Review of Sprint #2
Date: Mar 6, 2013
Location: Svpply, 625 Avenue of the Americas - 3rd Floor

Sprint #2 User Stories:

1. Svpply needs a web-crawler that takes as input *image_url, page_url* and produces an estimate of the current price based on proximity of to image_url.
2. Svpply needs a web-crawler that takes as input the *page_url* and using schema.org framework extracts the new price.

Summary of work done

Story # 1:

Tasks performed:
T1. Knowledge acquisition:

- We tried searching tags, like Price in html directly to get the Xpath. But it failed, not every website use Price as their product's price tag.
- We tried searching price by user input price. But we give up this way since we cannot get user input every time. And we cannot ensure user's input is correct every time.
- We tried searchings product's unique id or its name, get from url, to locate detail information about the product. Then tried to search price and status around that area. But it failed, since the website data structure are different, some of them arrange name and image at the very beginning of html, arrange price and status at the middle part of html. We can get all information in the same area. s

T2. Define data extraction strategy:

- Using image_url find XPATH of image.
- Search for currency and currency symbols (USD, $, EURO, RMB) to find out price Xpath.
- Compute distance between the XPATH of the image_url and that of currency symbols found.
- Return a list of prices and their distance to image_url's XPATH location.

T3. Test data extraction and price estimate generation with Scrapy

Initially we found the image_url and then went five DOM levels up and searched for currency.

```
$ scrapy shell http://woodwood.dk/store/product/category-men/m576nga

//Look for the image

>>>
image_locations=hxs.select('//a[@href="http://djte56t8ia8ps.cloudfront.net/client/woodwood/dynamic/images/4329_042d72459e.jpg"]')

>>> image_locations
[<HtmlXPathSelector
xpath='//a[@href="http://djte56t8ia8ps.cloudfront.net/client/woodwood/dynamic/images/4329_042d72459e.jpg"]' data=u'<a href="http://djte56t8ia8ps.cloudfront'>]

//Go five levels up in the position of the image
>>> search_node=image_locations.select('../../../../..')

//Look for anything with EUR and digits

>>> search_node.select('.//text()').re("\d*\s*EUR")
[u'140\xa0EUR']
>>>
```

Results

- This worked as a proof of concept, we still need to find a method of calculating distances between nodes holding currency values and the node which holds the image_url.
- Libxml does provide a way of generating XPATHs of the nodes, we still have explore if libxml is the better tool.

Story #2

Tasks performed:

T1. Knowledge acquisition

T2. Define data extraction strategy:

- Look for “itemprop="price” and extract price.

T3. Test data extraction

$ scrapy shell http://www.urbanoutfitters.com/urban/catalog/productdetail.jsp?id=26536607

//Look for price

>>> hxs.select('//*[@itemprop="price"]').extract()
[u'$320.00'**]**

//Look for product ID if needed

>>> hxs.select('//*[@itemprop="productID"]').extract()
[u' SKU # 26536607 ']

//Look for name using Schema.org framework
>>> hxs.select('//*[@itemprop="name"]').extract()
[u'<h2 itemprop="name" id="prodTitle">Braun Ceramic Watch</h2>']

Results

- Websites that use schema.org framework make very easy data extraction.

Recommendation for Sprint #3:

Continue working on user stories of Sprint #2