CSL 707

ASSIGNMENT 3

CONTRIBUTORS

1. Gaurav Mittal - 2012CSB1013
2. Rahul Mittal - 2012CSB1027
3. Kaushal Yagnik - 2012CSB1039

PROBLEM DESCRIPTION

To build an application which automate the tracking of prices of all products which are sold on a particular e-stores allowing users to analyze the trend in the prices and make some profitable decisions based on them.

APPLICATION SUMMARY

The application is essentially a crawler which on providing a start URL and a particular depth traverses the website (Amazon.in) searching for products and scraping their corresponding webpages for the following details:

- Product Name and Description
- Product's unique URL
- Product's Price
- Rating
- Number of Reviews

After collecting the necessary details, they are dumped into a database collection to be used later for observing price trends through graphs and performing specific web scraping based on URLs collected.

Finally, the client interface provides the user to analyze the price trends by specifying the product URL of the product and obtain a graph of the prices recorded over the various times the webpage is crawled by the application.

APPLICATION DESIGN

The entire application can be divided into three key components:

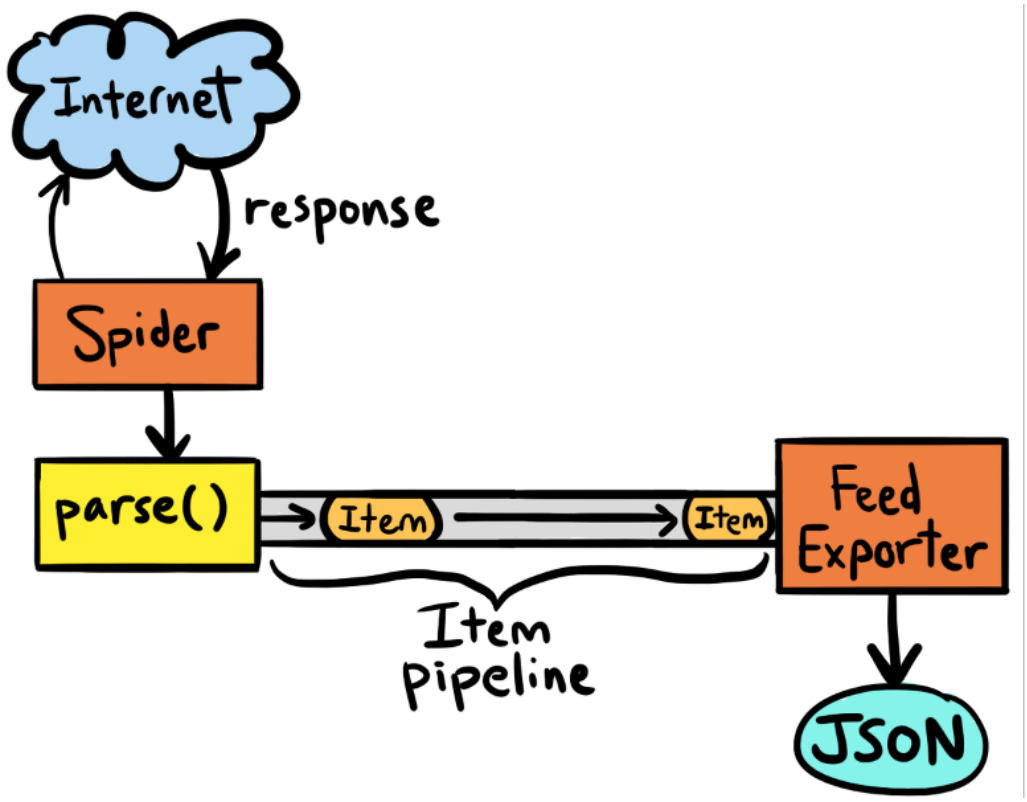
1. Crawling the website and parsing the necessary details
2. Dumping the acquired details into the database
3. Using the details to plot graphs for analyzing price trends of the products

CRAWLING THE WEBSITE AND PARSING THE DATA

The crux of the application lies in being able to traverse through the target e-commerce website as far as possible, extracting links and visiting then and finally, obtain the relevant data by characterising the desired webpages and segments of the webpages following by parsing them suitably.

In order to accomplish this task of crawling the website and parsing/scraping for the necessary data, Scrapy is used.

“Scrapy is an open source and collaborative python based framework for extracting the data you need from websites in a fast, simple and yet extensible way” - http://scrapy.org



Source : http://asheesh.org/pub/scrapy-talk/_images/scrapy-diagram-1.png

The above image explains the working of Scrapy which is essentially how the application goes about crawling the website and parse for relevant data:

- ➔ Scrapy allows building custom webcrawlers and spiders which crawl the website starting from a particular starting URL and upto a particular depth, all of which can be very easily specified and customized.

- ➔ It also allows to specify certains rules based on which the crawler will decide whether the particular webpage needs to be parsed, whether it need to halt there or simple continue crawling further deep.

- ➔ Next, in case the particular web page needs to be parsed, certain parts of it are specified in terms of there xpath.

- ➔ Finally all the data that is needed is encapsulated in a set of objects serving as a representation for this data both in the form it is collected and the way it is further processed in the various pipelines.

- ➔ In this particular application, two spiders are created by the name of aragog_spider and update_spider:
  - ◆ aragog_spider: This spider is responsible for a complete crawling of the website starting from the set of start_urls it is provided with.
  - ◆ update_spider: This spider is responsible to crawl and parse on specific URLs present in the database with the aim to update the pricing of the products for trend analyzing.

- ➔ The item representing the data collected from the website is AragogItem containing details like the product ID, name, decription, price and other things.

DUMPING INTO THE DATABASE

- ➔ On successful creation of an object representing the collected data, it is channelled to various available pipelines for things like simply displaying it on the screen or storing it in some database.

- ➔ In this application, MongoDb is used for storing the data collected over various cycles of crawling the website.

- ➔ The data collected is stored in a collection called ‘products’ present under the database called ‘amazon’.

- ➔ The data is stored essentially in a format close to the JSON format.

- ➔ The various attributes of the data element stored are:
  - ◆ pid
  - ◆ name
  - ◆ desc
  - ◆ url
  - ◆ price (as a list)
  - ◆ timestamp (as a list)
  - ◆ rating
  - ◆ reviews

- ➔ To connect to the database a pipeline by the name MongoDbPipeline is created along with certain specific settings.

- ➔ Checks are put into place to ensure duplicates do not find their way to the database.

- ➔ The prices and corresponding timestamps are stored as lists in the collection so as to be easily used to create plots for price analyzing.

- ➔ On subsequent runnings of the crawler, the new prices and timestamps are appended to the list.

On the running the crawler once, a sufficiently large number of URL can be collected from the websites and any change to them on future runnings will be negligible. Hence, in order to optimize running of the crawler in subsequent runs, rather than running a full crawl, only the webpages for the obtained URLs can parsed to obtain the new prices. However once in a while, it will be advised to run the full crawl in order to be collected any new URL created for a product in the time interval.

The running of the crawler at specific intervals can be automated using cron-job.

ANALYSING THE PRICE TRENDS USING THE DATA COLLECTED

- Final part of the application involves a client interface where the user can specify the product URL for the product to be analyzed.

- Based on the product URL, the product ID of a particular product can be generated using which the data related to the product present in the database can be fetched.

- Using this data, the prices and associated timestamps can be found and a nice graph can be plotted showing the price trends to be analyzed.