del dataset, nos gustaría realizar las siguientes aclaraciones. A la hora de realizar el *scraping,* se impusieron los siguientes límites para no sobredimensionar el conjunto de datos:

- Se han de *scrapear* como máximo 250 posts.
- Se limita el número de comentarios máximos *scrapeados* por post a 20.
- Los *posts* no han de tener una antigüedad mayor a 2 semanas (14 días). Esto se ha impuesto también debido a que nos interesan *posts* y comentarios únicamente a corto plazo.

Como se explicará en el apartado 9, cerca de 1 de cada 5 *posts scrapeados* eran contenido publicitario, que hemos decidido desechar durante el proceso de *scraping.* En cualquier caso, hemos decidido contarlos como *posts scrapeados,* y es por este motivo que finalmente tenemos 216 *posts* reales.

En relación al límite de la fecha, en este caso no ha tenido efecto, ya que el último *post* que *escrapeado* corresponde al 19 de abril , y el primero, al 23 de abril (Reddit proporciona los *posts* en orden cronológico).

Finalmente, con respecto a la estructura del *dataset*, no gustaría mencionar que las primeras 216 filas se corresponden con los posts, y las últimas 3112 con los comentarios. En ambos casos, las filas están ordenadas por orden en el que se han ido haciendo *scrape a* los datos.

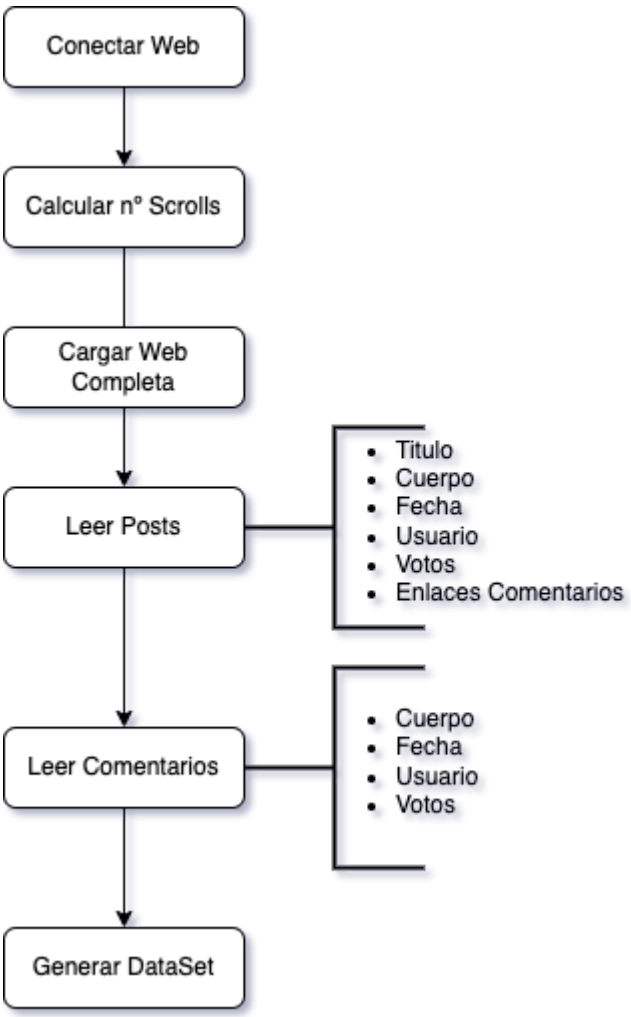
4. Representación gráfica

El proceso comienza conectándonos a la página web que queremos scrapear. Para realizar esta conexión y el *scraping* usaremos principalmente las librerías *Selenium* y *BeautifulSoup* de Python.

A priori, el número de *posts* que se pueden hacer *scraping* en una carga de la página es indeterminado, ya que tenemos una página web con *scroll* infinito. Por este motivo, hemos considerado prudente estimar el número de *posts* por cada scroll de la página para saber cuántos scrolls deberíamos hacer para poder *scrapear* el número de *posts* que deseamos.

Una vez cargada toda la página con todos los posts, procedemos a parsear el HTML y realizar el *scrape* de los *posts* (incluidos los enlace a los comentarios de cada *post*). Hecho esto, nos dirigimos a cada uno de los enlaces de los comentarios guardados y hacemos el *scrape* de cada uno.

El *scrape* de los *posts* y comentarios generan dos tablas (*dataframes)* separados con las mismas columnas, que podemos unir para generar el fichero final del *dataset*.



5. Contenido

Los campos que contiene el conjunto de datos son los siguientes:

- **id (**entero): número identificativo de un comentario o post
- **ID_padre** (entero): id del *post* al que está asociado un comentario. Puesto que los posts no tienen un *post* padre, para ellos se ha asignado por defecto el valor de -1. Hemos decidido incluir este campo para tener los comentarios y posts en una misma tabla en vez de en dos relacionadas.
- **Usuario** (cadena de caracteres): nombre del usuario que ha publicado el comentario o *post*
- **Fecha** (formato datetime D-M-A, H-M-S): fecha en la que se ha publicado un comentario o post. La precisión de la fecha (si se incluyen segundos, horas, minutos, etc…) varía en función de la precisión proporcionada por Reddit.
- **Votos** (cadena de caracteres): número de votos que tiene un *post* o comentario. Se establece como cadena de caracteres porque requiere limpieza para ser utilizado como tipo entero.

- **Título** (cadena de caracteres): título/encabezado de un *post.* Al no aplicar para los comentarios, en estos casos se ha asignado un -1 por defecto.
- **Texto** (cadena de caracteres): cuerpo de texto de un *post* o comentario.

Con respecto al periodo de tiempo de los datos, el dataset recoge *posts* y comentarios realizados entre el 19 y el 23 de abril de 2023.

6. Propietario

Reddit es una plataforma donde se agrupan diferentes foros de discusión de cualquier temática. Aquí los usuarios pueden compartir noticias y realizar comentarios, tanto de las noticias como de otros comentarios, generando interesantes debates. Las diferentes temáticas que existen abarcan desde los videojuegos, deportes, negocios, finanzas, etc..

En estos documentos explican como la publicación de comentarios en esta comunidad puede estar relacionada con importantes movimientos de los mercados, como por ejemplo, el famoso caso de la empresa *GameStop:*

- WallStreetBets Against Wall Street: The Role of Reddit in the GameStop Short Squeeze
- https://arxiv.org/ftp/arxiv/papers/2105/2105.02728.pdf

En relación a los conjuntos de datos usados en estos análisis, o *datasets* similares, hemos encontrado que en su mayoría se han obtenido los datos a partir de la propia API de Reddit. Así pues, dejando de lado los datos del conjunto, una diferencia fundamental entre nuestro *dataset* y los encontrados es que nosotros lo hemos generado a partir de *web scraping*.

Con respecto a los datos obtenidos, quizás la diferencia más importante es que nuestro conjunto de datos contiene los comentarios asociados a los *posts*, algo que no hemos encontrado en ningún otro dataset sobre *WallStreetBets*. Como ejemplo de conjunto de datos similar, añadimos un enlace a un *dataset* bastante popular en *Kaggle*:

- Reddit WallStreetBets Posts | Kaggle

Finalmente, nos gustaría mencionar que no hemos encontrado ningún obstáculo en términos de ética a la hora de hacer el *scraping* de la web. Al fin y al cabo, los datos son de acceso público, no tienen ningún tipo de información sensible, y ya existen conjuntos de datos similares.

7. Inspiración

Este proyecto nos parece interesante para preparar un conjunto de datos que pueden dar mucho juego al ser analizado mediante técnicas de *Natural Language Processing.* Algunos de los objetivos que se podrían proponer al realizar esto serían los siguientes:

- Conocer la opinión general de los inversores o clientes de una empresa para tomar decisiones empresariales
- Comparar las noticias publicadas en distintos medios de información sobre una empresa y lo que dice el público generar.
- Tratar de buscar correlaciones o predecir el valor de un activo en base a lo que se dice sobre él en foros de Internet (este sería el caso del estudio sobre GameStop comentado en el apartado anterior) .

Concebimos este proyecto como una parte esencial para realizar esto, aunque entendemos que la generación de este conjunto de datos es una pieza de un proyecto de mayor envergadura, ya que también sería necesario recolectar otros tipos de datos (p.ej. valor de cotización histórica de activos, noticias de distintos periódicos, etc…).

8. Licencia

La licencia escogida es **Creative Commons Attribution 4.0 International,** que es una licencia muy poco restrictiva: permite copiar y distribuir el conjunto en cualquier medio y formato, así como modificarlo para cualquier propósito. La razones principales es que los datos publicados son de dominio público (refiriéndonos a que se encuentra en una página completamente abierta), y que realmente no tenemos ningún derecho sobre la página ni el contenido a partir del cual hemos obtenido el conjunto de datos . Asimismo, esta licencia es bastante habitual en conjuntos de datos obtenidos de forma similar, por lo que hemos considerado que en este caso también sería apropiada.

9. Código

El código consta principalmente de dos archivos: ***perform_scrape.py*** y ***scraper.py***. El primero es un archivo que actúa como *pipeline* y que se puede llamar desde línea de comandos para realizar el *scraping.* Admite 5 argumentos: la URL que se ha de *scrapear*, la ruta de destino del *csv* que se genera, el número de *posts* a scrapear, el límite de comentarios por *post* a scrapear, y el límite de antigüedad de los posts. Asimismo, esta

pipeline también imprime en la terminal el User Agent, que es **Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/112.0.0.0 Safari/537.36** .

El segundo archivo reúne todas aquellas funciones y clases que son necesarias para realizar el *scraping*, y que se llaman en la *pipeline.*

Como hemos comentado en el apartado 4, Reddit es un sitio web que muestra **contenido de forma dinámica** (en concreto, a través de ***scroll infinito***). Para gestionar esto, hemos decidido trabajar con la librería de Python ***Selenium***, que permite simular interacciones con una página web.

Esta librería sirve, entre otras cosas, para ejecutar el código de JavaScript que nos permite cargar la página web en un navegador e ir simulando *scrolls.* De este modo, podemos ir cargando contenido para *scrapearlo* posteriormente.

Puesto que a priori desconocemos el número de *posts* que se cargan por *scroll,* hemos decidido en primer lugar hacer una **estimación del número de *scrolls* necesarios para obtener el número de *posts* deseados (**función *selenium_heuristic_computator)*. El método escogido es bastante sencillo, y consiste en realizar 5 *scrolls* iniciales y computar la media de *posts* cargados. La razón por la que hemos hecho esto es principalmente la de evitar tener que *parsear* el HTML de la página web cada vez que hacemos un único scroll, tarea que se realiza con la librería ***BeautifulSoup***. En cualquier caso, el código está diseñado de tal manera que si la estimación no fuese suficiente, se vayan haciendo estimaciones sucesivas hasta alcanzar el número de *posts* requeridos.

Con respecto al proceso de carga de la página web, también nos parece importante mencionar que hemos añadido una **demora de 2 segundos** cada vez que se hace un scroll para dar tiempo a que la página se cargue, así como para no saturar el servidor de Reddit. Otra funcionalidad que hemos incluido es también la de **no cargar contenido audiovisual**, ya que en este caso no se necesita y ralentiza la carga de la página. Finalmente, también consideramos añadir la funcionalidad de **navegación *headless***, que simularía la interacción con el navegador web sin necesidad de abrirlo en nuestra pantalla. No obstante, decidimos dejar las líneas del código respectivas comentadas para poder evaluar en nuestra pantalla el proceso de navegación.

Una vez cargado el total de la página que nos interesa (a través de la función *selenium_scroller*), procedimos a realizar el *scraping*. En primer lugar, nos centramos en obtener los datos de cada *posts (*función *scrape_posts*), que aparecen en orden cronológico (el primero es el más reciente). Aprovechando el parseo del HTML realizado, y conociendo las **clases identificadoras de los elementos en el HTML** que nos interesan (que hemos buscado previamente leyendo el HTML), hemos escrapeado y guardados en listas separadas los siguientes campos: votos, fecha, usuario, título y cuerpo del *post* y enlace a los comentarios del post.

En términos generales, el scrapeo se realizó de forma directa, ya que *BeautifulSoup* permite encontrar de manera sencilla los elementos de la clase deseada. No obstante, fue necesario realizar algunas tareas extra. Para **evitar guardar *posts* que eran anuncios**, hemos

decidido añadir unas líneas de código que nos permiten identificarlos y evitarlos, aunque como ya hemos comentado anteriormente, los hemos contado como *posts scrapeados.* Asimismo, Reddit proporciona las fechas en el formato tipo “hace 1 día”, por lo que fue necesario **crear una función para pasar las fechas a formato *datetime* de Python** (función *date_calculator*). Finalmente, para evitar la inclusión de *posts* con una antigüedad mayor al límite impuesto, hemos añadido líneas de código que terminan el *scraping* en caso de encontrar una fecha anterior al límite.

Una vez *scrapeados* los posts, hemos procedido a ***scrapear* los comentarios** correspondientes (función *scrape_comments).* En este caso, hemos hecho uso de los enlaces que hemos guardado en el *scrape* de los *posts* para ir **cargando de forma individual cada página de comentarios** en una pestaña secundaria del navegador (esto realmente no es necesario, pero lo consideramos interesante para practicar la **gestión de código de JavaScrip**t). La razón por la que hemos hecho esto es para evitar tener que hacer click directo en el enlace y hacer una vuelta posterior a la página principal del hilo, ya que consideramos que nuestra forma de proceder es más eficiente. Con la intención de evitar sobrecargas en el servidor y generar una óptima carga de las páginas, en este caso también hemos incluido una **demora de 2 segundos entre carga y carga**.

En relación a los datos *scrapeados,* en este caso hemos obtenido de forma similar que antes los siguientes campos: usuario, fecha, votos y cuerpo del comentario. En este caso, las únicas condiciones adicionales impuestas al *scraping* son la **desestimación de aquellos comentarios que no tienen usuarios (habitualmente son comentarios borrados), o aquellos generados por bots,** y un **límite de 20 comentarios *scrapeados* por página**.

Finalmente, con los datos recogidos de estos dos *scrapings*, hemos generado un dataframe conjunto, que posteriormente hemos exportado como archivo *csv.*

Como comentario final, nos gustaría mencionar que todos estos procesos se han incorporado como métodos a una clase que hemos llamado ***RedditScraper,*** que es heredada por la clase ***WSBetsScraper.*** La razón por la que hemos hecho esto es para tener un scraper base para todos los hilos de Reddit que pueda ser punto de partida para generar otras clases para los hilos particulares.

10. Dataset

Además de poder encontrarlo en el repositorio del proyecto en GitHub (carpeta */dataset*), también se ha publicado el dataset en formato CSV en Zenodo. El DOI del dataset es el siguiente: doi.org/10.5281/zenodo.7856609.

11. Vídeo

El enlace al vídeo de presentación de la práctica y el conjunto de datos se puede encontrar en el siguiente enlace:

https://drive.google.com/file/d/1MdHj8PMZNffcWAalXqcBDYDq9cZM-ly1/view?usp=share_link

12. Tabla de contribuciones

Contribuciones	**Firma**
Investigación previa	CDD, JOR
Redacción de las respuestas	CDD, JOR
Desarrollo del código	CDD, JOR
Participación en el vídeo	CDD, JOR