Web Crawling y análisis de datos con Python

GABRIEL M. LESKE
Tutor: Dr. Isaac Lera Castro
Trabajo Final del Máster Universitario en Tecnologías de la Información.
Universidad de las Islas Baleares

El presente trabajo tiene como objetivo la obtención de datos de una web turística para su explotación. En él se realiza una captura información a través del proceso de *web crawling*. La información capturada hace referencia a la oferta turística de viajes y estancias promocionadas a diferentes lugares del mundo. Dicha información es almacenada en una base de datos no relacional, procesada sintácticamente, y agrupada con el objetivo de encontrar algún tipo de patrón con el que poder obtener indicadores de la calidad de la oferta. Así pues, en este proyecto se usan técnicas de *web crawling* que consisten en inspeccionar páginas web de forma automatizada y extraer información de las mismas para propósitos como: búsqueda de enlaces rotos, indexado para acelerar motores de búsqueda o recopilación de información. Otras técnicas contempladas, son el procesamiento de lenguajes naturales, las cuales permiten identificar términos relevantes y patrones comunes, para clasificar la información y así almacenarla y utilizarla un modo eficiente.

Términos generales: Web crawling, scraping, Data Mining,

Palabras y frases adicionales: clustering, Scrapy, Scikit-learn, MongoDB, NoSQL, NLTK, Python.

1. INTRODUCCIÓN

El objetivo de este proyecto es extraer información, principalmente textos, de una web turística para la explotación. Esta web contiene anuncios en los que se ofrecen descuentos para viajes y hoteles, y cada uno de ellas presenta información de interés a extraer (título, lugar, descripción, precio, descuento, etc.).

El proceso se encuentra dividido en tres partes:
- Extracción.
- Almacenamiento.
- Análisis.

La primera consiste en desarrollar una aplicación capaz de extraer datos de forma automatizada. Para ello se utiliza un *framework* llamado Scrapy, que permite exportar los resultados en diferentes formatos, entre ellos JSON, es simple y eficaz.

La segunda parte se basa en el almacenamiento de la información. Para ello se usa MongoDB, una base de datos NoSQL, orientada a documentos. Estos documentos son almacenados en BSON, una representación binaria de JSON, lo que permite importar los resultados de la primera parte.

En la tercera parte analizan y agrupan los datos (*clústering*) en función de su contenido, para ello se emplea la librería Sci-klearn. Previamente se usa otra librería llamada Natural Language Tool Kit (NLTK) y para trocear el texto en palabras, eliminando información innecesaria como símbolos, saltos de línea, números, etc. Posteriormente Sci-klearn permite realizar el análisis sintáctico propiamente dicho: contar la cantidad de palabras, extraer las palabras más relevantes y *clústering* en función de parámetros deseados (contenido del texto, estrellas, lugar).

Además del *clústering*, se realiza una representación gráfica bidimensional del mismo y una agregación que incluye cálculo de la media de precios y descuentos, y las palabras más relevantes en función del lugar y las estrellas.

Por último, cabe destacar también que el presente proyecto está desarrollado íntegramente en Python y con software de código abierto.

2. TECNOLOGÍAS

2.1 Scrapy

El *web crawling* (raspado, recolección o rastreado *de páginas web*) consiste en la extracción de los datos significativos de una o varias páginas web determinadas, o de todas las páginas web que estén relacionadas mediante enlaces en un sitio web, para su posterior manipulación. Esta técnica (conocida bajo otros términos como *web scraping, web extraction, crawl spider, web-bot, spider robot, web mining, data harvest*, entre otros) consiste en capturar una página web, procesar algunas partes de su contenido y descartar otras. Entonces un *web crawler* es un programa diseñado para explorar páginas web en forma automática y extraer información. Inicialmente se le da a éste un grupo de direcciones iniciales, posteriormente el *crawler* descarga el contenido, lo analiza, extrae la información que desea y busca enlaces a otras páginas, repitiendo este proceso sucesivamente.

Scrapy puede ser utilizado para una amplia variedad de aplicaciones, como la minería de datos, el procesamiento de la información o procesamiento de históricos, y también puede ser usado para extraer información usando APIs.

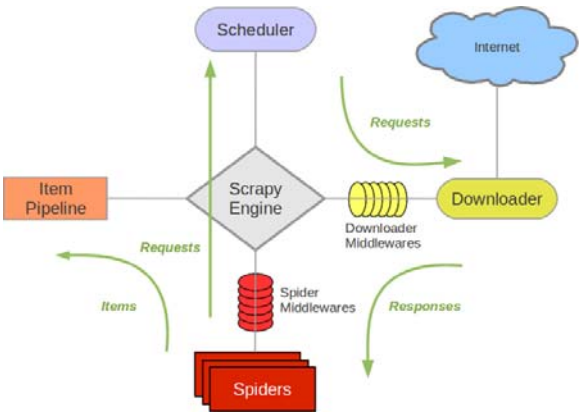

Fig. 1. Arquitectura de Scrapy.

2.1.1 En la Fig.1 está representada la arquitectura de Scrapy con los siguientes componentes:

Scrapy Engine: controla el flujo de datos entre todos los componentes del sistema, y activa eventos cuando se producen ciertas acciones.

Scheduler: recibe solicitudes desde el motor y las almacena ordenadamente para la suministrarlas más tarde cuando el motor las solicite.

Downloader: va a buscar páginas web y se las suministra al motor que, a su vez, se las suministra a las arañas.

Spiders: son clases personalizadas escritas por el usuario para parsear las respuestas y extraer ítems de ellas o parsear URLs adicionales (solicitudes) a seguir. Cada araña es capaz de manejar un dominio específico (o un grupo de dominios).

Item Pipeline: procesa los ítems una vez que han sido extraídos por las arañas. Las tareas típicas incluyen la limpieza, la validación y la persistencia (como el almacenamiento en una base de datos).

Downloader middlewares: se colocan entre el motor y el descargador y procesan solicitudes que pasan del motor al descargador y respuestas que pasan de descargador al motor. Proporcionan un mecanismo conveniente para extender la funcionalidad Scrapy añadiendo código personalizado.

Spider middlewares: se colocan entre el motor y las arañas y son capaces de procesar la entrada (respuestas) y salida (ítems y solicitudes) de la araña. Proporcionan un mecanismo conveniente para extender la funcionalidad Scrapy añadiendo código personalizado.

2.1.2 Flujo de datos

1. El motor abre un dominio, localiza la araña que se encarga de ese dominio, y le pregunta a la araña para las primeras URLs a rastrear.
2. El motor obtiene las primeras URLs a rastrear de la araña y las planifica en el planificador, como solicitudes.
3. El motor pregunta al planificador por las siguientes URLs a rastrear.
4. El planificador devuelve al motor las siguientes URLs a rastrear, y el motor las envía al descargador, pasando a través de la lógica de intercambio del descargador (en sentido solicitud).
5. Una vez la página termina de descargarse, el descargador genera una respuesta (con esa página) y la envía al motor, pasando a través del *Downloader middleware* (en sentido respuesta).
6. El motor recibe la respuesta del descargador y la envía a la araña para su procesamiento, pasando a través del *Spider middleware* (en sentido entrada).
7. La araña procesa la respuesta y devuelve elementos extraídos y nuevas solicitudes (para seguir) al motor.
8. El motor envía los elementos extraídos (devueltos por la araña) a la tubería de elementos, y solicitudes (devueltas por la araña) al planificador.
9. El proceso se repite (desde el paso 2) hasta que no hay más solicitudes del planificador, y el motor cierra el dominio [1].

2.2 MongoDB

MongoDB [2] (de la palabra en inglés "*humongous*" que significa enorme) es un sistema de base de datos orientado a documentos de código abierto y está programada en C++. Forma parte de la familia de sistemas de base de datos NoSQL. En vez de guardar los datos en tablas como se hace en las base de datos relacionales, MongoDB guarda estructuras de datos en documentos tipo JSON con un esquema dinámico (MongoDB llama ese formato BSON), haciendo que la integración de los datos en ciertas aplicaciones sea más fácil y rápida.

Un registro en MongoDB [3] es un documento tal como se ve en la Fig. 2, cuya estructura de datos está compuesta por parejas de clave valor. Los valores pueden incluir otros documentos, arreglos o arreglos de documentos.

Las ventajas de usar documentos son:

- Los documentos se corresponden con un tipo nativo de datos en la mayoría de lenguajes de programación.
- Los documentos anidados y los arreglos reducen la necesidad de costosos JOINs.
- El esquema dinámico soporta un polimorfismo fluido.

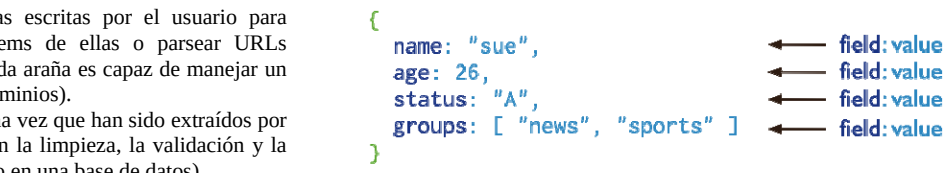

Fig. 2. Documento JSON.

2.3 Scikit-learn

El de *Machine Learning* [4] (aprendizaje automático) es una rama de la inteligencia artificial cuyo objetivo es desarrollar técnicas que permitan a las computadoras aprender. De forma más concreta, se trata de crear programas capaces de generalizar comportamientos a partir de una información no estructurada suministrada en forma de ejemplos. Es, por lo tanto, un proceso de inducción del conocimiento.

El proyecto Scikit-learn provee de una librería de aprendizaje automático para Python. Su ambición es proporcionar herramientas de aprendizaje automático eficientes y bien establecidas dentro de un entorno de programación que sea accesible a los no expertos en este campo y reutilizable en diversas áreas científicas [Lars Buitinck et al. 2013].

Es un conjunto de herramientas simples y eficientes para la minería y el análisis de datos [5]. Está construido sobre SciPy (Scientific Python), una librería fundamental en Computer Science, lo cual principalmente incluye:

- NumPy: permite trabajar con vectores y matrices.
- Pandas: permite hacer el análisis de datos.
- Matplotlib: permite la representación gráfica en 2D/3D.

Sin embargo Scikit-learn se enfoca en el modelado de datos, no se enfoca en la carga, la manipulación ni el sumatorio de los mismos, ya que para ello se usa NumPy y Pandas.

3. ESTADO DEL ARTE

3.1 Scrapy

El *web scraping* es el concepto básico en el que se basa Google para Indexar los sitios web de Internet y presentar los resultados en su buscador, empleando el *crawler* más famoso del mundo, llamado Googlebot [6].

[1]http://doc.scrapy.org/en/0.24/topics/architecture.html
[2]http://es.wikipedia.org/wiki/MongoDB
[3]http://docs.mongodb.org/manual/core/introduction/
[4]https://es.wikipedia.org/wiki/Aprendizaje_autom%C3%A1tico
[5]http://machinelearningmastery.com/a-gentle-introduction-to-scikit-learn-a-python-machine-learning-library/
[6]https://www.incapsula.com/blog/know-your-top-10-bots.html

Los tres *crawlers* de código abierto más robustos y ampliamente utilizados son [7]: Heritrix (Java), Nutch (Java) y Scrapy (Python). En este proyecto se utiliza Scrapy, es un *framework* multiplataforma y posee una buena documentación [8] mientras que Heritrix funciona sólo para Linux y Nutch ofrece una documentación escasa y confusa [9].

Un *crawler* llamado RWScraper para la identificación de palabras en rumano ha sido diseñado en torno a Scrapy [Ionut-Gabriel Radu et al. 2014]. En ese trabajo se afirma que Scrapy puede ser usado para implementar web *crawlers* personalizados, ya que provee de la infraestructura básica necesaria para extraer contenido web basándose en varias reglas definidas por el usuario. Otro ejemplo similar es ScraperWiki [Mikhail Galkin et al. 2015], también basado en Scrapy, que ha sido utilizado para la identificación de contenido web en tablas. La fusión de rendimiento, velocidad, extensibilidad y simplicidad hacen que Scrapy sea una solución popular en la industria. Muchos servicios se basan en Scrapy, tales como ScraperWiki o PriceWiki.

3.2 MongoDB

MongoDB es una de los sistemas de base de datos más popular y con mayor crecimiento en los últimos años (Fig. 3). Una plataforma para la minería de diseño a gran escala llamada Webzeitgeist [Ranjitha Kumar et al. 2013] utiliza MongoDB, ésta comprende un depósito de más de 100.000 páginas web y 100 millones de elementos de diseño. Otro trabajo compara MongoDB y Have frente a otras bases SQL realizando diferentes *benchmarks* [Avrilia Floratou et al. 2012]. Han demostrado que mientras las bases de datos relacionales pueden funcionar mejor, los sistemas NoSQL tienen ventajas en la usabilidad como modelos de datos flexibles, *auto-sharding*, tolerancia a fallos y balanceo de carga. Dichas ventajas han sido evaluadas y ratificadas en otro estudio [Elif Dede et al. 2013] que prueba la funcionalidad de MongoDB junto con Hadoop, la implementación más popular de MapReduce.

3.3 Scikit-learn

Se ha construido un ejemplo de API con Scikit-learn, en la que implementa todas sus características [Lars Buitinck et al. 2013]. Además se afirma que Scikit-learn es cada vez más popular, está diseñado para ser simple y eficiente, accesible a los no expertos, y reutilizable en distintos contextos. Por otra parte, en otro trabajo [Fabian Pedregosa et al. 2013] se concluye que Scikit-learn expone una amplia variedad de algoritmos de aprendizaje automático, tanto supervisados como no supervisados, utilizando una interfaz consistente y orientado a las tareas, lo que permite una fácil comparación de métodos para una aplicación determinada.

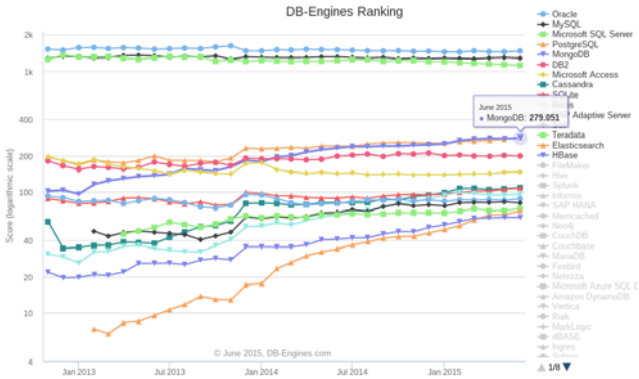

Fig. 3. Ranking de popularidad de bases de datos [10].

4. DESARROLLO

La arquitectura de la aplicación (Fig. 4) comienza con la ejecución del *crawler* hecho con Scrapy para extraer los datos de la web. Como resultado de la extracción se crean dos archivos JSON, los que posteriormente son importados a MongoDB, para luego analizar la información con Scikit-learn.

Se usa el sistema operativo Linux Manjaro 64 bits, una distribución derivada de Arch linux, y su gestor de paquetes Yaourt. También se utiliza la aplicación virtualenv, un entorno virtual que permite "aislar" el proyecto en cuestión, permitiendo utilizar dependencias en sus respectivas versiones sin que se mezclen con otros proyectos o con el entorno global del sistema operativo. Virtualenv instala automáticamente pip (Python Index Package), que permitirá instalar el resto de herramientas (como Scrapy, MongoDB y Scikit-learn) y sus dependencias necesarias.

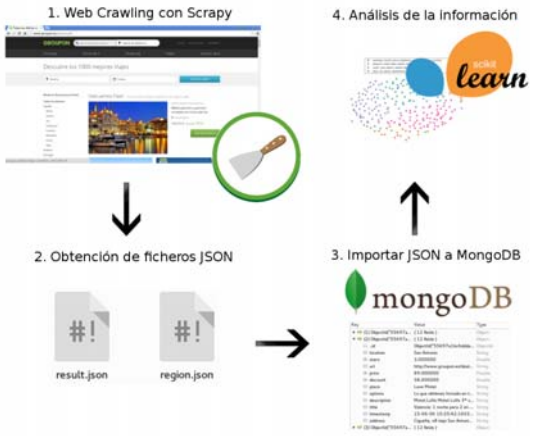

Fig 4. Arquitectura del desarrollo.

4.1 Extracción

Scrapy se configura principalmente a través de dos archivos: items.py, donde se definen el nombre de los elementos a extraer, y grouponSpyder.py, donde se define el comportamiento de la araña: dominio inicial, estrategia u orden de navegación, elementos a extraer, entre otros. Scrapy para extraer la información se basa en *xpaths* (Fig. 5), son rutas únicas que identifican a cada elemento dentro del documento html (Fig 6).

Un aspecto importante a tener en cuenta, es que la navegación que realiza Scrapy es completamente por consola, esto significa que no contempla dentro del contenido del html datos originados asíncronamente mediante AJAX. Para poder obtener este contenido se ha utilizado Selenium, una librería que permite abrir la web gráficamente en el navegador, y disponer de todo su contenido.

[7]http://bigdata-madesimple.com/top-50-open-source-web-crawlers-for-data-mining/
[8] http://link.springer.com/chapter/10.1007/978-3-642-41175-5_34#page-1,
[9] http://blog.blikk.co/comparison-of-open-source-web-crawlers/
[10]http://db-engines.com/en/ranking_trend

```
item_dict = {
     'title':       '//*[@class="deal-page-title"]/text()',
     'price':       '//*[@id="deal-hero-price"]/span[2]/text()',
     'discount':    '//*[@id="purchase-cluster"]/div[3]/table/tbody/tr[2]/td[2]/text()',
     'description': '//*[@id="tabs-1"]/div/article[1]/div//text()',
     'address':     '//*[@id="redemption-locations"]/li/div[2]/p[2]/text()',
     'place':       '//*[@id="redemption-locations"]/li/div[2]/p[1]/strong/text()',
     'options':     '//*[@id="tabs-1"]/div/article[2]/div[2]//text()'
     }
```

Fig. 5. Xpaths de cada uno de los elementos a extraer.

4.1.1 Estrategia de navegación

La página web utilizada muestra unas ventanas emergentes después de su carga inicial. Una vez cerradas, los anuncios se cargar por AJAX y aparecen en bloques de dos en dos. Además al cargar inicialmente la página no carga todos los bloques, sino que en la sección inferior posee un botón de "Mostrar más anuncios", que al ser pulsado añade nuevos bloques, hasta que desaparece cuando todos los anuncios existentes son mostrados. Por lo tanto, el primer objetivo de la araña es remover las primeras ventanas emergentes y posteriormente pulsar el botón repetidas veces hasta mostrar todos los anuncios. Entonces se extraen todos los links, uno correspondiente a cada anuncio, para posteriormente ser parseados individualmente. El contenido html de cada anuncio no posee contenido AJAX, por lo que se no es necesario cargar cada página en el navegador, sino que Scrapy extrae y muestra su contenido por consola.

Se crea además otro *crawler* que permita agrupar los anuncios por regiones. Su estrategia de navegación consiste, en remover las ventanas emergentes, y examinar cada uno de los elementos de la sección de destinos, cargar todos los existentes pulsando el botón de más anuncios, y guardar todos los links que aparezcan en cada uno de ellos.

Fig. 6. Una de las ofertas donde se observa la información a extraer: título, precio, descuento, dirección, descripción, etc.

4.1.2 Ejecución

Ya finalizada la configuración del *crawler*, se ejecuta pone en acción mediante un comando el cual permite definir como parámetros el nombre del archivo de salida y el tipo (csv, xml o json). Los comandos utilizados para cada uno de los dos *crawlers* son:

scrapy crawl grouponScrapy -o result.json -t json
scrapy crawl grouponRegionScrapy -o region.json -t json

4.2 Almacenamiento

Una vez extraída la información deseada en los archivos JSON, se pueden importar a MongoDB fácilmente. Pero primero mediante Robomongo, una interfaz gráfica de usuario para MongoDB, se crea una nueva base de datos, llamada tfm, a la que se importarán los datos extraídos con los comandos:

mongoimport --db tfm --collection result --type json --file result.json –jsonArray
mongoimport --db tfm --collection region --type json --file region.json –jsonArray

4.3 Análisis

Ya con las colecciones en base de datos, se procede al análisis de los datos. Para interactuar con MongoDB desde Python, se utiliza la librería Pymongo. Previo al análisis se hace un troceado de los textos en palabras con NLTK, removiendo todos aquellos símbolos y números. Luego se realizan los siguientes análisis:

- Agrupamiento con Kmeans y extracción de términos más relevantes por clúster.
- Representación gráfica de los clústers.
- Agregación:
    - Cálculo del precio medio y descuento medio por clúster y por categoría.
    - Términos más relevantes por región y por categoría.

4.3.1 *Agrupamiento con Kmeans*

Como primer paso se obtiene la frecuencia de término (*term frequency* o tf), y se representa en una matriz llamada matriz término-documento (*document-term matrix* o dtm) (Fig. 7), básicamente es una especie de tabla en la que aparece una fila por cada término existente, y una columna por cada documento. Cada fila representa un vector de palabras, ya que en cada posición contiene la cantidad de veces que esa palabra aparece en cada documento. Cada columna representa un vector de documento, ya que cada posición contiene la cantidad de veces que aparece cada término.

Por otra parte, se obtiene la frecuencia inversa de documento (*inverse document frequency* o idf), una medida que representa si determinado término es común o no en una colección de documentos. Se obtiene dividiendo el número total de documentos por el número de documentos que contienen el término, y se toma el logaritmo de ese cociente. El resultado es un vector con una posición por cada término, cuyos valores serán el resultado de la operación mencionada anteriormente.

Finalmente se obtiene la matriz tf-idf, el resultado de multiplicar cada valor tf por su idf, en otras palabras, multiplicar los valores de cada columna de la dtm por el vector idf.

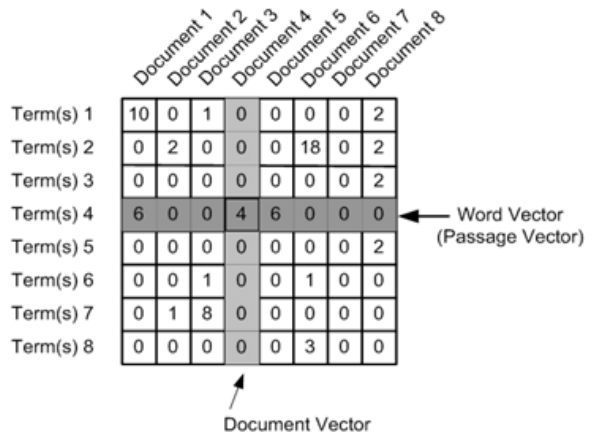

Fig 7. Estructura de la Matriz dtm [11].

Como el cociente dentro de la función logaritmo del idf es siempre mayor o igual que 1, el valor del idf (y del tf-idf) es mayor o igual que 0. Cuando un término aparece en muchos documentos, el cociente dentro del logaritmo se acerca a 1, ofreciendo un valor de idf y de tf-idf cercano a 0.

En resumen, un peso alto en tf-idf se alcanza con una elevada frecuencia de término en el documento dado y una pequeña frecuencia de ocurrencia del término en la colección completa de documentos.

Ahora se tiene una representación vectorial de cada documento en función de todos los términos existentes.

La obtención de la matriz tf-idm con Scikit-learn es sencilla y configurable:

- Permite añadir una lista palabras que no aportan significado para ser ignoradas (*stopwords)*. Se usa una lista mixta compuesta por: las *stopwords* pertenecientes a la librería NLTK, una lista obtenida de internet [12], más un conjunto de términos agregados manualmente para optimizar los resultados.
- Permite añadir una función personalizada para trocear el texto en palabras (*tokenizer*).
- Tiene parámetros muy interesantes como min_df y max_df. Éstos son valores entre 0 y 1 que representan un porcentaje mínimo y máximo de aparición de alguna palabra para ser tenida en cuenta. Por ejemplo, si min_df es 0.1, cualquier palabra que aparezca menos de en un 10% del total no será tenida. Por otra parte, si max_df es 0.9 significa que aquellas palabras que aparecen en más del 90% del total son ignoradas.

A partir de la matriz tf-idf se procede a aplicar Kmeans [APÉNDICE A], indicando el número de clústers deseados.

4.3.2 *Representación gráfica de los clústers*

A partir de matriz tf-idf, se obtiene una matriz de distancias, cada posición es el resultado de la distancia de un documento con otro. A partir de la matriz de distancias se aplica un escalado multidimensional para representar cada documento como una pareja de coordenadas X e Y. Dependiendo del clúster al que pertenezca cada documento, se le asigna un color y se usa la librería Matplotlib dibujar el gráfico. Además Scikit-learn permite extraer los términos más cercanos a cada centroide, los que son usados en la leyenda (Fig. 8).

4.3.3 *Agregación*

La librería Pandas permite la creación de marcos o estructuras de datos haciendo posible una manipulación interesante de los mismos. Un ejemplo de ello es la agrupación de datos según parejas de parámetros, como las siguientes:

- Agrupación de precios según clústers con cálculo de medias.
- Agrupación de descuentos según clústers con cálculo de medias.
- Agrupación de precios según categoría del anuncio (estrellas) con cálculo de mínimos, máximos y medias.
- Agrupación de precios según categoría con cálculo de mínimos, máximos y medias.
- Agrupación de documentos según categoría y extracción de términos más importantes.
- Agrupación de documentos según región y extracción de términos más importantes.

5. ANÁLISIS DE LOS RESULTADOS

5.1 Extracción

Mediante la ejecución del crawler en diferentes intervalos de tiempo y se aprecia que la web seleccionada siempre tiene aproximadamente entre 140 y 200 anuncios, y aparecen alrededor de 5 anuncios nuevos cada día.

En el log de Scrapy (Fig. 9) se observa: item_scraped_count 162, start_time y finish_time, cuya diferencia es de 113 segundos. Esto significa que ha examinado 1.43 ofertas por segundo. Es necesario tener en cuenta que en este tiempo está incluido la carga del navegador a través de Selenium, que son aproximadamente 5 segundos.

Otra característica particular de Scrapy es que cuando guarda los resultados en JSON, no sobrescribe el archivo existente con el mismo nombre, sino que anida los elementos a partir del fichero ya existente.

5.2 Almacenamiento

Para añadir nuevas ofertas a la base de datos y a la vez preservar las ya encontradas *mongoimport* permite utilizar el parámetro *upsertFields*, que permite definir uno o varios campos como índices. En este caso se usa como índice la URL de cada oferta, así que en el caso de ya existir la URL el proceso de importación actualiza el resto de campos del documento, y cuando no existe pues crea un documento nuevo.

No obstante para actualizar la colección de regiones, se implementa un script en Python, que va anidando las URLs a cada región, ya que no es posible anidar elementos a un campo mediante *mongoimport*.

[11]http://brandonrose.org/clustering
[12]http://members.unine.ch/jacques.savoy/clef/spanishST.txt

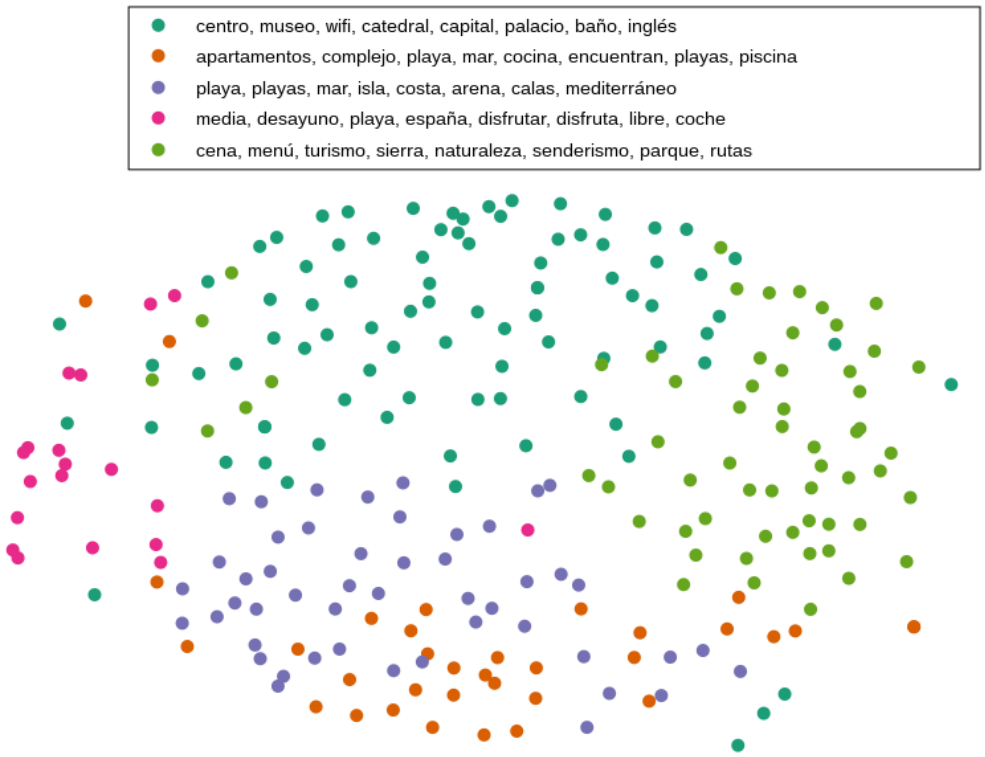

Fig. 8. Representación gráfica del *clústering*.

```
2015-06-18 10:36:53+0200 [grouponScrapy] INFO: Closing spider (finished)
2015-06-18 10:36:53+0200 [grouponScrapy] INFO: Stored json feed (162 items) in: result.json
2015-06-18 10:36:53+0200 [grouponScrapy] INFO: Dumping Scrapy stats:
     {'downloader/request_bytes': 115616,
      'downloader/request_count': 163,
      'downloader/request_method_count/GET': 163,
      'downloader/response_bytes': 3678953,
      'downloader/response_count': 163,
      'downloader/response_status_count/200': 163,
      'finish_reason': 'finished',
      'finish_time': datetime.datetime(2015, 6, 18, 8, 36, 53, 924431),
      'item_scraped_count': 162,
      'log_count/DEBUG': 327,
      'log_count/INFO': 9,
      'request_depth_max': 1,
      'response_received_count': 163,
      'scheduler/dequeued': 163,
      'scheduler/dequeued/memory': 163,
      'scheduler/enqueued': 163,
      'scheduler/enqueued/memory': 163,
      'start_time': datetime.datetime(2015, 6, 18, 8, 35, 0, 509713)}
2015-06-18 10:36:53+0200 [grouponScrapy] INFO: Spider closed (finished)
```

Fig. 9. Log de Scrapy.

5.3 Análisis

El agrupamiento más lógico de las ofertas consta de 5 clústeres. Según las palabras clave (Fig. 10) se puede observar que la temática de cada clúster es distinta. El clúster 0 se refiere a ofertas relacionadas con la naturaleza, el 1 a zonas de costa, el 2 a anuncios relacionados con alimentación y bebidas, el 3 está relacionado con zonas de ciudad y el último observando sólo las palabras no queda claro exactamente su temática, pero luego de analizar las ofertas se deduce que está relacionadas con relax (spa, balnearios, masajes, etc).

Cluster	Cantidad	Palabras
0	60	naturaleza, turismo, sierra, senderismo, parque, región, rutas, activo
1	80	playa, mar, playas, apartamentos, costa, isla, arena, metros
2	20	cena, menú, bebidas, vino, casa, restaurante, cafetería, patrimonio
3	68	centro, palacio, capital, catedral, museo, estación, ofrece, wifi
4	20	desayuno, media, playa, disfrutar, españa, libre, coche, pequeño

Fig. 10. Resultado del *clústering*.

Un dato interesante (Fig. 11) es que el descuento medio por clúster es muy similar, entre el 53-56%. El clúster 4 destaca por tener precio medio más caro 194.3€, por lo tanto los términos de este clúster como disfrutar, coche y libre le dan más valor a las ofertas. Le sigue el clúster 1 con un precio medio de 133.9€, siendo el clúster con mayor cantidad de ofertas con términos como costa, mar e isla. Cabe destacar que el término playa es significativo en ambos clústers, en consecuencia puede decirse que es el que tiene más valor. Por otro lado, el clúster 2 tiene el precio medio más bajo y el menor número de ofertas con términos como cena, menú y restaurante.

Cluster	Descuento Medio	Precio Medio
0	55.196429	63.206897
1	54.75	133.909091
2	55.263158	50.578947
3	53	97.651163
4	56.5	194.333333

Fig.11. Descuento y precio medio por clúster.

En la Fig. 12 se puede observar las palabras clave por categoría, éstas están ordenados por pesos de mayor a menor. El término centro es relevante en ofertas de 3 y 4 estrellas, al igual que el término palacio lo es para las de 4 y 5, y el término wifi lo es para las de 2 y 3. Puede interpretarse que la mayoría de las ofertas de 5 estrellas son de media pensión con desayuno, las de 2 estrellas son hostales, y en las de 3 estrellas donde más mascotas se admiten. Por último destacar que sólo aparece una oferta de 1 estrella.

```
Palabras clave por categoría (estrellas):
1: apartamentos, canaria, playa, junio, mar, mayo, palmera, agosto
2: hostal, municipio, wifi, zona, patrimonio, aprovecha, comarca, playa
3: wifi, centro, baño, guía, tv, catedral, mascotas, partir
4: centro, aire, piscina, baño, spa, wifi, palacio, tv
5: desayuno, media, pensión, palacio, disfrutar, golf, tiempo, locales
```

Fig.12. Palabras clave por categoría (estrellas).

La clasificación por regiones (Fig. 13) está la hecha por la misma web, ya que resulta complejo realizar de un modo sencillo un agrupamiento a partir de la dirección de todas las ofertas, además que en gran medida se trata de ofertas en España, por lo que este agrupamiento es válido para analizar. Nuevamente los términos playa y wifi son de los que más abundan. Es curioso que en regiones de Montaña el término más destacado sea playa y que en Centro aparezcan términos como rural y aire. Se puede apreciar que las palabras hacen referencia a la temática de cada región como por ejemplo en Europa museo e inglés, España playa, en el Norte gastronomía y naturaleza, y en Levante castillo e histórico.

```
Islas: playa, wifi, mar, centro, zona, playas, baño, gratuito
Especial: precio, viaje, oportunidad, directo, mundo, pensión, españa, mínimo
Europa: museo, centro, baño, tv, inglés, guía, aire, encuentra
España: playa, mar, wifi, zona, baño, parque, guía, naturaleza
Portugal: playa, wifi, zona, mar, baño, centro, guía, gratuito
Costa: mar, playa, costa, castillo, municipio, mascotas, visitar, zona
Norte: mar, gastronomía, naturaleza, restaurante, wifi, baño, playa, guía
Centro: cena, madrid, palacio, centro, aire, rural, tv, wifi
Montaña: playa, wifi, mar, baño, zona, centro, gratuito, playas
Catalunya: barcelona, mar, zona, actividades, municipio, mediterráneo, playas, guía
Andorra: julio, andorra, jueves, apartamentos, caldea, sábado, montaña, rutas
Sur: granada, cena, wifi, playa, parque, partir, guía, zona
Fuera de Europa: ammán, barcelona, madrid, desayuno, salida, prevista, traslado, llegada
Levante: playa, apartamentos, murcia, mar, castillo, costa, histórico, castellón
```

Fig.13. Palabras clave por región.

6. CONCLUSIONES

En primer lugar, la extracción de datos de la web seleccionada no ha proporcionado gran cantidad de resultados (aprox. 200 como máximo), por lo que ha sido necesario realizar más extracciones para anidar datos nuevos y así poder enriquecer los resultados. Luego el almacenamiento los resultados, tanto en la primera importación como en los posteriores anidamientos, ha funcionado siempre de forma rápida y sin causar problemas.

Por último, el análisis ha permitido encontrar patrones comunes en las ofertas, haciendo posible agruparlas en función de su contenido y deducir cuáles son las palabras que les proporcionan mayor o menor valor.

En resumen, ha sido posible realizar la extracción, almacenamiento y análisis de una web turística, gracias al uso del lenguaje Python mediante la combinación de las tecnologías como Scrapy, MongoDB, NLTK y Scikit-learn. Scrapy ha resultado realmente simple y rápido, tanto en el desarrollo como en la implementación, pero cuando se trata de cargar contenido asíncrono pierde algo de eficacia y es necesario utilizar librerías como Selenium, que permiten hacer una navegación gráfica y hacen el proceso más lento. La posibilidad de Scrapy de generar un fichero de salida JSON, hace sencilla la importación de la información en MongoDB, la cual puede hacer *merge* (fusión de los datos) sin complicación, ya sea a través de comando propio o con la creación de un script. Finalmente las herramienta de análisis como NLTK y Scikit-learn son realmente poderosas, rápidas y configurables, permitiendo implementar, en pocas líneas, análisis complejos con gran cantidad de parámetros.

7. TRABAJO FUTURO

Es importante afirmar que cualquiera de las tecnologías y procesos utilizados en el presente trabajo pueden ser aplicados a cualquier campo. Sería posible una continuación del mismo mediante el indexado de la información para implementar buscadores, o clasificar grandes volúmenes de información, o simplemente realizar estudios estadísticos. Finalmente, este proyecto puede ser ampliamente aplicado a la obtención de datos con el objetivo de iniciar nuevas aplicaciones o extender aplicaciones ya existentes.

APÉNDICE

A. KMEANS

Kmeans es un algoritmo de agrupamiento basado en iteraciones. Funciona del siguiente modo (Fig. 14):

1. Se inicializa con un número K de clúster elegido por defecto, y toma K cantidad de muestras al azar (en este caso documentos) como centroides.

2. Se calcula la distancia de cada muestra respecto de cada centroide, y se asigna la muestra al clúster cuya distancia es menor. Existen diferentes tipos de distancias, las más comunes son:
   - La distancia euclidiana (equivalente a la distancia entre dos puntos aplicando el teorema de Pitágoras). Siendo P y Q vectores del mismo tamaño:

$$d_E(P,Q) = \sqrt{(p_1-q_1)^2 + (p_2-q_2)^2 + \cdots + (p_n-q_n)^2} = \sqrt{\sum_{i=1}^{n}(p_i-q_i)^2}$$

   - La similitud coseno. Siendo A y B vectores del mismo tamaño:

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n}(A_i)^2} \times \sqrt{\sum_{i=1}^{n}(B_i)^2}}$$

3. Se recalculan los centroides para cada clúster, como los valores medios de todas las muestras asignadas al clúster.
4. Finalmente se repite el proceso desde el segundo paso hasta que no se produzcan reasignaciones de *clústers*.

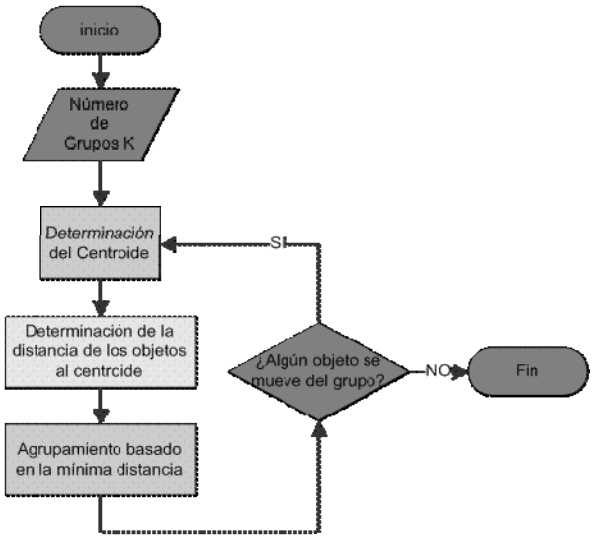

Fig.14. Algoritmo Kmeans [13].

REFERENCIAS

Lars Buitinck, Gilles Louppe, Mathieu Blondel. 2013. API design for machine learning software: experiences from the scikit-learn project.

Ionut-Gabriel Radu, Traian Rebedea. 2014. A Focused Crawler for Romanian Words Discovery.

Mikhail Galkin, Dmitry Mouromtsev, Sören Auer. 2015. Identifying Web Tables – Supporting a Neglected Type of Content on the Web.

Ranjitha Kumar, Arvind Satyanarayan, Cesar Torres. 2013. Webzeitgeist: Design Mining the Web.

Avrilia Floratou, Nikhil Teletia, David J. DeWitt. 2012. Can the Elephants Handle the NoSQL Onslaught?

Elif Dede, Madhusudhan, Daniel Gunter. 2013. Performance Evaluation of a MongoDB and Hadoop Platform for Scientific Data Analysis.

Fabian Pedregosa, Ga¨el Varoquaux, Alexandre Gramfort. 2013. Scikit-learn: Machine Learning in Python.

REPOSITORIO

https://github.com/leskeg/tfm

[13]http://people.revoledu.com/kardi/tutorial/kMean/EjemploNumerico.htm