Práctica 2. Programación en bash

Disponible: 18 de septiembre. 13:00h

Entrega: 25 de septiembre. 23:55h

1 Introducción

Esta práctica va a consistir principalmente en hacer un varios scripts de bash para ver la potencia que tiene a la hora de desplegar modelos y realizar pipelines (procesamientos automatizados) en un ordenador. **En ningún momento de esta práctica se debe programar en Python ni se debe modificar ningún código Python proporcionado.**

2 Estructura de la entrega

Cada estudiante debe hacer entrega de un fichero .zip conteniendo los ficheros .sh (scripts en bash, un fichero para cada ejercicio) necesarios para la práctica. Estos ficheros deben replicar toda la funcionalidad requerida en los apartados. Si se quiere incluir algún fichero de texto adicional explicando la funcionalidad de los scripts, también está permitido. El fichero zip que se entregue debe seguir este formato (sin tildes, espacios ni eñes):

```
<nombreAlumno>_<Apellido1>_<Apellido2>_bash.zip
```

Por ejemplo:

```
Pablo_DelasHeras_Fernandez_bash.zip
```

IMPORTANTE: en esta práctica (como en todas), además de la funcionalidad, se tendrá en cuenta la **calidad del código entregado**. **Debido a que hay varios apartados, será obligatorio documentar el código. Es decir, habrá que comentar en cada ejercicio los pasos que se están realizando. Recuerda también que el fichero debe tener caracteres de fin de línea de LF, no de CRLF. Si desarrolláis en Visual Studio code, abajo a la derecha podéis ver si vuestro fichero está con CRLF o con LF y cambiarlo en caso de ser necesario.**

RECOMENDACIÓN: de cara a trabajar con la práctica, como nos estaremos descargando ficheros y procesándolos, recomiendo crear un directorio de trabajo llamado "PracticaBash" y realizar ahí todas las acciones de la práctica. Ese directorio no hay que entregarlo, solo hay que entregar el zip con los ficheros .sh requeridos. Será necesario trabajar con la máquina virtual o la aplicación Ubuntu.

3 Ejercicios de la práctica

Deberás descargarte el zip de moodle que contiene todos los datos de la práctica 2. Además, se recomienda encarecidamente tener abierto el tutorial de bash y los códigos de ejemplo de bash disponibles en Manuales de Instalación y uso de software, debajo de la etiqueta "Manuales con códigos, ejercicios, etc (comandos, bash, códigos Python)".

3.1 Apartado 1

Un uso muy habitual de los scripts de bash es ejecutar modelos de Machine Learning. En este sentido, es común disponer de un framework que permita ejecutar diferentes modelos con diferentes parámetros. Haciendo un script de bash es posible ejecutar esos modelos sin tener que ir uno a uno. Para hacer una prueba de esta funcionalidad, **se adjunta un código Python llamado ap1_codigo_python.py que trabaja con el fichero iris_csv.csv**. Si miráis el código veréis que a partir de los argumentos del programa, se puede ejecutar un modelo de Machine Learning y hacer una predicción. No tienes que entender el programa en su totalidad, pero si puedes abrir una terminal y ejecutar el programa Python de esta forma:

```
python3 ap1_codigo_python.py KNeighborsClassifier --neigh 10
```

Deberíais observar que se imprime los resultados del modelo KNeighborsClassifier (vecinos próximos) con 10 vecinos. Es posible que te pida instalar algún paquete, para ello, deberás ejecutar en la terminal lo siguiente:

```
sudo apt install python3-numpy
sudo apt install python3-pandas
sudo apt install python3-sklearn
```

NOTA: veremos en la práctica siguiente que esta no es una buena forma de instalar paquetes para Python. Para ello, en la práctica 3, emplearemos entornos virtuales.

Deberás completar el script de bash llamado ap1_alumnos.sh (.sh) para que realice las siguientes tareas:

- Imprimir la versión de Python del sistema (podéis obtenerlo con el comando python3 -V).
- **Debe ejecutar el programa Python empleando bucles, con estos tres modelos: KNeighborsClassifier, GaussianNB y RandomForestClassifier. En el caso del KNeighborsClassifier, se debe ejecutar con 10, 30 y 50 vecinos, aunque podrían ejecutarse con más, siempre en incrementos de 20 (argumento --neigh)**. Emplea bucles también para los vecinos. **Puedes basarte en el código que hay en moodle de ejecucion_python_mejorado.sh de ejemplos de códigos de bash**. El resto de modelos, no tienen parámetros. Cada resultado del modelo debe ser redirigido a un fichero llamado salida_ML.txt, de forma que al ejecutar el script, dicho fichero debería contener lo que se muestra en la caja inferior **(al iniciar el script, deberías eliminar dicho fichero de salida_ML.txt**, si ya existe por si ejecutas el script varias veces).

```
-----------------------------------
KNeighborsClassifier
Neighs 10
```

```
Accuracy: 1.0
-----------------------------------
-----------------------------------
KNeighborsClassifier
Neighs 30
Accuracy: 0.9666666666666667
-----------------------------------
-----------------------------------
KNeighborsClassifier
Neighs 50
Accuracy: 0.8666666666666667
-----------------------------------
-----------------------------------
GaussianNB
Accuracy: 0.9666666666666667
-----------------------------------
-----------------------------------
RandomForestClassifier
Accuracy: 0.9666666666666667
-----------------------------------
```

- Finalmente, cuando hayas ejecutado los tres modelos, el script de bash debe leer línea a línea el fichero de resultado e imprime aquellas líneas que contengan la palabra "Accuracy". **Puedes basarte en el código de bash de lectura_fichero y de condicionales.sh**.

  Entrega el script en un fichero que se llame "<nombre_completo_alumno>_ap1.sh". Recuerda aplicar las buenas prácticas de programación que ya conoces.

3.2 Apartado 2.

En este apartado, se pide completar el **fichero ap2_alumnos.sh que trabaja con el fichero ap2_codigo_python.py** que contiene algunas variables definidas. La primera, DOWNLOAD_PATH, contiene la url (dirección web) donde se encuentra un fichero de datos, HEADER, que contiene la cabecera (primera línea) de dicho fichero de datos, DATA_FILE, que es el nombre del fichero de datos y CSV_FILE, que contiene el nombre del fichero de datos con la cabecera.

Con esas variables, se pide crear un código bash que realice las siguientes tareas:

- Debe descargar el fichero estipulado en DOWNLOAD_PATH si y solo si no existe el fichero estipulado en la variable DATA_FILE (recuerda los comandos para descargar ficheros) y lo almacene en un fichero con el nombre estipulado en la variable DATA_FILE.

- Debe generar un fichero resultante de nombre estipulado en la variable CSV_FILE si y sólo si no existe. Ese fichero, debe contener como primera línea la cabecera (variable HEADER) y el resto de su contenido será el estipulado en DATA_FILE.

- Finalmente, empleando bucles, será necesario ejecutar el código Python (ap2_codigo_python.py) con los porcentajes de entrenamiento de 50, 60, 70, 80 y 90 (cada uno por separado) con el fichero con las cabeceras obtenido. La salida debe ser la siguiente:

```
El archivo wine.data ya existe. Omitiendo descarga.
El archivo wine_headers.csv ya existe. Omitiendo creacion de
     CSV con cabeceras.
Ejecutando el script de Python con 50% entrenamiento y 50%
    de test
Precision del modelo con 50% entrenamiento y 50% prueba:
    94.38%
Ejecutando el script de Python con 60% entrenamiento y 40%
    de test
Precision del modelo con 60% entrenamiento y 40% prueba:
    95.83%
Ejecutando el script de Python con 70% entrenamiento y 30%
    de test
Precision del modelo con 70% entrenamiento y 30% prueba:
    100.00%
Ejecutando el script de Python con 80% entrenamiento y 20%
    de test
Precision del modelo con 80% entrenamiento y 20% prueba:
    100.00%
Ejecutando el script de Python con 90% entrenamiento y 10%
    de test
Precision del modelo con 90% entrenamiento y 10% prueba:
    100.00%
```

Para este apartado, se recomienda consultar los ficheros bash de condicionales_fichero_directorio.sh y descarga.sh. Entrega el script en un fichero que se llame "<nombre_completo_alumno>_ap2.sh". Recuerda aplicar las buenas prácticas de programación que ya conoces.

3.3 Apartado 3.

En este apartado se pide trabajar con el fichero **ap3_codigo_python.py**, pensado para trabajar con el fichero de Movielens1M (un conjunto de datos que contiene 1 millón de interacciones entre usuarios y películas, disponible en este enlace). Dentro del zip que se indica en esa página, hay varios ficheros. El

que nos interesa, es el de ratings.dat, que contiene las notas que le han dado cada usuario a las diferentes películas. Se pide desarrollar un código bash que realice las siguientes tareas (de nuevo, puedes trabajar a partir del fichero de ap3_alumnos.sh):

- Descargarse el fichero zip de m1-1m.zip si no existe. Una vez descargado, es necesario que el script descomprima el .zip (puedes hacer uso del comando unzip <fichero.zip>).
- Se pide crear 4 directorios, llamados "headers", "descriptions", "filters" y "selections" (si ya existen, no se debe mostrar ningún error).
- Analiza el código de Python (pero no lo modifiques) y añade al script de bash la siguiente funcionalidad llamando al código de Python llamando a los argumentos correspondientes para realizar lo siguiente. Para cada una de las opciones habrá que indicar el fichero de origen y el de destino (es decir, es el propio programa Python el que genera el fichero de salida):
  - Con el argumento de show_head del código Python: genera en el directorio de headers 3 ficheros, uno con las primeras 100 líneas del fichero de ratings, otro con las primeras 200 líneas y otro con las 300 primeras líneas. Llámalos 100_lines.csv, 200_lines.csv y 300_lines.csv. Si te fijas, el formato de ejecutar el código Python en este caso, sería:

```
python3 ap3_codigo_python.py show_head <fichero_entrada>
     <fichero_salida> <tamanio>
```

  - Crea un fichero de descripción de datos del fichero de ratings.dat en el directorio de descriptions. Si te fijas, el formato de ejecutar el código Python en este caso, sería:

```
python3 ap3_codigo_python.py describe <fichero_entrada>
    <fichero_salida>
```

  - Filtra los datos de los ratings creando 2 ficheros en el directorio de filters, uno por cada uno de estos dos valores: 1 y 5. Los nombres de los ficheros creados deben ser m1_filter<numero>.csv de forma que cada fichero SOLO tenga ratings con una nota igual a dicho número. Explora el código Python viendo el argumento filter. El valor de "column" en ese caso, debe ser siempre Rating, y la columna value, cada uno de los valores que se piden.
  - Por cada uno de los valores de 10, 20 y 30, crea un fichero de nombre m1_selection<numero>.csv en el directorio de selections, que contenga el porcentaje de ratings indicado por cada uno de los números. Explora el código Python viendo el argumento select_percentaje.

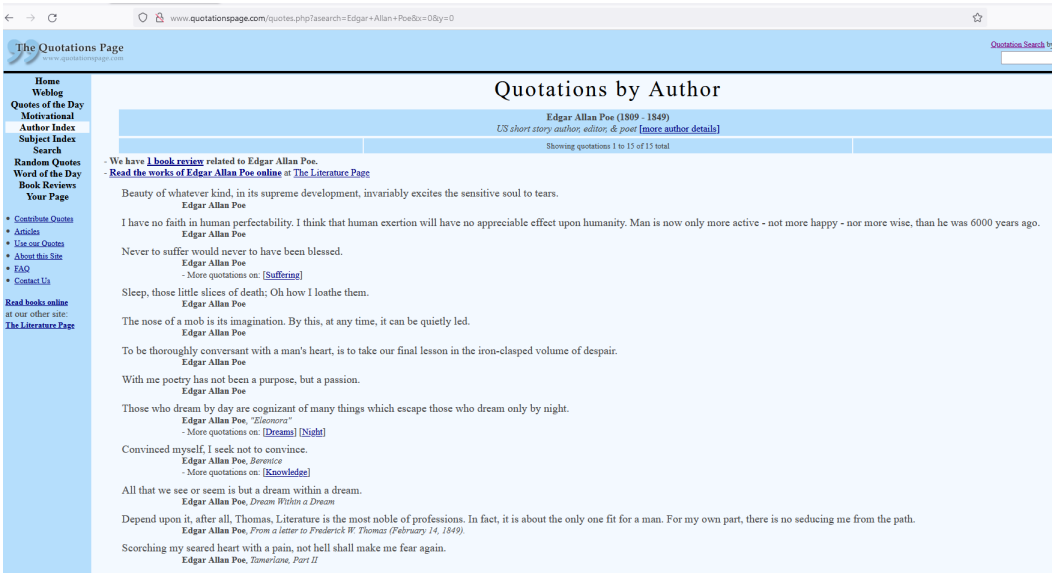

Figure 1: Ejemplo de búsqueda de las citas de Edgar Allan Poe

.

- Pon un sleep durante 5 segundos (escribe sleep 5) y posteriormente elimina los 4 directorios creados anteriormente junto con todo su contenido. **MUCHO CUIDADO CON LA ELIMINACIÓN DE ESTOS DIRECTORIOS. ASEGÚRATE DE NO ELIMINAR NADA MÁS.**
- **Entrega el script en un fichero que se llame "<nombre_completo_alumno>_ap3.sh". Recuerda aplicar las buenas prácticas de programación que ya conoces.**

3.4 Apartado Opcional

Para el apartado extra, vamos a crear un script que nos permita descargarnos datos de una página web de manera automática.

Para ello, vamos a trabajar con The Quotations Page, una página web que está en línea desde 1994 y que contiene 28.000 citas de más de 3.400 autores. El script que desarrollaremos en este apartado opcional nos va a permitir acceder a dicha página web y obtener las citas de los autores que queramos sin tener que hacerlo a mano. Como se puede ver, la URL de la página sería la siguiente: http://www.quotationspage.com/. Supongamos que queremos buscar citas del autor "Edgar Allan Poe". Para ello, podemos buscar por autor y en el buscador incluir dicho nombre, devolviéndonos la salida indicada en la Figura 1. Si os fijáis, la URL en la imagen ha cambiado, porque se le ha enviado los datos del autor en cuestión. Si por ejemplo la URL fuese

esta: http://www.quotationspage.com/quotes/Edgar_Allan_Poe, también estaríamos obteniendo esa misma salida.

El objetivo es, por tanto, dado el inicio de un script en bash, completarlo para producir la salida por terminal de las citas de varios autores.

3.4.1 Introducción básica html

Como en este apartado opcional vamos a descargarnos datos de la web, es importante saber cómo están construidos esos archivos (HTML). El lenguaje html es el empleado para "escribir" páginas web que luego nuestro navegador (Firefox, Chrome, Edge, etc) los leen y los muestran por pantalla. Los códigos html emplean unas instrucciones con marcas o etiquetas que siguen la siguiente sintaxis:

```
<id-etiqueta Atributos: informacion adicional>
```

Escribid, por ejemplo en un fichero (podéis visual studio code), el siguiente contenido (guardadlo como fichero .html):

```
<b> HOLA MUNDO </b>
```

Si lo abrís con un navegador, os debería mostrar el mensaje de hola mundo en negrita (dado que la etiqueta indica negrita). Si os fijáis, una vez que se abre una etiqueta, muchas veces es necesario "cerrarla" (se puede ver en este caso que hola mundo está entre , que es la apertura y que es el cierre). Todo documento html está delimitado por la etiqueta <html> y </html> y dentro de ese documento, se puede distinguir 2 partes fundamentales:

- El encabezado, delimitado por <head> y </head>, que es donde se colocan diversas etiquetas como puede ser el título de la página.
- El cuerpo, que se encuentra entre las etiquetas de <body> y </body> y que contiene el cuerpo de la página web.

Teniendo en cuenta lo anterior, esta podría ser una página web muy básica (copiad este contenido en un archivo llamado "primeraPag.html").

```
<html>
<head>
<title>Pagina web basica</title>
</head>
<body bgcolor="F3C1C1">
<p><b>Bienvenido a esta primera pagina,</b></p>
<p>Estas en una pagina de prueba</b>.</p>
<p>Y hay pocas cosas que tenemos que aprender.</p>
</body>
</html>
```

Si lo abrís con un navegador, mostrará un fondo rojo y un el texto de bienvenido ... y luego dos párrafos indicando que estamos en una página de prueba.

3.4.2 Preparativos

Cread un fichero .sh llamado **"<nombre_completo_alumno>_webscraping.sh"** que tenga el siguiente contenido (el archivo, deberás adjuntarlo en la entrega):

```
#!/bin/bash

# Nombre del/de la estudiante: <nombre> <apellidos>
```

Se deberá incluir en el fichero el nombre completo del/de la estudiante. Como primer paso, se os pide crear una variable web_page que contenga la URL básica de la aplicación (http://www.quotationspage.com/quotes/) y otra variable llamada "authors" que contenga el nombre de tres autores separados con _, es decir, "Edgar_Allan_Poe Agatha_Christie ..." y el último (los puntos suspensivos), debe ser un autor elegido por el estudiante. Fijaros que cada autor tiene el nombre pegado a los apellidos con el símbolo "_" debido a que cogiendo la cadena del nombre del autor separado por "_", podéis concatenarlo a la variable de web_page para acceder a las citas de cada uno de los autores, como se mostraba en la Sección 1. Es decir, http://www.quotationspage.com/quotes/Agatha_Christie os lleva a las citas que ha dicho Agatha Christie.

Para la realización del opcional, revisad muy bien y comprended los programas de ejemplo que hay publicados en el manual de Comandos de Linux (Sección 4, de Bash scripting). Recordad que bash no deja de ser un programa que nos va a permitir encadenar de manera secuencial un conjunto de comandos, por lo que deberéis de hacer uso de los diferentes comandos (descargar datos, filtrar, redirigir salidas de comandos, etc) todo ello en un único fichero (el script de bash). También será importante que aprendáis a construir y concatenar cadenas. Por ejemplo:

```
#!/bin/bash
variable=$(whoami)

echo "hola $variable" > $variable".txt"
```

Mediante ese script de bash, lo que estamos haciendo es meter en la variable llamada "variable" el resultado del comando de "whoami". Posteriormente, con echo, lo que hacemos es escribir "hola" y el resultado de la variable (esto gracias al símbolo de "$"). Si os fijáis, estamos redirigiendo después el resultado del comando echo al fichero de texto llamado <contenido_variable>.txt. Es decir, podemos crear un fichero cuyo nombre dependa del resultado de otro comando.

Se pide añadir a ese script la funcionalidad necesaria para:

- Crear un directorio llamado "quotes" si no existe (evitando que salga cualquier mensaje de error).
- Por cada uno de los autores, debe descargarse el html (la página web) correspondiente a sus citas (ya sabéis que comandos hay para descargar archivos de una web). El fichero de descarga debe guardarse en el directorio de "quotes" y seguir el formato de <nombre_autor>_unprocessed.txt.

Por ejemplo, para Edgar_Allan_Poe, sus citas deben almacenarse en Edgar_Allan_Poe_unprocessed.txt. Esta parte debe hacerse de la forma más genérica posible, de forma que si a la variable de authors se añadiese algún autor adicional, todo funcionase de manera correcta.

- Antes de proceder a descargar los datos, debe mostrarse el siguiente mensaje:

```
Descargando los datos de <nombre_autor>
```

Y después de la descarga, debe mostrarse lo siguiente:

```
Finalizada la descarga de los datos de <nombre_autor>
```

A continuación se muestra un ejemplo de salida:

```
Descargando los datos de Edgar Allan Poe
  % Total    % Received % Xferd  Average Speed   Time
    Time     Time  Current
                                 Dload  Upload   Total
    Spent    Left  Speed
100 22948    0 22948    0     0  44604      0 --:--:-- --:--
    :-- --:--:-- 44732
Finalizada la descarga de los datos de Edgar Allan Poe
Descargando los datos de Agatha Christie
  % Total    % Received % Xferd  Average Speed   Time
    Time     Time  Current
                                 Dload  Upload   Total
    Spent    Left  Speed
100 20548    0 20548    0     0  42159      0 --:--:-- --:--
    :-- --:--:-- 42193
Finalizada la descarga de los datos de Agatha Christie
[...]
```

- Fijaros que cuando se muestra el nombre del autor, se muestra con espacios. No obstante, cuando enviáis el nombre del autor en la URL, cada uno las componentes de su nombre deben estar con el símbolo de "_". Debéis de diseñar algún tipo de mecanismo para poder implementar esta funcionalidad.

Después del código para cumplir con la funcionalidad anterior, se pide añadir otro bloque de código que realice lo siguiente:

- Por cada fichero obtenido en el apartado anterior, filtrar únicamente las tags html haciendo referencia a las citas (quotes), que están contenidas en etiquetas <dt>, con el atributo class="quote". Recordad que las tags (etiquetas) html son aquellas "palabras" que están entre los símbolos de mayor y menor <>. El objetivo de este apartado es simplificar los archivos

descargados, ya que nos interesa el contenido de las citas, no el resto del html. Si por ejemplo Agatha Christie tiene 12 citas, el objetivo sería tener un archivo con 12 líneas siguiendo un formato similar a este (cada línea tendría un contenido parecido a este):

```
<dt class="quote"><a title="Click for further information
    about this quotation" href="/quote/39933.html">A mother's
     love for her child is like nothing else in the world.</a
    > </dt><dd class="author"><div class="icons"><a title="
    Further information about this quotation" href="/quote
    /39933.html"><img src="/icon_info.gif" width="16" height=
    "16" alt="[info]" border="0"></a><a title="Add to Your
    Quotations Page" href="/myquotations.php?add=39933"><img
    src="/icon_plus.gif" width="16" height="16" alt="[add]"
    border="0"></a><a title="Email this quotation" href="/
    quote/39933.html#email"><img src="/icon_email.gif" width=
    "16" height="16" alt="[mail]" border="0"></a><a title="
    Read notes about this quotation" href="/quote/39933.html#
    note"><img src="/icon_note.gif" width="16" height="16"
    alt="[note]" border="0"></a></div><b>Agatha Christie</b><
    /dd>
```

- El resultado de cada uno de estos procesamientos debe almacenarse en un fichero llamado <nombre_autor>_html_quotes.txt dentro del directorio de quotes (todas las citas de cada autor en su correspondiente fichero, es decir, si tenemos 3 autores distintos, escribiremos las citas de cada uno de ellos en un fichero independiente). Además, se pide también eliminar el fichero <nombre_autor>_unprocessed.txt obtenido en el apartado anterior.

Después del código para cumplir con la funcionalidad anterior, se pide añadir otro bloque de código que realice lo siguiente:

- Por cada autor (por cada fichero <nombre_autor>_html_quotes.txt), se pide eliminar todas las tags html, delimitadas por símbolos de mayor (>) y menor (<), y quedarse solo con el texto que no está entre esos símbolos. Por ejemplo, para el ejemplo mostrado de la línea anterior, el resultado sería el siguiente:

```
      A mother's love for her child is like nothing else in
           the world. Agatha Christie
```

- El resultado eliminando todos los tags (etiquetas) html de los ficheros <nombre_autor>_html_quotes.txt deben almacenarse en el fichero <nombre_autor>_processed_quotes.txt (de nuevo, en el directorio de quotes). Además, se pide también eliminar el fichero <nombre_autor>_html_quotes.txt obtenido en el apartado anterior. Para este apartado va a resultar útil el

uso de comandos que ya deberíais conocer para eliminar todos aquellos datos que no estén entre los símbolos < y >, que son los que delimitan las tags html.

En este punto, deberíamos tener por cada uno de los autores, un fichero llamado <nombre_autor>_processed_quotes.txt donde aparezcan cada una de las citas del autor correspondiente sin tags html. En este apartado se pide añadir un bloque de código que permita:

- Imprimir el nombre del autor, y luego imprima "Cita X" (donde X es el número de la cita). Por ejemplo, para Edgar Allan Poe, la salida sería la siguiente:

```
Citas de Edgar_Allan_Poe
Cita 1 Beauty of whatever kind, in its supreme development,
    invariably excites the sensitive soul to tears. Edgar
    Allan Poe
Cita 2 I have no faith in human perfectability. I think that
     human exertion will have no appreciable effect upon
    humanity. Man is now only more active - not more happy -
    nor more wise, than he was 6000 years ago. Edgar Allan
    Poe
Cita 3 Never to suffer would never to have been blessed.
    Edgar Allan Poe - More quotations on: [Suffering]
Cita 4 Sleep, those little slices of death; Oh how I loathe
    them. Edgar Allan Poe
[...]
```

  Inmediatamente después de terminar las citas de Edgar Allan Poe, se escribirían las citas del siguiente autor y así sucesivamente.

Añadid la funcionalidad requerida en los Apartados 1-3 para que no se ejecute nada si ya existe el fichero final de _quotes.txt para el autor en cuestión. Es decir, si ya existe el fichero procesado, no debe descargarse ni procesarse ningún fichero adicional para ese autor. Si para algún autor no está el fichero de final _quotes.txt, sí se debería proceder a descargar el fichero como antes. Para el código del apartado 1, además, en caso de que ya existan los datos, se debe mostrar el siguiente mensaje:

```
Ya existen los datos asociados a <nombre_autor_cuestion>. No se
    procede a la descarga
```