UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Politècnica Superior d'Enginyeria de Vilanova i la Geltrú

RECUPERACIÓ DE LA INFORMACIÓ (REIN)

Sessió 3 laboratori: Rastrejador web amb Scrapy

En aquesta sessió:

- Aprendrem a implementar un senzill rastrejador web per extraure informació de les pàgines descarregades.
- Usarem l'ElasticSearch per emmagatzemar la informació.

1 Scrapy: un entorn per implementar rastrejadors web

L'Scrapy és una llibreria del Python per desenvolupar rastrejadors web i extraure informació de les pàgines descarregades. Aquesta llibreria està dissenyada per facilitar la implementació de rastrejadors que es dirigeixen a pàgines web específiques i l'anàlisi i l'extracció del seu contingut.

En aquesta sessió, veurem les bases del rastreig web però teniu un exemple més detallat en l'apartat tutorial de la documentació de l'Scrapy.

El codi bàsic per generar l'estructura d'un rastrejador web, es crea automàticament fent:

```
$ scrapy startproject reinscrapy
```

Aquesta acció crea un projecte per poder emmagatzemar diversos rastrejadors. Com a exemple, extraurem de la pàgina web d'UPC Commons la informació de les pàgines que guarden els TFG presentats per estudiants de la Facultat d'Informàtica de Barcelona en els darrers anys.

La url de la pàgina és http://upcommons.upc.edu/handle/2099.1/18595/recent-submissions

En un terminal, escriviu el següent:

```
$ cd reinscrapy/reinscrapy
```

```
$ scrapy genspider UPCommonsTFG upcommons.upc.edu
```

Això generarà el fitxer `UPCommonsTFG.py` dins el directori `spiders` amb una classe del Python pel rastrejador i la seva configuració bàsica. Heu de modificar la variable `start_urls` de manera que el punt de partida del rastrejador sigui:

```
http://upcommons.upc.edu/handle/2099.1/18595/recent-submissions
```

La classe només té un mètode, anomenat `parse`, que és el que rep les pàgines obtingudes pel rastrejador. Aquest és el mètode que extrau informació d'una pàgina i decideix quins enllaços seguir. Només té un paràmetre (`response`) que conté la resposta del servidor web a la petició del rastrejador. Ara mateix, el rastrejador no fa res, només descarrega la primera url i no genera cap sortida.

2 Parsing de pàgines web

2.1 Extraure informació d'un TFG

Per decidir com i quina informació extraure d'una pàgina web, primer hem d'analitzar com està emmagatzemada aquesta informació en la pàgina. Atès que l'`HTML` representa una pàgina web com un arbre amb diferents etiquetes i informació en els seus nodes, podem aprofitar aquesta estructura per accedir a allò que vulguem extraure.

L'Scrapy té dos mètodes per extraure informació d'una pàgina web. Un està basat en `CSS` i l'altre en `XPATH`. Nosaltres usarem el primer perquè és més simple (però és menys potent).

Afortunadament, les pàgines que volem rastrejar estan ben estructurades i les etiquetes `HTML` que tenen la informació que volem estan més o menys marcades. Per analitzar una pàgina, podem descarregar-la i obrir-la en un editor de textos o usar un navegador com Chrome o Firefox per inspeccionar-la fent `ctrl+shift+I`.

Mirant de prop l'estructura de la pàgina, podem veure que tots els TFG es troben dins l'etiqueta `<li>` de la classe `ds-artifact-item`. El mètode `css` de l'objecte emmagatzemat en el paràmetre `response`, ens permet extraure totes les aparicions d'aquesta etiqueta. Podem iterar sobre tots aquests elements per extraure la informació que contenen. El mètode `css` només extrau la part de l'arbre `HTML` que comença amb la `tag` passada com a paràmetre. Podem usar els atributs de l'etiqueta per ser més selectius. Cada element que extraiem pot ser parsejat usant el mateix mètode.

Cada element té un títol, una url que enllaça a la informació detallada del TFG, un autor, un editor, una data, els drets de publicació i un resum.

Aquest és el codi que trobareu dins la funció `parse` del fitxer `parse1.txt` dins dels fitxers d'aquesta sessió:

```
for tfg in response.css('li.ds-artifact-item'):
    doc = {}
    data = tfg.css('div.artifact-info')
    doc['title'] = tfg.css('h4 a::text').extract_first()
    doc['url'] = response.urljoin(tfg.css('h4 a::attr(href)').extract_first())
```

```
    doc['author'] = data.css('span.author span::text').extract_first()
    doc['publisher'] = data.css('span.publisher::text').extract_first()
    doc['date'] = data.css('span.date::text').extract_first()
    doc['rights'] = data.css('span.rights::text').extract_first()
    doc['abstract'] = data.css('div.artifact-abstract::text').extract_first()

    yield doc
```

Bàsicament, extrau les etiquetes `<li>` adequades usant el mètode `css` i, per cadascuna, s'aplica de nou el mètode `css` per parsejar cadascun dels camps d'informació. Els mètodes `extract_first` retornen la primera aparició o `None` si no n'hi ha. Per la url, afegim el domini de la web per completar-la si aquesta és relativa.

Tota la informació és emmagatzemada en un diccionari i retornada a l'Scrapy usant `yield`, que converteix la funció en un generador de manera que els elements de la pàgina són recuperats un a un quan el rastrejador els necessiti.

Ja podeu activar el rastrejador web fent:

```
$ scrapy crawl UPCommonsTFG -o tfg.json
```

En el canal estàndard de sortida podreu veure, barrejada amb *logs* de les accions que fa l'Scrapy, la informació de la primera pàgina de TFG que també s'emmagatzemarà en el fitxer `tfg.json` en format `JSON`.

2.2 Anant més enllà

La informació de la pàgina que conté la llista de TFG no és completa. Cada TFG té, de forma individual, una pàgina amb més informació com ara un resum complet i una llista de *keywords*. Hem obtingut l'enllaç a aquesta pàgina des de cada TFG i l'hem emmagatzemat en el camp `url`.

L'Scrapy permet seguir enllaços i afegir la informació obtinguda d'aquests enllaços a la que ja tenim. Per fer-ho, podem usar el mètode `Request`. Aquest mètode rep una `url`, la funció que processarà la pàgina obtinguda de la url i també se li pot donar informació que ja hàgim recollit prèviament.

El fitxer `parse2.txt` conté aquest codi actualitzat. Hem substituït el `yield` del diccionari `doc` per un `yield` del valor retornat per la crida `Request`. Aquesta rep una nova funció per parsejar la pàgina detallada (`parse_detail`) i els camps ja obtinguts en el paràmetre `meta`. Observeu que si també retornem a l'Scrapy la informació del diccionari `doc`, tindrem dos ítems per cada TFG: un de la llista de TFG i un altre de la pàgina detallada, cosa que no és el que volem.

La funció `parse_detail` extrau el resum complet i les keywords d'un TFG. El dissenyador de la pàgina, afortunadament, ha marcat les etiquetes com `expandable` (el resum complet) i `descripcio` (les keywords). El resum que es troba dins `expandable` està en diferents idiomes però no estan identificats en els tags. En la solució que teniu, apareix tot junt en un únic string. Podríeu usar un algorisme de detecció d'idioma per separar-los però això se us deixa com a exercici.

Substituïu la primera versió del codi per la de `parse2.txt` i executeu el rastrejador per veure'n els resultats.

2.3 Emmagatzematge dels ítems en l'ElasticSearch

Emmagatzemar les dades en un fitxer de text està bé però seria més útil fer-ho en una base de dades. L'Scrapy permet posar una pipeline enmig del procés de rastreig i emmagatzemar les dades en una base de dades (o qualsevol cosa que necessitem fer).

Substituïu el fitxer `pipelines.py` generat automàticament pel que teniu en els fitxers de la sessió. Aquest fitxer conté una classe amb uns mètodes que són cridats a l'inici i al final del procés de rastreig, i cada vegada que s'extrau un ítem. Si obriu el fitxer veureu que es crea un índex nou en l'ElasticSearch, anomenat `scrapy`, i que s'hi emmagatzemen cadascun dels ítems tal i com estan. Una acció que podria fer-se és comprovar si tots els camps extrets són vàlids, eliminant els ítems en el cas de que no ho fossin o canviant els valors no vàlids per uns altres "per defecte". Aquesta acció, nosaltres no la farem.

Per activar la pipeline heu de modificar primer el fitxer `settings.py`. Descomenteu la línia amb la configuració de `ITEM_PIPELINES` i canvieu-la fent:

```
ITEM_PIPELINES = {
'reinscrapy.pipelines.ReinscrapyElasticPipeline':  300,
}
```

2.4 Seguir els enllaços

Encara ens queda un pas per fer i és obtenir informació d'altres pàgines a més de la primera pàgina de la llista de TFG. Per fer això, només hem d'extraure de dins la pàgina l'enllaç que apunta cap a la pàgina següent. El dissenyador web també ha marcat aquest enllaç amb l'etiqueta `<a>` de la classe `next-page-link`. El tros de codi que es troba al final de la funció `parse` fa justament el que volem:

```
next = response.css('a.next-page-link::attr(href)').extract_first()}
```

```
if next is not None:
    next_page = response.urljoin(next)
    yield scrapy.Request(next_page, callback=self.parse)
```

Bàsicament, es busca un enllaç a la pàgina següent i, si existeix, se segueix. El fitxer `parse3.txt` conté el codi actualitzat. Ara el rastrejador seguirà l'enllaç a la pàgina següent fins que no hi hagi més pàgines.

2.5 El procés complet del rastreig

Ara ja podeu executar el rastrejador i obtenir les dades de tots els TFG. Primer inicieu l'ElasticSearch i després el procés de rastreig:

```
$ scrapy crawl UPCommonsTFG
```

El procés trigarà un parell de minuts (o més) i al final tindreu tota la informació dels TFG en la base de dades. Podeu fer consultes per cercar informació de l'índex. Teniu una versió modificada de l'script `SearchIndex.py` que us permet fer cerques en l'índex usant la sintaxi de `LUCENE`. La sintaxi permet posar com a prefix d'una paraula el camp que volem usar per la cerca. Per exemple, `author:jordi` buscarà la paraula `jordi` només en el camp `author`.

Podeu fer proves com ara:

```
$ python SearchIndex.py --index scrapy --query keywords:bases AND keywords:dades
$ python SearchIndex.py --index scrapy --query keywords:machine AND keywords:learning
$ python SearchIndex.py --index scrapy --query description:game
$ python SearchIndex.py --index scrapy --query description:dades
$ python SearchIndex.py --index scrapy --query title:dades~2
$ python SearchIndex.py --index scrapy --query author:miquel~1
```

Inventeu-vos altres consultes i mireu els resultats. També podeu modificar el codi per intentar extraure el director del projecte de la pàgina detallada del TFG o alguna altra informació.

3 Lliuraments

Heu d'escriure un breu report (2-3 pàgines màxim) on:

- Expliqueu l'objectiu del vostre rastrejador: què rastreja, quines dades extrau i principals modificacions que hàgiu fet al/s rastrejador/s donat/s com a exemple.

- Expliqueu si us ha funcionat fàcilment o heu tingut alguns problemes en la implementació (si n'heu tingut, expliqueu breument quins).

El que heu de penjar a la Tasca d'Atenea per aquesta activitat és un fitxer `.zip` que contingui:

1. un fitxer en format `.pdf` amb el report
2. el/s fitxer/s font `.py`
3. un fitxer amb un exemple de la sortida que produeix el vostre rastrejador