



Universidad
Internacional
Menéndez Pelayo

Máster Universitario en Investigación en Inteligencia Artificial

Curso 2020-2021

**Recuperación y extracción de información,
grafos y redes sociales**

Práctica Bloque II: Recuperación de información y minería de texto

11 de abril de 2021

Laura Rodríguez Navas
DNI: 43630508Z

e-mail: rodrigueznabas@posgrado.uimp.es

Índice

| | |
|--|----------|
| 1. Resumen | 3 |
| 2. Rastreador web (crawler) | 3 |
| 3. K-Means | 5 |
| 3.1. Datos de entrada para K-Means | 5 |
| 3.2. Extracción de características | 5 |
| 3.3. Entrenamiento del algoritmo | 6 |
| 3.4. Visualización | 6 |
| Bibliografía | 6 |

1. Resumen

En esta práctica se ha implementado un rastreador web (crawler) en Python [1] (ver sección 2), que se complementa con un proceso de agrupamiento, también implementado en Python, de la información extraída de las páginas web que ha recopilado (ver sección 3).

2. Rastreador web (crawler)

En esta sección se describe como se ha implementado el rastreador web (crawler) en Python usando la librería [Scrapy](#). Para empezar con la implementación se debe ejecutar el siguiente comando:

```
$ scrapy startproject books
```

Este comando crea un proyecto Scrapy en el directorio books, siguiendo la [estructura por defecto](#) común para todos los proyectos Scrapy, y el fichero *scrapy.cfg* que contiene el nombre del módulo de Python que define la configuración del proyecto books (*books.settings*). El proyecto lo he nombrado books, porqué se rastreará el catálogo de libros que se encuentra en la página web: <http://books.toscrape.com>.

Una vez se ha creado el proyecto, se definen los ítems de cada libro que se quieren extraer del catálogo. En este caso los ítems que se van a extraer son: el título, la categoría, la descripción, el precio y la valoración de cada libro. Para ello, se tiene que modificar el fichero *books/items.py*, para incluir los cinco ítems que se quieren extraer. Vemos el contenido de *items.py* a continuación:

```
import scrapy

class BooksItem(scrapy.Item):
    # define the fields for your item here like:
    # name = scrapy.Field()
    title = scrapy.Field()
    category = scrapy.Field()
    description = scrapy.Field()
    price = scrapy.Field()
    rating = scrapy.Field()
```

El siguiente paso es describir la manera de extraer la información definida en el fichero *items.py*. Para ello, se utilizarán reglas de expresión [XPath](#) y [CSS](#). Por ejemplo, si nos fijamos en el código HTML de uno de los libros que se van rastrear (ver Figura 1), veremos que el título del libro es fácil de extraer con la siguiente regla de expresión CSS: "**h1 ::text**". Cuando la extracción de información se complica un poco más, se usan reglas de expresión XPath. Por ejemplo, para extraer las descripciones de todos los libros se usará la regla de expresión: "**//*[@id='product_description']/following-sibling::p/text()**". Una vez, definidas todas las reglas de expresión para cada ítem que se va a rastrear, se crea la araña *books/spiders/books_toscrape.py*.

Las arañas son clases que definen cómo se rastrea una página web determinada (o un grupo de páginas web), incluido cómo realizar el rastreo y cómo extraer la información deseada. En otras palabras, las arañas son el lugar donde se define el comportamiento personalizado para rastrear y analizar las páginas web. En el caso de la práctica, en la araña *books.toscrape* será el lugar donde se definen las reglas de expresión. En las arañas también se tienen que especificar las solicitudes iniciales para rastrear las URLs y una función de devolución de llamada (*parse*) a la que se llamará para generar los ítems de respuesta de esas solicitudes. Por último, los ítems devueltos por las arañas normalmente se conservan en una base de datos o se escriben en un archivo. En el caso de la práctica, en la araña *books.toscrape*, los ítems (título, categoría, descripción, precio y valoración de cada libro) serán guardados en el fichero *books.json*. Esta araña que procesa todas las URLs descubiertas de la

```
import scrapy

class BooksToscraperSpider(scrapy.Spider):
    name = 'books.toscraper'
    allowed_domains = ['books.toscraper.com']
    start_urls = ['http://books.toscraper.com/']

    def parse(self, response):
        for book_url in response.css("article.product_pod > h3 > a ::attr(href)").extract():
            yield scrapy.Request(response.urljoin(book_url), callback=self.parse_book_page)
        next_page = response.css("li.next > a ::attr(href)").extract_first()
        if next_page:
            yield scrapy.Request(response.urljoin(next_page), callback=self.parse)

    @staticmethod
    def parse_book_page(response):
        item = {}
        product = response.css("div.product_main")
        item["title"] = product.css("h1 ::text").extract_first()
        item['category'] = response.xpath("//ul[@class='breadcrumb']/li[@class='active']/preceding-sibling::li[1]/a/text()").extract_first()
        item['description'] = response.xpath("//div[@id='product_description']/following-sibling::p/text()").extract_first()
        price = response.xpath("//th[text()='Price (incl. tax)']/following-sibling::td/text()").extract_first()
        item['price'] = price.replace('£', '')
        rating = response.xpath("//*[contains(@class, 'star-rating')]/@class").extract_first()
        item['rating'] = rating.replace('star-rating', '')
        yield item
```

[illegible]

4 de 6

Finalmente, ya podemos iniciar la araña para que recupere la información del catálogo y la guarde en el fichero `books.json`:

```
$ cd books
$ scrapy crawl books.toscrape -o books.json
```

3. K-Means

En esta sección se describe como se ha implementado el proceso de agrupamiento en Python (ver directorio `kmeans` en [1]) usando la librería `scikit-learn` [2]. Existen muchos algoritmos de agrupación, y para esta práctica se ha elegido el algoritmo [K-Means](#). Concretamente, el algoritmo K-Means agrupará los títulos de los libros del catálogo web (recuperados en la sección 2) en diferentes clústeres.

3.1. Datos de entrada para K-Means

Si nos fijamos en el fichero `kmeans/kmeans.py`, vemos que empieza extrayendo la información de los libros almacenada en el fichero `books/books.json`, que contiene 1000 documentos, y la convierte en un `DataFrame`. Después se eliminan los valores NaN que pudieran existir en él y también es almacenado en un fichero CSV. Para ello, se usa la librería `pandas` [3].

```
# create df
books = open("../books/books.json", "r")
dict_books = json.load(books)
df = pd.DataFrame.from_dict(dict_books)

# remove null values from df
df = df.dropna()

# save df to csv
df.to_csv("books.csv", index=False, sep=",")
```

Definición 1. *Un `DataFrame` es una estructura de datos bidimensional etiquetada que acepta diferentes tipos de datos de entrada organizados en columnas. Se puede pensar en un `DataFrame` como una hoja de cálculo o una tabla SQL.*

Las diez primeras líneas que forman el conjunto de datos de entrada:

| title | category | description | price | rating |
|---|--------------------|--|-------|--------|
| Sapiens: A Brief History of Humankind | History | From a renowned historian comes a groundbreaking narrative of hum | 5423 | Five |
| Sharp Objects | Mystery | WICKED above her hipbone, GIRL across her heart Words are like a ro | 4782 | Four |
| Soumission | Fiction | Dans une France assez proche de la nôtre, un homme s'engage d | 5010 | One |
| Tipping the Velvet | Historical Fiction | "Erotic and absorbing...Written with startling power."--"The New York | 5374 | One |
| A Light in the Attic | Poetry | It's hard to imagine a world without A Light in the Attic. This now-clas | 5177 | Three |
| It's Only the Himalayas | Travel | Wherever you go, whatever you do, just . . . don't do anything | 4517 | Two |
| Libertarianism for Beginners | Politics | Libertarianism isn't about winning elections; it is first and foremost a | 5133 | Two |
| Mesaerion: The Best Science Fiction Stories 1800-1849 | Science Fiction | Andrew Barger, award-winning author and engineer, has extensively | 3759 | One |
| Olio | Poetry | Part fact, part fiction, Tyehimba Jess's much anticipated second book | 2388 | One |
| Our Band Could Be Your Life: Scenes from the American Indie Underground | Music | This is the never-before-told story of the musical revolution that hap | 5725 | Three |

Figura 2: Primeras líneas del conjunto de datos de entrada.

3.2. Extracción de características

Para cada título del conjunto de datos, calcularemos los valores de [TF-IDF](#).

```
vec = TfidfVectorizer(stop_words='english')
vec.fit(df['title'])
features = vec.transform(df['title'])
```

Ahora que tenemos la matriz de características, podemos enviarla al modelo para el entrenamiento.

3.3. Entrenamiento del algoritmo

3.4. Visualización

Bibliografía

- [1] Laura Rodríguez-Navas. Recuperación de información y minería de texto. <https://github.com/lrodrin/masterAI/tree/master/A14>, 2021.
- [2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [3] Jeff Reback, Wes McKinney, jbrockmendel, Joris Van den Bossche, Tom Augspurger, Phillip Cloud, gflyoung, Sinhrks, Simon Hawkins, Matthew Roeschke, Adam Klein, Terji Petersen, Jeff Tratner, Chang She, William Ayd, Shahar Naveh, Marc Garcia, Jeremy Schendel, Andy Hayden, Daniel Saxton, Vytutas Jancauskas, Ali McMaster, Pietro Battiston, Skipper Seabold, patrick, Kaiqi Dong, chris b1, h vetinari, Stephan Hoyer, and Marco Gorelli. pandas-dev/pandas: Pandas 1.1.5, December 2020.