Tipología y Ciclo de Vida de los Datos

Práctica 1

Nombre y apellidos componentes del grupo

Luís García Tarraga (lgarciatar@uoc.edu) - LGT

Marco Emilio Rodríguez Serrano (mrodrise@uoc.edu) - MERS

Respuestas

1. Contexto. Explicar en qué contexto se ha recolectado la información. Explique por qué el sitio web elegido proporciona dicha información.

Hemos decidido usar para la práctica el sitio web https://www.techpowerup.com/cpu-specs

Se trata de un sitio web donde se pueden consultar los distintos CPUs que se han lanzado al mercado con todas sus especificaciones desde el año 2000.

Se recopila distinta información en detalle como el número de transistores, tipo de arquitectura, caches, etc. Además, incorpora un texto informativo sobre el origen del procesador y su funcionamiento.

Hemos considerado que recopilar esta información puede ser útil para:

- Hacer predicciones mediante machine learning sobre las características de los siguientes procesadores que se lanzarán al mercado.
- Hacer gráficas sobre la evolución histórica de los procesadores que sean visuales y que permitan analizar de forma más visual los datos y así sacar conclusiones.
- La información se podría utilizar para entrenar alguna herramienta cognitiva como IBM Watson Discovery, lo que permitiría disponer de un corpus de conocimiento que podría ser explotado por un chatbot cara a darle a los usuarios información referente a procesadores.
- Se podría utilizar para implantar un sistema que ayude a los usuarios en la compra de un procesador.

El sitio web elegido dispone de la información que nos permitiría llevar a cabo estos proyectos.

2. Definir un título para el dataset. Elegir un título que sea descriptivo.

Hemos decidido utilizar como título: Especificaciones Técnicas de CPUs desde el Año 2000.

3. Descripción del dataset. Desarrollar una descripción breve del conjunto de datos que se ha extraído (es necesario que esta descripción tenga sentido con el título elegido).

En el dataset se incluye información de diferentes CPUs desde el año 2000.

Se incluye información técnica a mucho nivel de detalle que comprende desde las cachés que pueda tener, el voltaje de la CPU, la memoria utilizada hasta de información sobre qué tecnologías incluyen esa CPU.

Esta información puede utilizarse para estudiar la evolución de las CPU's, tanto de Intel como de AMD que son los dos principales productores internacionales de este producto. En plataformas como Kaggle donde los usuarios pueden compartir sus datasets y los análisis que realizan sobre ellos encontramos, por ejemplo, este análisis: https://www.kaggle.com/trion129/intel-cpus-eda en el que se analiza la evolución de las CPU de Intel, este mismo análisis se podría actualizar con este dataset para añadir una comparativa con las CPU de AMD.

Vemos por tanto que podríamos utilizar los datos que obtenemos de esta web para preparar un análisis histórico, y generar un modelo que nos permita predecir cómo serán las CPU futuras. Además, otro estudio muy interesante a realizar con estos datos seria comprender la relación entre los diferentes atributos de las CPU, por ejemplo, entender en las diferentes arquitecturas cual es la relación entre consumo y frecuencia, entre tamaño del die y capacidad de cache, etc. Esta información puede ser muy valiosa para diseñar futuras arquitecturas.

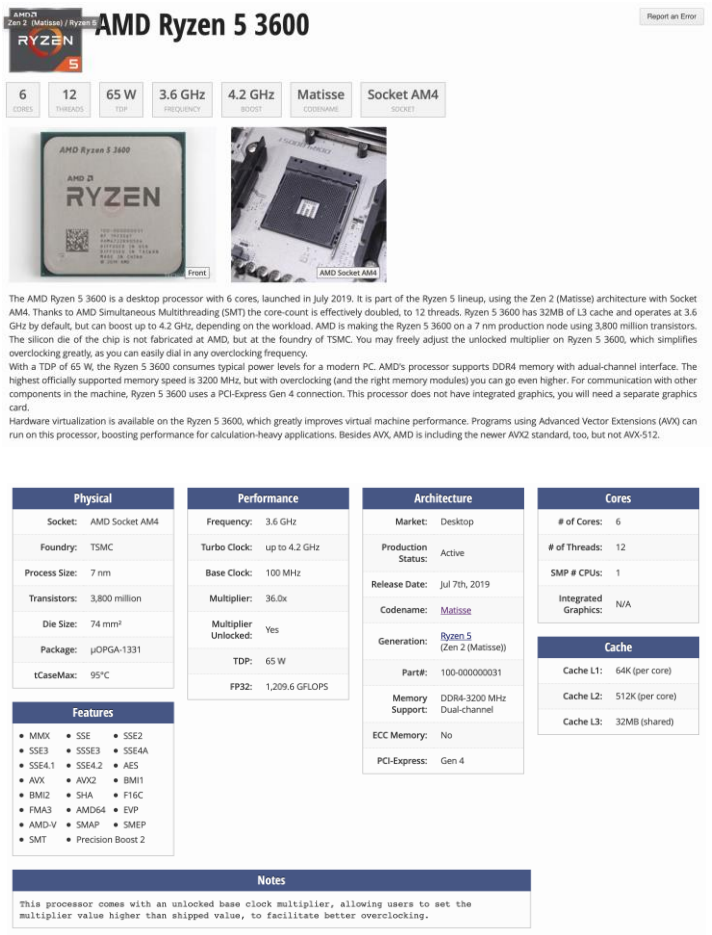
A modo de conclusión, podemos ver que este dataset puede utilizarse para:

- Analizar la evolución de las CPU's.
- Comprender la relación entre los diferentes aspectos de una CPU.
- Entrenar modelos que puedan generar CPU inexistentes o identificar CPU con el mínimo posible de información.

4. Representación gráfica. Presentar una imagen o esquema que identifique el dataset visualmente

A continuación, se muestra una lista de distintas CPUs y una página con el detalle de una de ellas.

Name	Codename	Cores	Clock	Socket	Process	L3 Cache	TDP	Released
Ryzen 5 3600	Matisse	6 / 12	3.6 to 4.2 GHz	Socket AM4	7 nm	32MB	65 W	Jul 7th, 2019
Ryzen 3 3100	Matisse	4 / 8	3.6 to 3.9 GHz	Socket AM4	7 nm	16MB	65 W	Apr 24th, 2020
Ryzen 7 3700X	Matisse	8 / 16	3.6 to 4.4 GHz	Socket AM4	7 nm	32MB	65 W	Jul 7th, 2019
A8-7680	Godaveri	4	3.5 to 3.8 GHz	Socket FM2+	28 nm	N/A	45 W	Oct 26th, 2018
Core i7-10700K	Comet Lake	8 / 16	3.8 to 5.1 GHz	Socket 1200	14 nm	16MB	125 W	Apr 30th, 2020
Ryzen 5 3500X	Matisse	6	3.6 to 4.1 GHz	Socket AM4	7 nm	32MB	65 W	Sep 24th, 2019
Core i9-10900K	Comet Lake	10 / 20	3.7 to 5.3 GHz	Socket 1200	14 nm	20MB	125 W	Apr 30th, 2020
Ryzen 9 3900X	Matisse	12 / 24	3.8 to 4.6 GHz	Socket AM4	7 nm	64MB	105 W	Jul 7th, 2019
Core 2 Duo E8400	Wolfdale	2	3 GHz	Socket 775	45 nm	N/A	65 W	Jan 1st, 2008



AMD Ryzen 5 3600

Report an Error

6	12	65 W	3.6 GHz	4.2 GHz	Matisse	Socket AM4
CORES	THREADS	TDP	FREQUENCY	BOOST	CODENAME	SOCKET

The AMD Ryzen 5 3600 is a desktop processor with 6 cores, launched in July 2019. It is part of the Ryzen 5 lineup, using the Zen 2 (Matisse) architecture with Socket AM4. Thanks to AMD Simultaneous Multithreading (SMT) the core-count is effectively doubled, to 12 threads. Ryzen 5 3600 has 32MB of L3 cache and operates at 3.6 GHz by default, but can boost up to 4.2 GHz, depending on the workload. AMD is making the Ryzen 5 3600 on a 7 nm production node using 3,800 million transistors. The silicon die of the chip is not fabricated at AMD, but at the foundry of TSMC. You may freely adjust the unlocked multiplier on Ryzen 5 3600, which simplifies overclocking greatly, as you can easily dial in any overclocking frequency.
With a TDP of 65 W, the Ryzen 5 3600 consumes typical power levels for a modern PC. AMD's processor supports DDR4 memory with adual-channel interface. The highest officially supported memory speed is 3200 MHz, but with overclocking (and the right memory modules) you can go even higher. For communication with other components in the machine, Ryzen 5 3600 uses a PCI-Express Gen 4 connection. This processor does not have integrated graphics, you will need a separate graphics card.
Hardware virtualization is available on the Ryzen 5 3600, which greatly improves virtual machine performance. Programs using Advanced Vector Extensions (AVX) can run on this processor, boosting performance for calculation-heavy applications. Besides AVX, AMD is including the newer AVX2 standard, too, but not AVX-512.

Physical

Socket:	AMD Socket AM4
Foundry:	TSMC
Process Size:	7 nm
Transistors:	3,800 million
Die Size:	74 mm²
Package:	µOPGA-1331
tCaseMax:	95°C

Performance

Frequency:	3.6 GHz
Turbo Clock:	up to 4.2 GHz
Base Clock:	100 MHz
Multiplier:	36.0x
Multiplier Unlocked:	Yes
TDP:	65 W
FP32:	1,209.6 GFLOPS

Architecture

Market:	Desktop
Production Status:	Active
Release Date:	Jul 7th, 2019
Codename:	Matisse
Generation:	Ryzen 5 (Zen 2 (Matisse))
Part#:	100-000000031
Memory Support:	DDR4-3200 MHz Dual-channel
ECC Memory:	No
PCI-Express:	Gen 4

Cores

# of Cores:	6
# of Threads:	12
SMP # CPUs:	1
Integrated Graphics:	N/A

Cache

Cache L1:	64K (per core)
Cache L2:	512K (per core)
Cache L3:	32MB (shared)

Features

- MMX
- SSE
- SSE2
- SSE3
- SSSE3
- SSE4A
- SSE4.1
- SSE4.2
- AES
- AVX
- AVX2
- BMI1
- BMI2
- SHA
- F16C
- FMA3
- AMD64
- EVP
- AMD-V
- SMAP
- SMEP
- SMT
- Precision Boost 2

Notes

This processor comes with an unlocked base clock multiplier, allowing users to set the multiplier value higher than shipped value, to facilitate better overclocking.

5. Contenido. Explicar los campos que incluye el dataset, el periodo de tiempo de los datos y cómo se ha recogido.

A continuación, se describen los distintos campos que componen el dataset:

Campo	Descripción	Ejemplo	Tipo
cpu_name	Nombre de la CPU	AMD Ryzen 5 3600	String
cpu_logo	Link a la imagen con el logo de la CPU	(configurable por el usuario)	String or PNG (Image binary)
socket	Tipo de socket que necesita la CPU	AMD Socket AM4	String
foundry	Fabricante	TSMC	String
process_size	Tecnología de fabricación	7nm	String
transistors	Número de transistores	3,800 million	String
die_size	Tamaño del die (la pastilla)	74 mm²	String
package	Nombre técnico del socket de la CPU	µOPGA-1331	String
tcasemax	Temperatura máxima del encapsulado	95°C	String
frequency	Frecuencia de la CPU	3.6 GHz	String
turbo_clock	Frecuencia máxima	up to 4.2 GHz	String
base_clock	Frecuencia placa base	100 MHz	String
multiplier	Multiplicador	36.0x	String
multiplier_unlocked	Multiplicador desbloqueado	Yes	String
voltage	Voltaje de la CPU	-	String
tdp	Vatios que va a consumir	65 W	String
fp32	Número de unidades de la CPU	1,209.6 GFLOPS	String
market	Tipo de mercado	Desktop	String
production_status	Estado de la producción en la actualidad	Active	String
release_date	Fecha de lanzamiento	Jul 7th, 2019	String
codename	Nombre código del proyecto	Matisse	String
generation	Generación	Ryzen 5 (Zen 2 (Matisse))	String
part_number	Código de procesador	100-000000031	String
memory_support	Memoria soportada	DDR4-3200 MHz Dual-channel	String
ecc_memory	Memoria ECC	No	String
pci_express	Generación PCI Express	Gen 4	String
cores_num	Número de cores	6	Integer
threads_num	Número de threads	12	Integer
smp_cpus_num	Multiprocesadores simétricos	1	Integer

integrated_graphics	Tarjeta gráfica integrada	N/A	String
cache_l1	Caché nivel 1	64K (per core)	String
cache_l2	Caché nivel 2	512K (per core)	String
cache_l3	Caché nivel 3	32MB (shared)	String
features_list	Listado de características del procesador	MMX, SSE, SSE2, SSE3, SSSE3, SSE4A, SSE4.1, SSE4.2, AES, AVX, AVX2, BMI1, BMI2, SHA, F16C, FMA3, AMD64, EVP, AMD-V, SMAP, SMEP, SMT, Precision Boost 2	String
imgs	Link a las imágenes del procesador	(configurable por el usuario)	String or Multiple PNG (Image binary)
notes	Notas sobre el procesador	This processor comes with an unlocked base clock multiplier, allowing users to set the multiplier value higher than shipped value, to facilitate better overclocking.	String

El dataset comprende el periodo que va desde el año 2000 hasta la actualidad.

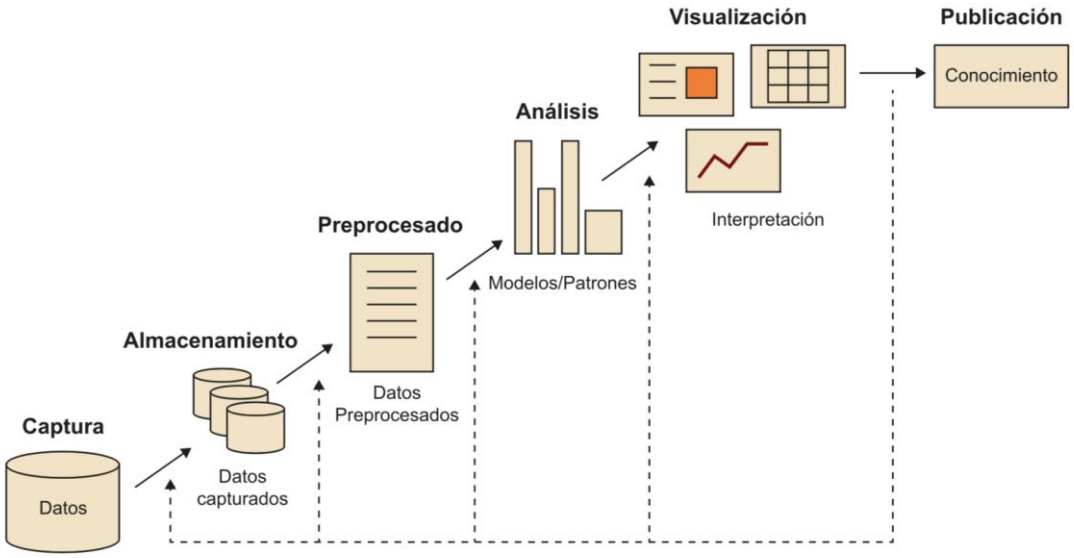
En la página web se permite filtrar por distintos parámetros, entre ellos el año. Indicar que al seleccionar el año tan solo muestra un total de 101 resultados y no da opción a navegar a otras páginas. No se puede filtrar por año y mes, por lo que no se puede extraer la información de esa forma. Se trata de un error de la página web.

Para llevar a cabo la recopilación de la información, se ha procedido del siguiente modo:

1. Se ha cargado la página inicial con la información.
2. Se ha extraído de la lista despegable de fechas todas las fechas disponibles.
3. Se ha creado un bucle para recorrer todas las fechas:
    a. Se ha navegado al link de la página web con la fecha concreta filtrada.
    b. Se han extraído los distintos campos utilizando el tag, clase e id de los campos según conveniencia
    c. Las imágenes se guardan en local y en el dataset se indica un link a dónde se guarda la imagen

Dentro del proyecto se ha utilizado Selenium para cargar las páginas web, también se ha utilizado la librería urllib para probar las distintas opciones.

Podemos recordar que las etapas típicas del ciclo de vida de los datos comprenden la captura, almacenamiento y preprocesado de datos entre otras etapas.



En este caso el web scraping correspondería a la primera etapa de captura de los datos, en la que hacemos un almacenamiento de los mismos, en nuestro caso en forma de fichero csv e imágenes en carpetas en caso de que el usuario así lo indique al ejecutar el programa que hemos desarrollado.

Es posteriormente al proceso de web scraping donde se haría el preprocesado de los datos, por lo que no hemos incorporado esta etapa en nuestro desarrollo. Un motivo es que el tipo de preprocesado a realizar dependerá del análisis que se vaya a hacer. Existen distintas acciones que se pueden llevar a cabo en la etapa de preprocesado: integración, selección, reducción de datos, conversión y limpieza.

En nuestro proceso de web scraping, nos hemos quedado en el punto de almacenamiento de los datos, pero a un nivel básico de fichero. Sería en la siguiente etapa en la que esos datos se podrían llegar a incluir en una base de datos e incluso enriquecerlos con datos de otras fuentes.

También se han considerado las buenas prácticas a la hora de hacer web scraping:

- Se ha revisado si existía una API para descargar la información, pero no existía API alguna.
- No se ha parseado el html manualmente, se ha utilizado BeautifulSoup para ello.
- Hemos procurado no bloquear de peticiones el servidor, poniendo tiempos de espera. Además, hemos intentado contactar con el administrador del sitio, aunque no hemos tenido respuesta.
- Hemos modificado el user agent, hemos probado a quitar la cabecera y en otras extracciones hemos puesto una cabecera propia.
- Cuando hemos tenido problemas, hemos hecho un check utilizando el navegador en incógnito, también lo hemos hecho en modo normal, lo que nos ha permitido ver que se estaba solicitando indicar que no éramos un robot.
- Se ha asumido que el web scraper puede dejar de funcionar, introduciendo advertencias. Al recoger los datos de la CPU, guardamos en un fichero aparte aquellas propiedades que no tengamos contempladas en el código para que se puedan tratar posteriormente.

- Al extraer los datos hemos tenido en cuenta su calidad y robustez, hemos revisado que cumplan las siguientes dimensiones:
    - **Completitud** – hemos visto que al extraer la información se corta a las 101 filas, por lo que no sería completo. Se debe a un bug de la página web.
    - **Unicidad** – cada CPU es única, hemos revisado que en el dataset no se repitan CPUs.
    - **Puntualidad** – cada CPU viene acompañada de la fecha de creación.
    - **Validez** – los datos se ajustan al formato, tipo y rango.
    - **Exactitud** – los datos representan las diferentes especificaciones de CPUs a un nivel de detalle muy alto.
    - **Consistencia** – la información es consistente, aunque en algunos casos no se disponen de todos los campos rellenados.
- Hemos revisado los aspectos legales al hacer web scraping aunque no hemos visto licencias asociadas a los datos, se ha escrito al ad.ministrador para saber qué tipo de licencia usan pero no se ha recibido respuesta
- Se ha tenido en cuenta el fichero robots.txt. Se ha utilizado la librería robotparser para "parsear" este fichero y ver si se puede acceder al contenido. En caso de que no se pueda, el programa finaliza avisando al usuario.

6. Agradecimientos. Presentar al propietario del conjunto de datos. Es necesario incluir citas de investigación o análisis anteriores (si los hay).

La información pertenece a la empresa TechPowerUp, que es una empresa orientada a aquellas personas entusiastas de los ordenadores, enfocándose al hardware y la industria que le rodea.

Son un sitio de noticias, no se dedican a la compra y venta de ordenadores, la organización está distribuida entre distintas geografías.

Para conocer más sobre sus colaboradores, consultar la página:
https://www.techpowerup.com/contact/

Comentar que no hemos encontrado investigaciones o análisis anteriores, aunque nos hemos basado en las siguientes referencias para implementar el código:

Muthukadan, J. (2018). Selenium with Python. Disponible en:
https://selenium-python.readthedocs.io

Singhal, G. (2020). Advanced Web Scraping Tactics. Disponible en:
https://www.pluralsight.com/guides/advanced-web-scraping-tactics-python-playbook

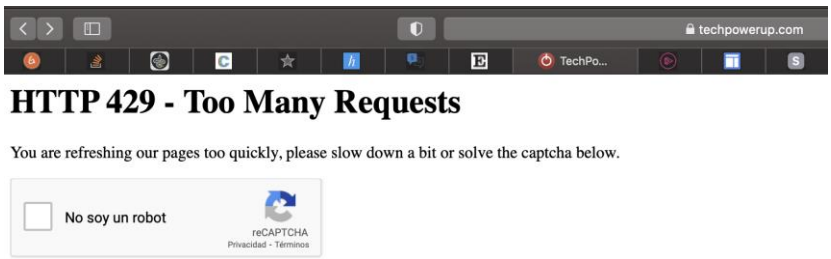
FindDataLab. The Ultimate Guide To Ethical Web Scraping. Disponible en:
https://finddatalab.com/ethicalscraping#Time_outs_and_responsive_delays

7. Inspiración. Explique por qué es interesante este conjunto de datos y qué preguntas se pretenden responder.

Una de las motivaciones principales es que los dos colaboradores en esta práctica compartimos una afición por el hardware. A la hora de seleccionar una página web para extraer información estuvimos revisando páginas de cursos de tecnología, páginas de resultados de deportes; finalmente, decidimos optar por esta página, ya que se trata de una afición compartida.

Por otro lado, al hacer web scraping sobre la página web vimos la posibilidad de conseguir las siguientes metas:

- Recorrer una lista desplegable con fechas para cargar tablas de CPUs por año.
- Navegar a una página diferente dentro de cada registro de CPU donde se podía conseguir más detalle y además extraer imágenes.
- Una vez comenzamos a explotar la información encontramos que el webmaster había introducido una medida de seguridad que bloqueaba las IPs que hacían llamadas sin tiempos de espera contra el site. Para evitar el ser detectados tuvimos que introducir tiempos de espera proporcionales al tiempo de respuesta del servidor. A modo de curiosidad, nos encontramos que pedía indicar que no eras un robot a nivel de página web cuando recibía muchas peticiones seguidas. Además, a nivel de línea de comando daba un error rechazando conexiones.




```
➜  dev python3 webscraping_cpus.py
Traceback (most recent call last):
  File "webscraping_cpus.py", line 52, in <module>
    driver.get(f"https://www.techpowerup.com/cpu-specs/?sort=name")
  File "/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/selenium/webdriver/remote/webdriver.py", line
333, in get
    self.execute(Command.GET, {'url': url})
  File "/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/selenium/webdriver/remote/webdriver.py", line
321, in execute
    self.error_handler.check_response(response)
  File "/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/selenium/webdriver/remote/errorhandler.py", li
ne 242, in check_response
    raise exception_class(message, screen, stacktrace)
selenium.common.exceptions.WebDriverException: Message: unknown error: net::ERR_CONNECTION_REFUSED
  (Session info: headless chrome=86.0.4240.80)
```

Además de lo interesante que resulta esta página desde el punto de vista práctico, nos parece que como hemos explicado en la descripción del dataset estos datos pueden dar lugar a estudios evolutivos muy interesantes, además de ayudar a generar modelos que intenten explicar la relación entre las diferentes características de las diferentes CPU, lo que puede ayudar a diseñar mejor futuras CPU o chips nuevos a medida que se requieran para productos específicos. Aunque no hemos encontrado estudios específicos con estos objetivos sí que hemos encontrado estudios como el de

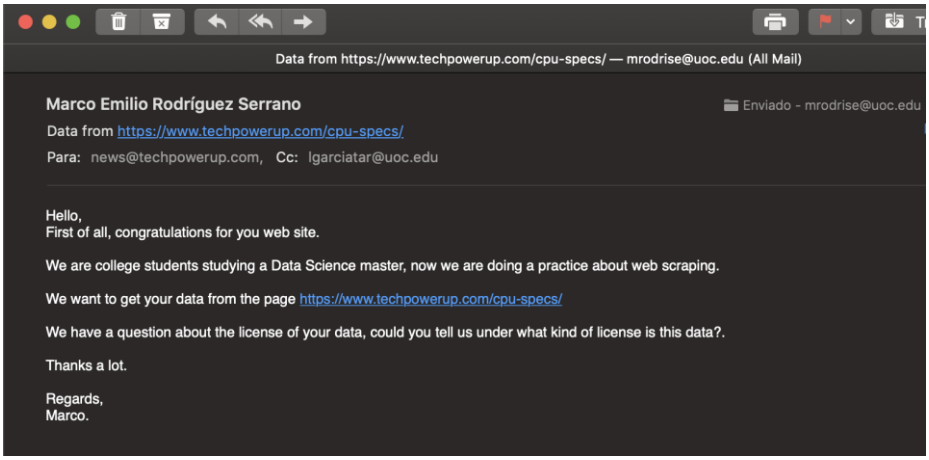
la evolución de las CPU de Intel que se encuentra en la descripción del dataset y este otro: https://www.kaggle.com/skalskip/using-regression-to-predict-gpus-of-the-future en el que se aplica regresión para intentar predecir la GPU del futuro y que sería extrapolable a nuestro dataset de CPU's ya que son chips con características muy similares.

8. Licencia. Seleccione una de estas licencias para su dataset y explique el motivo de su selección:

- Released Under CC0: Public Domain License
- Released Under CC BY-NC-SA 4.0 License
- Released Under CC BY-SA 4.0 License
- Database released under Open Database License, individual contents under Database Contents License
- Other (specified above)
- Unknown License

Hemos revisado la página web de la compañía y no hemos encontrado información en cuanto al tipo de licencia bajo la que se encuentra esta información.

Hemos remitido un email con la pregunta a la empresa, pero no hemos obtenido todavía respuesta:



Vamos a analizar a continuación cuál de las licencias del ejercicio sería la que mejor se ajusta a nuestro dataset:

Tipo de licencia	Análisis
Released Under CC0: Public Domain License	En este caso la información es de dominio público, por lo que cualquier persona la puede utilizar
Released Under CC BY-NC-SA 4.0 License	En este caso se puede utilizar, pero haciendo referencia al creador de la información e indicando si se han hecho cambios. No está permitido para uso comercial
Released Under CC BY-SA 4.0 License	Sería parecido al caso anterior pero no se restringe su uso comercial
Database released under Open Database License,	Sería como una doble licencia: - Open Database License

individual contents under Database Contents License	- Database Contents License para contribuciones individuales Esta licencia permite a los usuarios compartir, modificar y usar libremente una base de datos, manteniendo la misma libertad para los demás
Other (specified above)	No tenemos información de la empresa como para saber si utiliza otra licencia
Unknown License	Sería nuestro caso, licencia desconocida ya que no sabemos qué tipo de licencia tiene la fuente original

En conclusión, como no sabemos la licencia de la página original, no podemos determinar una licencia para este dataset. Aunque al no especificarse la licencia vamos a utilizar una licencia CC BY-NC-SA 4.0.

9. Código. Adjuntar el código con el que se ha generado el dataset, preferiblemente en Python o, alternativamente, en R.

Código fuente incluido en repositorio donde se encuentra este documento.

Main.py

```
from bs4 import BeautifulSoup
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.support.ui import WebDriverWait
from selenium.common.exceptions import TimeoutException
import time
import platform
import sys
# importamos robotparser que tiene utilidades para leer el fichero robots.txt
from urllib import robotparser

# Definimos en otro fichero la clase CpuData que a partir de una URL de una CPU,
# extraiga todos sus datos e imágenes, se utilizará más adelante en el bucle principal
de extracción
# de los datos de las distintas CPUs.
from src.CpuData import CpuData


def driver_config():

    # En caso de que se quieran añadir opciones se puede hacer aquí
    chrome_options = Options()

    # Una opción para evitar que no nos dejen hacer webscraping es no usar cabeceras
    chrome_options.add_argument("--headless")

    # Antes de iniciar el webdriver necesitamos saber si el sistema operativo es Linux,
```

```
Mac o Windows
    # El path del ejecutable cambia para windows con respecto a Linux y Mac ya que
necesita extensión .exe
    os = platform.system()
    print("Detected OS:", os)

    # Iniciamos en webdriver dependiendo del sistema operativo
    if os == "Windows":
        inner_driver = webdriver.Chrome(executable_path="./chromedriver.exe",
options=chrome_options)
    else:
        inner_driver = webdriver.Chrome(executable_path="./chromedriver",
options=chrome_options)

    return inner_driver


def get_info_cpus(driver_inner, year_inner):
    # La página web detecta web scraping, bloqueando si detecta que se hacen muchas
peticiones seguidas
    # Vamos a introducir un delay para evitar que nos bloqueen
    t0 = time.time()
    # Cargamos la página web para el año que nos han pasado
    driver_inner.get(f"https://www.techpowerup.com/cpu-
specs/?released={year_inner}&sort=name")
    # Esperamos a que cargue la página hasta el elemento que necesitamos
    try:
        WebDriverWait(driver_inner, 5).until(lambda s:
s.find_element_by_class_name("processors").is_displayed())
    except TimeoutException:
        print("ERROR - TimeoutException: Webpage can't be loaded.")
        return None

    # Vemos cuánto es el delay entre que lanzamos la petición y tenemos el resultado
    response_delay = time.time() - t0

    # Utilizamos BeautifulSoup para tratar con más facilidad la informaicón cargada
    soup_inner = BeautifulSoup(driver.page_source, "lxml")

    # Utilizamos la variable j para tener un contador a modo de resumen de número de
procesadores para ese año
    j = 0

    # Buscamos la tabla processors donde está el listado de procesadores con sus links a
las páginas de detalle
    for table in soup_inner.select("table.processors"):
        for detail_link in table.select("a"):
            print(detail_link['href'])

            # Creamos un objeto de tipo CpuData y lo inicializamos con la URL de la que
extraer la información
            cpu = CpuData('https://www.techpowerup.com'+detail_link['href'])

            # Almacenamos los datos que hemos extraido de esa url
```

```
            cpu.store_data(dataset_path, include_headers, create_images, img_as_path)

            j += 1
            # Tenemos que poner tiempos de espera ya que el servidor detecta acciones
muy seguidas desde una misma IP
            # Estos tiempos de espera se han establecido por prueba y error, en base a
las distintas ejecuciones realiadas
            time.sleep(2*60)

            # Cada 100 extracciones debemos meter un tiempo de espera prudencial para
que el servidor no nos bloquee
            if j % 100 == 0:
                time.sleep(5*60)

    print("A total of: ", j, " links where found.")

    # Procedemos a meter un tiempo de espera prudencial en base al tiempo de respuesta
del servidor.
    # Así evitamos errores de tipo 423
    time.sleep(10 * response_delay)

# ************************************************
# ************** PARTE MAIN DEL CÓDIGO **************
# ************************************************

if __name__ == '__main__':

    # Antes de comenzar necesitamos saber si el administrador del sitio da permiso para
bajar los datos
    # Utilizamos un parser que nos ayude a leer el fichero robots.txt
    rp = robotparser.RobotFileParser()

    # Establecemos como url donde se encuentra el fichero robots.txt
    rp.set_url("https://www.techpowerup.com/robots.txt")

    # Leemos el fichero robots.txt
    rp.read()

    # Utilizando el parser vemos si para cualquier tipo de agente está permitido bajarse
los datos
    # En el momento en que se hizo este código se podía bajar la página que nos interesa
para
    # cualquier tipo de agente menos para SentiBot
    if (rp.can_fetch("*", "https://www.techpowerup.com/cpu-specs/?sort=name") == False):
        # Si el fichero robots.txt indica que no podemos bajar esa página, entonces
finalizamos el programa
        print("The administrator of the site doesn't allow to download this data, the
program will finish")
        sys.exit()

    # Creamos el objeto de driver y lo inicializamos
    driver = driver_config()

    # Accedemos a la página principal donde está la información que queremos extraer
```

```
    driver.get(f"https://www.techpowerup.com/cpu-specs/?sort=name")

    # Esperamos a que cargue el elemento que necesitamos, en este caso es la tabla
processors que contine
    # la información de procesadores que necesitamos extraer

    try:
        WebDriverWait(driver, 5).until(lambda s:
s.find_element_by_class_name("processors").is_displayed())
    except TimeoutException:
        print("ERROR - TimeoutException: Webpage can't be loaded.")

    # Utilizamos BeautifulSoup con el parser que parece más extendido: lxml
    soup = BeautifulSoup(driver.page_source, "lxml")

    # Buscamos el elemento que contiend el id "released"
    # El objetivo es leer de este selector todos los años para los que la página web
tiene información
    course_page = soup.find(id='released')

    # Cargamos todos los años, buscando el tag option dentro del selector
    # Ignoramos el primer valor de option ya que es para cargar los resultados más
populares y queremos años
    years = []
    i = 0
    for year_selector in course_page.find_all("option"):
        if i > 0:
            # print(year_selector['value'])
            years.append(year_selector['value'])
        i = i + 1

    print("Years ", years)

    # Preguntamos al usuario en qué carpeta quiere guardar la información.
    # Si pulsa D o pulsa enter, entonces creará el dataset en el directorio actual

    dataset_path = input("Default path (\"D\") or custom:")
    if dataset_path == "D" or len(dataset_path) < 2:
        dataset_path = 'dataset.csv'

    # Se le pregunta al usuario si quiere que las imágenes las guardemos en una carpeta
o si prefiere
    # que las guardemos en un campo de dataset en formato bs4.
    img_as_path = input("Img as path in dataset (\"Y\") or as BS4 (\"N\") (BS4 could
lead to broken lines in dataset):")
    if img_as_path == 'N':
        img_as_path = False
    else:
        img_as_path = True

    # Preguntamos al usuario si quiere que generemos las imágenes o no
    create_images = input("Generate images (\"Y\"/\"N\"):")
    if create_images == "N":
        create_images = False
```

```
    else:
        create_images = True

    # Preguntamos al usuario si quiere que se incluyan headers
    include_headers = input("Include headers (\"Y\"/\"N\"):")
    if include_headers == "N":
        include_headers = False
    else:
        include_headers = True

    # Preguntamos al usuario qué años quiere que extraigamos de la página web
    # Si pulsa enter, se bajará todos los años
    selected_years = input("As default dataset will contain all the loaded years, if you
prefer to pick just some "
                           "years write them separated by coma for example
(2010,2011,2012):")

    # selected_years es un string, miramos que el usuario al menos haya introducido 2
caracteres
    if len(selected_years) > 2:
        selected_years_list = selected_years.split(',')
        selected_years_list_clean = []
        # Revisamos que los años introducidos estén dentro del rango de años de la lista
de años de la página web
        for year in selected_years_list:
            if year in years and year not in selected_years_list_clean:
                selected_years_list_clean.append(year)
            else:
                print('Year '+year+' has been removed, incorrect input or not
available.')
        # Si se comprueba después de hacer limpieza que no hay años válidos, se cargan
todos
        if len(selected_years_list_clean) < 1:
            print('Not even 1 year was available, all years will be downloaded.')
            selected_years = None
    else:
        # En caso de que lo haya dejado vacío o bien introducido 2 caracteres o menos,
entendemos
        # que se quieren descargar todos los años
        selected_years = None

    # Ya tenemos todos los años que se quieren extraer por parte del usuario
    # Recorremos la lista de años obtenemos la información que necesitamos por año
    for year in years:
        print("***************************")
        print("Year to load: ", year)
        print("***************************")
        search_keyword = year

        # Esta es la función que hemos creado para extraer la información por año
        get_info_cpus(driver, year)

    # Cerramos el driver que habíamos creado
    driver.close()
```

CpuData.py

```
import urllib.request
from urllib.parse import urlparse
from bs4 import BeautifulSoup
import base64
import os

# Clase CpuData, dispone de los siguientes métodos:
        # __attr_init__ --> Para inicializar las distintas propiedades del objeto
        # __init__ --> Para inicializar el objeto, se apoya en el método __attr_init__
        # store_data --> Se utiliza para para almacenar los datos de una CPU junto con
sus imágenes
        # __get_b64_img --> Se utiliza para pasar a base64 una imágen en caso de que la
queramos guardar en el dataset
        # __collect_data --> Se utiliza para recoger los datos de la página web, lo que
permitirá luego almacenarlos
        # __download_url --> Es el método que baja el html de la página web para que se
pueda tratar

class CpuData:

    # Se utiliza para inicializar las distintas propiedades que tendrá el objeto
    def __attr_init__(self):
        # Sería la url de donde sacamos la información
        self.url = ''
        self.plain_html = ''
        self.soup_html = None

        # Los distintos datos que puede tener una CPU
        self.cpu_properties = {
            'cpu_name': '',
            'cpu_logo': '',
            'socket': '',
            'foundry': '',
            'process_size': '',
            'transistors': '',
            'die_size': '',
            'package': '',
            'tcasemax': '',
            'tj_max': '',
            'frequency': '',
            'turbo_clock': '',
            'base_clock': '',
            'multiplier': '',
            'multiplier_unlocked': '',
            'voltage': '',
            'tdp': '',
            'fp32': '',
            'market': '',
            'production_status': '',
            'release_date': '',
            'codename': '',
            'generation': '',
```

```
            'part_number': '',
            'memory_support': '',
            'ecc_memory': '',
            'pci_express': '',
            'cores_num': '',
            'threads_num': '',
            'smp_cpus_num': '',
            'integrated_graphics': '',
            'cache_l1': '',
            'cache_l2': '',
            'cache_l3': '',
            'features_list': [],
            'imgs': {},
            'notes': '',
            'pl1': '',
            'pl2': '',
            'chipsets': '',
            'chipset': ''
        }

        # Son las distintas tablas que contienen los datos de una CPU dentro de la
página web
        # Se deben cargar los datos de estas tablas para disponer de toda la información
de la CPU
        self.standard_tables = ['Physical', 'Performance', 'Architecture', 'Cores',
'Cache']

        # Se trata de un diccionario que nos permite traducir los distintos campos de la
página web de CPUs
        # a los distintos campos dentro de la propiedad cpu_properties que hemos
definido anteriormente
        self.enum_properties = {
            'Socket:': 'socket',
            'Foundry:': 'foundry',
            'Process Size: ': 'process_size',
            'Transistors:': 'transistors',
            'Die Size:': 'die_size',
            'Package:': 'package',
            'tCaseMax:': 'tcasemax',
            'Frequency:': 'frequency',
            'Turbo Clock:': 'turbo_clock',
            'Base Clock:': 'base_clock',
            'Multiplier:': 'multiplier',
            'Multiplier Unlocked:': 'multiplier_unlocked',
            'Voltage:': 'voltage',
            'TDP:': 'tdp',
            'FP32:': 'fp32',
            'Market:': 'market',
            'Production Status:': 'production_status',
            'Release Date:': 'release_date',
            'Codename:': 'codename',
            'Generation:': 'generation',
            'Part#:': 'part_number',
            'Memory Support:': 'memory_support',
```

```
            'ECC Memory:': 'ecc_memory',
            'PCI-Express:': 'pci_express',
            '# of Cores:': 'cores_num',
            '# of Threads:': 'threads_num',
            'SMP # CPUs: ': 'smp_cpus_num',
            'Integrated Graphics:': 'integrated_graphics',
            'Cache L1: ': 'cache_l1',
            'Cache L2: ': 'cache_l2',
            'Cache L3: ': 'cache_l3',
            'tJMax:': 'tj_max',
            'PL1:': 'pl1',
            'PL2:': 'pl2',
            'Chipsets:': 'chipsets',
            'Chipset:': 'chipset'
        }

    # Utiliza el método __attr_init__ para inicializar las distintas propiedades del
objeto
    # Además, lee la página web para la CPU de la que queremos descargar los datos
    def __init__(self, url):
        # Inicializamos los atributos
        self.__attr_init__()
        # En la propiedad url ponemos la url de la que queremos extraer la información
        self.url = urlparse(url)
        # Llamamos al método __download_url para leer el código html de la página y
        # pasarlo a la propiedad plain_html
        self.plain_html = self.__download_url(url)
        # Utilizamos BeautifulSoup para parsear el código html
        self.soup_html = BeautifulSoup(self.plain_html, 'html.parser')
        # Llamamos a la función que nos permite colectar los datos a partir del html ya
parseado
        self.__collect_data()

    # Método utilizado para almacenar la información de CPU en el disco
    def store_data(self, path, include_headers, create_images, img_as_path):
        if include_headers and not os.path.isfile(path):
            headers = ",".join(self.cpu_properties.keys())
            with open(path, 'w') as f:
                f.write(headers + "\n")
        data = ''

        # Recorremos los distintos campos de la estructura cpu_properties
        # Es donde van a almacenar los distintos datos de la página
        for key in self.cpu_properties:
            writed_data = False
            # Si el campo es cpu_logo, guardamos el path donde vamos a guardar la imagen
del logo
            if key == 'cpu_logo' and img_as_path:
                data = data + '"' + 'img/' + self.cpu_properties['cpu_name'].replace('
', '_') + '/' + \
                       self.cpu_properties['cpu_name'].replace(' ', '_') + '_logo.png' +
'"' + ','
                writed_data = True
            # Si se ha pulsado la opción de guardar las imágenes, almacenamos el logo
```

```
            if key == 'cpu_logo' and create_images:
                if self.cpu_properties['cpu_logo'] is not "":
                    if not os.path.isdir('img'):
                        os.mkdir('img')
                    if not
os.path.isdir('img/'+self.cpu_properties['cpu_name'].replace(' ', '_')):
                        os.mkdir('img/'+self.cpu_properties['cpu_name'].replace(' ',
'_'))
                    with open('img/' + self.cpu_properties['cpu_name'].replace(' ', '_')
+ '/' +
                              self.cpu_properties['cpu_name'].replace(' ', '_') +
'_logo.png', "wb") as f:
                        f.write(base64.decodebytes(self.cpu_properties['cpu_logo']))
            # Hacemos lo mismo con el resto de imágenes de la página web
            # indicamos la ruta donde vamos a grabar las imágenes
            if key == 'imgs' and img_as_path:
                imgs_paths = {}
                for img in self.cpu_properties[key]:
                    imgs_paths[img] = 'img/' + self.cpu_properties['cpu_name'].replace('
', '_') + '/' + \
                                      self.cpu_properties['cpu_name'].replace(' ', '_')
+ '_' + \
                                      img.replace(' ', '_') + '.png'
                data = data + '"' + str(imgs_paths) + '"' + ','
                writed_data = True
            # Bajamos las imágenes a la ruta que se haya indicado
            if key == 'imgs' and create_images:
                for img in self.cpu_properties[key]:
                    if not os.path.isdir('img'):
                        os.mkdir('img')
                    if not os.path.isdir('img/' +
self.cpu_properties['cpu_name'].replace(' ', '_')):
                        os.mkdir('img/' + self.cpu_properties['cpu_name'].replace(' ',
'_'))
                    with open('img/' + self.cpu_properties['cpu_name'].replace(' ', '_')
+ '/' +
                              self.cpu_properties['cpu_name'].replace(' ', '_') + '_' +
                              img.replace(' ', '_') + '.png', "wb") as f:
                        f.write(base64.decodebytes(self.cpu_properties[key][img]))
            if not writed_data:
                data = data + '"' + str(self.cpu_properties[key]) + '"' + ','
        data = data[0:-1]
        # En cada vuelta del bucle, almacenamos la información en el fichero de datos
        with open(path, 'a') as f:
            f.write(data + "\n")

    # Método para descargar una imagen y transformarla a base64
    @staticmethod
    def __get_b64_img(url):
        # Preparamos la url y la cabecera
        req = urllib.request.Request(
            url,
            data=None,
            headers={
```

```
                'User-Agent': 'UOC Test Bot. Img downloader. lgarciatar@uoc.edu &
mrodrise@uoc.edu'
            }
        )

        # Descargamos la imagen
        img = urllib.request.urlopen(req, timeout=10).read()

        # Transformamos la imagen a base64
        b64_img = base64.b64encode(img)

        return b64_img

    # Este método recoge la información de la página web y rellena las propiedades del
objeto
    # Implica conocer cómo está estructurada la página web, en base a los distintos tags
podrá
    # recolecar los disintos datos cara a rellenar las propiedades del objeto
    def __collect_data(self):

        # Nos vamos al punto del html donde podremos comenzar a recoger los datos
        article = self.soup_html.find('article')

        # Creo una lista con todos los tags de tipo div
        article_div_list = article.find_all('div')

        # Recorro la lista y voy recogiendo los distintos valores
        for div in article_div_list:
            # Si el div tiene clases, revisamos las distintas clases ya que dentro
estará la información

            if 'class' in div.attrs:

                # Si tiene la clase clearfix y no tiene la clase images, estamos en la
zona de nombre y logo
                if 'clearfix' in div.attrs['class'] and 'images' not in
div.attrs['class']:
                    for h1 in div.find_all('h1'):
                        if 'cpuname' in h1.attrs['class']:
                            self.cpu_properties['cpu_name'] = str(h1.string)
                    for img in div.find_all('img'):
                        if 'cpulogo' in img.attrs['class']:
                            self.cpu_properties['cpu_logo'] =
self.__get_b64_img('{uri.scheme}://{uri.netloc}'.

format(uri=self.url) +

img.attrs['src'])
                # Si tiene la clase clearfix y tiene la clase images, estamos en la zona
de imágenes
                # Por tanto, procedemos a descargarlas si el usuario lo ha solicitado
                if 'clearfix' in div.attrs['class'] and 'images' in div.attrs['class']:
                    for inner_div in div.findAll('div'):
                        if 'chip-image' in inner_div.attrs['class']:
```

```
                            src = inner_div.find('img', {"class": "chip-image--
img"}).attrs['src']
                            if 'http' not in src:
                                img =
self.__get_b64_img('{uri.scheme}://{uri.netloc}'.format(uri=self.url) + src)
                            else:
                                img = self.__get_b64_img(src)
                            name = str(inner_div.find('div', {"class": "chip-image--
type"}).string)
                            self.cpu_properties['imgs'][name] = img

                # Se trataría de la zona donde están las tablas con la información de la
CPU
                if 'sectioncontainer' in div.attrs['class']:
                    for section in div.find_all('section'):
                        if str(section.find('h1').string) in self.standard_tables:
                            table = section.find('table')
                            for row in table.findAll('tr'):
                                th = str(row.find('th').string)
                                td = str(row.find('td').text)

                                # Evaluamos el valor del campo de información, si no lo
tenemos lo añadimos
                                # al fichero no_enum para que el usuario disponga
también de estos datos
                                try:
                                    self.cpu_properties[self.enum_properties[th]] =
td.strip().replace("\n", "")\
                                        .replace("\r", "")
                                except Exception as e:
                                    with open('no_enum', 'a') as f:
                                        f.write(th)
                        if str(section.find('h1').string) == 'Features':
                            for item in section.findAll('li'):

self.cpu_properties['features_list'].append(str(item.string).strip().replace("\n", ""))
                        if str(section.find('h1').string) == 'Notes':
                            text_container = section.find('td', {"class": "p"})
                            self.cpu_properties['notes'] =
str(text_container.string).strip().replace("\n", "")

    # Este método se encarga de descargar el código html de la página web
    @staticmethod
    def __download_url(url):
        # Ponemos la cabecera e indicamos la url de la que vamos a descargar el código
        req = urllib.request.Request(
            url,
            data=None,
            headers={
                'User-Agent': 'UOC Test Bot. lgarciatar@uoc.edu & mrodrise@uoc.edu'
            }
        )

        # Hacemos la petición de descarga del código de la url
```

```
        html_doc = urllib.request.urlopen(req, timeout=10).read()

        # Devolvemos el código que hemos descargado
        return html_doc
```

10. Dataset. Publicación del dataset en formato CSV en Zenodo (obtención del DOI) con una breve descripción.

El dataset ha sido publicado en Zenodo bajo el siguiente enlace:
https://zenodo.org/record/4249728#.X6kJblCCGUk

El DOI del dataset es: 10.5281/zenodo.4249728

Contribuciones

A continuación, se indican las contribuciones realizadas por cada componente:

Contribuciones	Firma
Investigación previa	LGT y MERS
Redacción de las respuestas	LGT y MERS
Desarrollo código	LGT y MERS