Product Cataloging & Intelligence

A Project By

Aditya Gaykar

Saksham Aggarwal

Anil Kumar

(Team 17)

Under the guidance of

Anurag Tyagi

Dr. Vasudeva Verma

Project Report

Problem Statement

Product comparison sites scrape data from various e-commerce websites and provide user with a one-stop pricing details page. The goal is to similarly build a product database, extract vendor pricing information and perform analytics over the data.

Modules

- Master catalogue development (Scrape)
- Product dump extraction
- Vendor dump extraction
- Price dump extraction
- Product pricing sensitivity analysis
- Product master creation
- Intelligence to avoid screen scraping blockages
- HTML GUI query interface along with the required backend.

Overview

- The team will be responsible for developing a scrape module that would generate a product master dump and create an automated process to update the same on a daily basis
- Run web scraper across pre-defined competitor and brand websites to fetch a master list of products across pre-defined categories
- Run web scraper across pre-defined competitor and brand websites to fetch and validate metadata for the final list of products
- The team will receive set of comparison parameters to perform analysis on the scrapped data
- Additional flags for qualitative parameters need to be incorporated in the product metadata
- Add-on flags need to be incorporated in the product metadata for Key Words for which the product listing needs to be displayed in an event of a keyword search
- Add-on flags and logic needs to be incorporated in the product catalogue for Similar products and Related products
- Above is a one-time effort that needs to be done at the start to create the first render of the product catalogue

Applications and Advantages

- An aggregated view of products across e-retailers on the web with detailed comparison on various aspects of the product such as its price, vendor reliability etc can be created
- User would be able to gauge the quality of the product by reviewing its ratings across multiple websites at one place
- Method is applicable across any sellable product such as electronic appliances, clothing etc
- Analytics such as history of vendor rating, product rating, product price would further facilitate the purchase for the user

Challenges

- Many websites have in place a number of different traps to detect and ban web scraping tools and programs. This usually slows down the scraping process, or in some cases, completely block it. We will be looking at various alternatives to avoid this.
- Certain websites make it clear in their terms and conditions that they consider web scraping an infringement of their privacy and might even consider legal redress. Though it is uncommon in e-commerce websites, we still have to go through their content policies to make sure whatever we are doing is legal.
- Scraping each website requires a new scrapper. Also, if the structure of the website is dynamic/changes frequently, we'll have to update/re-code our scrapers again and again.

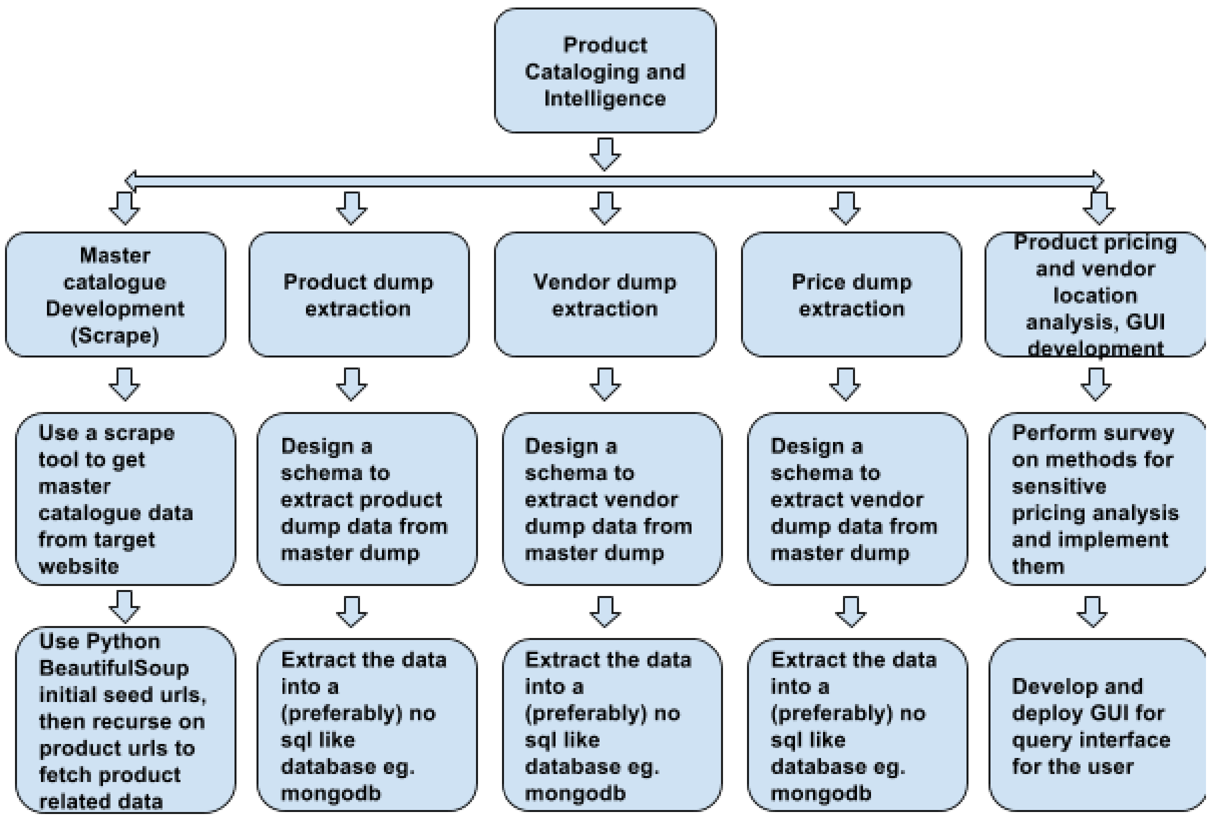
Major Project - Second Deliverables Details

- Master catalogue development (Scrape): The idea is to have one "master" dump that contains all the data, and multiple "slave" dumps such that each slave dump targets a specific functionality. We can then save our scraped data in this master dump and refresh/update it on regular intervals. Any further scraping/extraction will be done on this dump instead of fetching pages again from the internet.
- Product dump extraction: Product data dump would be extracted from the master catalogue dump scraped. This would further facilitate data analysis on product data such as product specifications, category and subcategory etc
- Vendor dump extraction: Product data dump would be extracted from the master catalogue dump scraped. This would further facilitate data analysis on vendor data such as vendor rating history, location, availability
- Price dump extraction: Price data dump would be extracted from the master catalogue dump scraped. This would further facilitate data analysis on price data like the price history for a product

Major Project - Third Deliverables Details

- Product pricing sensitivity analysis: product pricing will be evaluated for fluctuations over time, and data analysis would be performed to predict future changes in pricing for the product
- Product master creation: After all the modules are created, they would be merged to create one product master which will act like a black box to take input query for a certain product as an input and would provide a comprehensive product comparison data for the product
- Intelligence to avoid screen scraping blockages
- Express-Nodejs based GUI query interface: This would act like an end point to the user, where the user would feed in its queries and the the results would be displayed in user friendly GUI.

Task Breakdown Structure



Product Cataloging and Intelligence
Master catalogue Development (Scrape)
Product dump extraction
Vendor dump extraction
Price dump extraction
Product pricing and vendor location analysis, GUI development
Use a scrape tool to get master catalogue data from target website
Design a schema to extract product dump data from master dump
Design a schema to extract vendor dump data from master dump
Design a schema to extract vendor dump data from master dump
Perform survey on methods for sensitive pricing analysis and implement them
Use Python BeautifulSoup initial seed urls, then recurse on product urls to fetch product related data
Extract the data into a (preferably) no sql like database eg. mongodb
Extract the data into a (preferably) no sql like database eg. mongodb
Extract the data into a (preferably) no sql like database eg. mongodb
Develop and deploy GUI for query interface for the user

- **Master Catalogue Development**
    - **Target approach :**
        - A target product website would be finalized and the task of extracting product data from the website would be initialized
        - Website sitemaps would be first extracted from robots.txt file.
        - Per the sitemap, initial seed urls would be generated
        - Available open source scraping tools would be explored on the internet
        - Scraping tools with added dynamic page handling features would be prefered
        - Initial seed url set would be maintained in a queue to fetch webpage for a product, then from the similar/recommended products section on the same page, further seed urls can be feeded back into the queue.
        - Precautions would be taken so that repeated seed urls do not enter the feed queue, thus avoiding infinite looping
        - Data extracted would be dumped into a text file using delimiters for further processing
    - **Tools to be used**
        - Python BeautifulSoup :
    -
            - Beautiful Soup is a Python package for parsing HTML and XML documents (including having malformed markup, i.e. non-closed tags, so named after tag soup).
            - It creates a parse tree for parsed pages that can be used to extract data from HTML, which is useful for web scraping.
            - It is available for Python 2.6+ and Python 3.
            - Beautiful Soup is a Python library for pulling data out of HTML and XML files. It works with your favorite parser to provide idiomatic ways of navigating, searching, and modifying the parse tree. It commonly saves programmers hours or days of work
        - Scrapy :
            - Scrapy is a free and open source web crawling framework, written in Python. Originally designed for web scraping, it can also be used to extract data using APIs or as a general purpose web crawler. It is currently maintained by Scrapinghub Ltd., a web scraping development and services company.
            - Scrapy project architecture is built around ‘spiders’, which are self-contained crawlers which are given a set of instructions. Following the spirit of other don’t repeat yourself frameworks, such as Django, it makes it easier to build and scale large crawling projects by allowing developers to re-use their code. Scrapy also provides a web crawling shell which can be used by developers to test their assumptions on a site’s behavior.
            - Some well-known companies and products using Scrapy are: Lyst, CareerBuilder, Parse.ly, Sciences Po Medialab, Data.gov.uk’s World Government Data site.

- Selenium :
    - Selenium is a set of different software tools each with a different approach to supporting test automation.
    - Most Selenium QA Engineers focus on the one or two tools that most meet the needs of their project, however learning all the tools will give you many different options for approaching different test automation problems.
    - The entire suite of tools results in a rich set of testing functions specifically geared to the needs of testing of web applications of all types.
    - These operations are highly flexible, allowing many options for locating UI elements and comparing expected test results against actual application behavior. One of Selenium's key features is the support for executing one's tests on multiple browser platforms.
    - The biggest change in Selenium recently has been the inclusion of the WebDriver API. Driving a browser natively as a user would either locally or on a remote machine using the Selenium Server it marks a leap forward in terms of browser automation.
    - WebDriver is designed in a simpler and more concise programming interface along with addressing some limitations in the Selenium-RC API.
    - WebDriver is a compact Object Oriented API when compared to Selenium1.0
    - It drives the browser much more effectively and overcomes the limitations of Selenium 1.x which affected our functional test coverage, like the file upload or download, pop-ups and dialogs barrier
    - WebDriver overcomes the limitation of Selenium RC's Single Host origin policy

- **References**

    - Python beautiful soup : http://www.crummy.com/software/BeautifulSoup/
    - https://en.wikipedia.org/wiki/Beautiful_Soup_(HTML_parser)
    - http://www.seleniumhq.org/projects/webdriver/
    - http://stackoverflow.com/questions/17975471/selenium-with-scrapy-for-dynamic-page
    - http://www.seleniumhq.org/docs/01_introducing_selenium.jsp#introducing-selenium

- **Product Dump Extraction**
    - **Target approach**
        - Create a map consisting of each product and URL(s) to its specifications page.
        - Update product dump periodically from master product dump .
        - Make an initial seed list of the specifications needed to be crawled.
        - Crawl the destined product pages for the required specifications .
        - Dump the extracted data in a .csv or some database thus allowing for faster retrievals in the future.
        - Update this periodically.
    - **Tools to be used**
        - MongoDB

            - MongoDB is a cross-platform document-oriented database.Classified as a NoSQL database, MongoDB eschews the traditional table-based relational database structure in favor of JSON-like documents with dynamic schemas(MongoDB calls the format BSON), making the integration of data in certain types of applications easier and faster.
            - MongoDB supports field, range queries, regular expression searches. Queries can return specific fields of documents and also include user-defined JavaScript functions.
            - Any field in a MongoDB document can be indexed – including within arrays and embedded documents (indices in MongoDB are conceptually similar to those in RDBMSes). Primary and secondary indices are available.
            - MongoDB is easy to deploy, and new machines can be added to a running database.

        - Python BeautifulSoup
            - Beautiful Soup is a Python package for parsing HTML and XML documents (including having malformed markup, i.e. non-closed tags, so named after tag soup).
            - It creates a parse tree for parsed pages that can be used to extract data from HTML, which is useful for web scraping.
            - It is available for Python 2.6+ and Python 3.
            - Beautiful Soup is a Python library for pulling data out of HTML and XML files. It works with your favorite parser to provide idiomatic ways of navigating, searching, and modifying the parse tree. It commonly saves programmers hours or days of wor

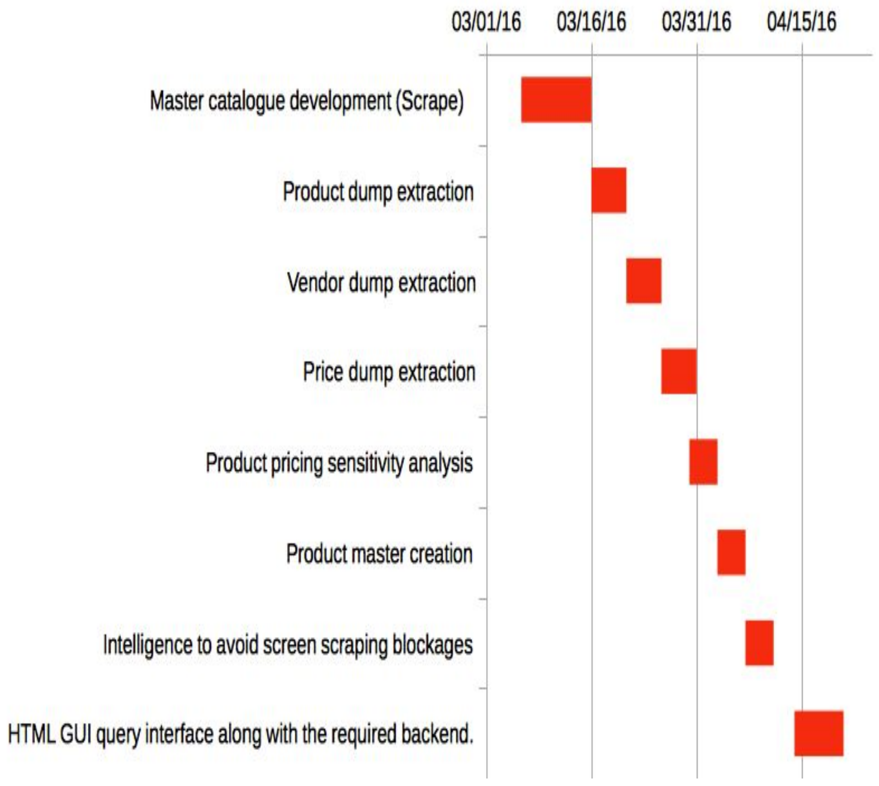
    - **References**
        - Python BeautifulSoup : http://www.crummy.com/software/BeautifulSoup/
        - MongoDB : https://docs.mongodb.org/manual/

- **Vendor Dump Extraction**
    - **Target approach**
        - Create a map with each product to its vendors.It may be possible that some products have only single vendor.
        - Update vendor dump periodically from master product dump in order to get newer addition to vendor’s list.
        - Extract vendor’s information like location,rating inorder to perform analytics on them.
        - Dump the extracted data to some database which allows easier access.
    - **Tools to be used**
        - Python BeautifulSoup
            - Beautiful Soup is a Python package for parsing HTML and XML documents (including having malformed markup, i.e. non-closed tags, so named after tag soup).
            - It creates a parse tree for parsed pages that can be used to extract data from HTML, which is useful for web scraping.
            - It is available for Python 2.6+ and Python 3.
            - Beautiful Soup is a Python library for pulling data out of HTML and XML files. It works with your favorite parser to provide idiomatic ways of navigating, searching, and modifying the parse tree. It commonly saves programmers hours or days of work
        - MongoDB
            - MongoDB is a cross-platform document-oriented database.Classified as a NoSQL database, MongoDB eschews the traditional table-based relational database structure in favor of JSON-like documents with dynamic schemas(MongoDB calls the format BSON), making the integration of data in certain types of applications easier and faster.
            - MongoDB supports field, range queries, regular expression searches. Queries can return specific fields of documents and also include user-defined JavaScript functions.
            - Any field in a MongoDB document can be indexed – including within arrays and embedded documents (indices in MongoDB are conceptually similar to those in RDBMSes). Primary and secondary indices are available.
            - MongoDB is easy to deploy, and new machines can be added to a running database.
    - **References**
        - Python BeautifulSoup : http://www.crummy.com/software/BeautifulSoup/
        - MongoDB : https://docs.mongodb.org/manual/

- **Price Dump Extraction**
  - **Target approach**
    - Create a map to each product to its price according to vendor.For single vendor products only one price may exist.
    - Update price dump periodically form master product dump inorder to get vendor specific latest price.
    - Extract price and compare among different vendors in order to perform analytics on them.
    - Dump the extracted data to some database which allows easier access.
  - **Tools to be used**
    - Python BeautifulSoup
      - Beautiful Soup is a Python package for parsing HTML and XML documents (including having malformed markup, i.e. non-closed tags, so named after tag soup).
      - It creates a parse tree for parsed pages that can be used to extract data from HTML, which is useful for web scraping.
      - It is available for Python 2.6+ and Python 3.
      - Beautiful Soup is a Python library for pulling data out of HTML and XML files. It works with your favorite parser to provide idiomatic ways of navigating, searching, and modifying the parse tree. It commonly saves programmers hours or days of work
    - MongoDB
      - MongoDB is a cross-platform document-oriented database.Classified as a NoSQL database, MongoDB eschews the traditional table-based relational database structure in favor of JSON-like documents with dynamic schemas(MongoDB calls the format BSON), making the integration of data in certain types of applications easier and faster.
      - MongoDB supports field, range queries, regular expression searches. Queries can return specific fields of documents and also include user-defined JavaScript functions.
      - Any field in a MongoDB document can be indexed – including within arrays and embedded documents (indices in MongoDB are conceptually similar to those in RDBMSes). Primary and secondary indices are available.
      - MongoDB is easy to deploy, and new machines can be added to a running database.
  - **References**
    - Python BeautifulSoup : http://www.crummy.com/software/BeautifulSoup/
    - MongoDB : https://docs.mongodb.org/manual/

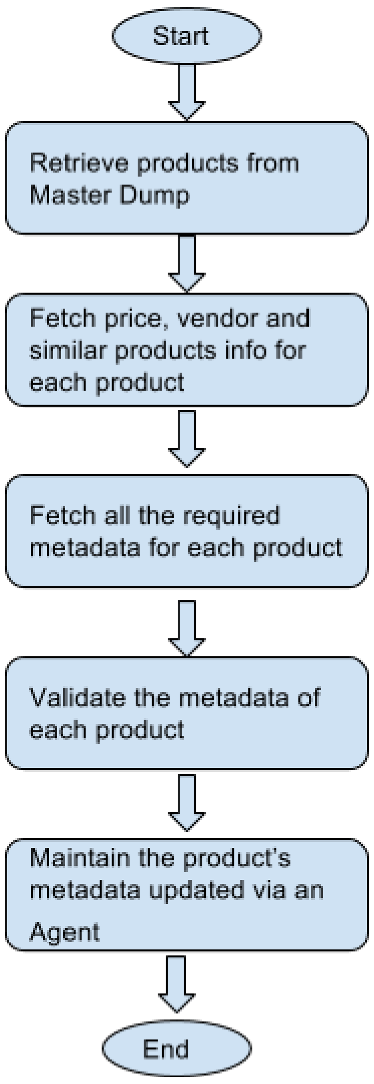
- **Vendor location and product price based analytics and GUI development**
    - **Target approach**
        - Extract location of all vendors across products.
        - Using this we can observe what product are readily available in which region and which products aren’t
        - We can also see for a specific product, where it is common/popular/easily available using clustering.
        - Display the above information on a map-like structure using tools such or Google Maps, Open Street Maps, Leaflet, Polymaps
        - See how different vendors set different prices for the same product.
        - Observe the change in prices for all products over time
        - Display the above in a histogram
        - Observation of this histogram enables to see product trends over time
        - All of this calculated data will be saved in a structured database and is refreshed periodically based on the freshness of the respective parent data dump.
        - A web based GUI will be developed to display the above information in a Human friendly fashion
        - We plan to use a client-server model.
        - The server fetches the required data from the database and client just displays it in the web browser.
        - We also plan to use additional JS plugins for Histograms and maps.
    - **Tools to be used**
        - Django
            - Django is a free and open-source web framework, written in Python, which follows the model–view–controller (MVC) architectural pattern.
            - Django's primary goal is to ease the creation of complex, database-driven websites
            - a lightweight and standalone web server for development and testing
        - Leaflet MarkerCluster
            - Provides Beautiful Animated Marker Clustering functionality for Leaflet, a JS library for interactive maps.
            - Open Source and available for free on github
            - Uses the Free open source LeafletJS library as base
        - Plotly.js
            - plotly.js is a high-level, declarative charting library in Javascript
            - Will be used to plot Histogram/Bar charts
    - **References**
        - PolyMaps: http://polymaps.org/ex/cluster.html
        - LeafletJS: http://leafletjs.com/
        - Google Maps API: https://developers.google.com/maps/
        - Django Framework: https://www.djangoproject.com/
        - Leaflet MarkerCluster: https://www.mapbox.com/mapbox.js/example/v1.0.0/leaflet-markercluster/
        - https://github.com/Leaflet/Leaflet.markercluster
        - Plotly.js: https://plot.ly/javascrip

Timeline


03/01/16
03/16/16
03/31/16
04/15/16
Master catalogue development (Scrape)
Product dump extraction
Vendor dump extraction
Price dump extraction
Product pricing sensitivity analysis
Product master creation
Intelligence to avoid screen scraping blockages
HTML GUI query interface along with the required backend.

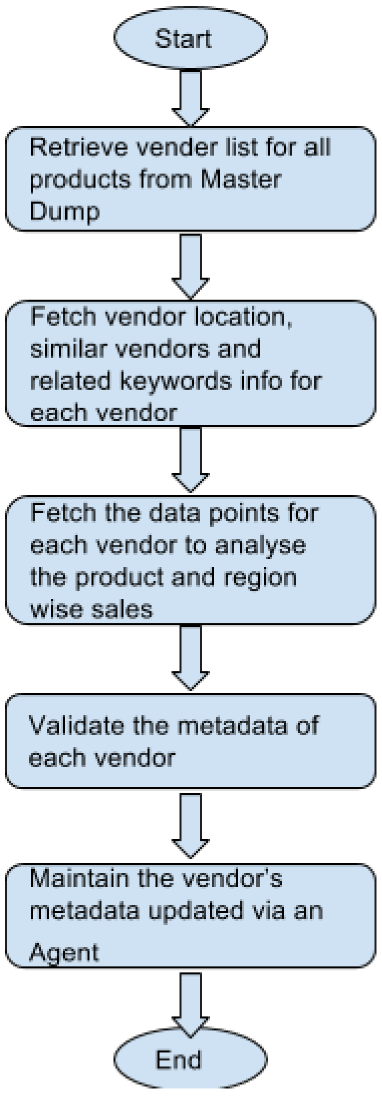
Some important flow charts

Product Master

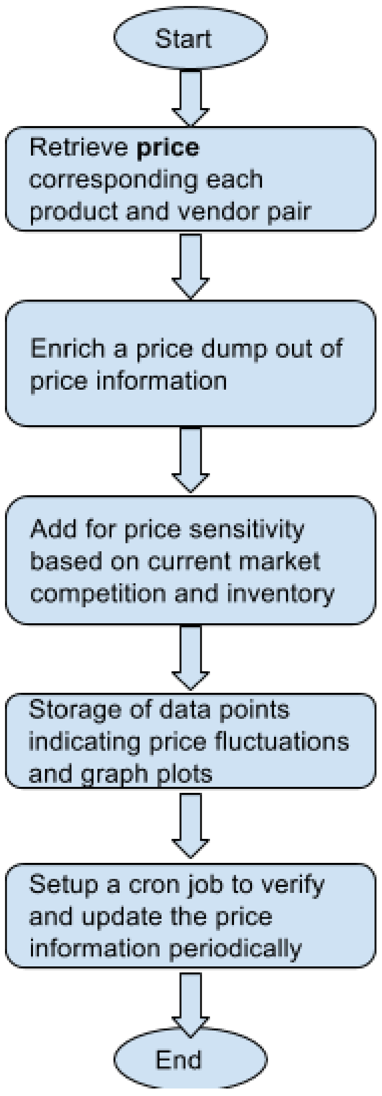


Start
Retrieve products from Master Dump
Fetch price, vendor and similar products info for each product
Fetch all the required metadata for each product
Validate the metadata of each product
Maintain the product's metadata updated via an Agent
End

Vendor Master


Start
Retrieve vender list for all products from Master Dump
Fetch vendor location, similar vendors and related keywords info for each vendor
Fetch the data points for each vendor to analyse the product and region wise sales
Validate the metadata of each vendor
Maintain the vendor's metadata updated via an Agent
End

Price Master



Start
Retrieve **price** corresponding each product and vendor pair
Enrich a price dump out of price information
Add for price sensitivity based on current market competition and inventory
Storage of data points indicating price fluctuations and graph plots
Setup a cron job to verify and update the price information periodically
End

Few important tools

- Scraper: This will be used to scrape product related data from different e-commerce websites. We are looking at various scraping techniques and which would suit us best. Automated tools such as **Scrapy, BeautifulSoup, Selenium** already have a lot of plumbing but require a specific structure and may be slow. Regex based scrapers like **raw PHP or Python** on the other hand are super fast but we'll have to consider a lot of extra things.
- Web server: This will be used to make a web interface to present a simple query engine to search for different products. Every search would provide a comprehensive comparison as per the data scraped from e-commerce websites. Since this is not our main job, we are looking at simple frameworks such as **Web2Py**, **Django** and **PHP on Apache**.
- Rapidminer/Weka : Tools to perform analytics on scraped data

Implementation

Scrapper:

- **Python scraper**
  - The entire scraper is written in python. Whenever we need to run the scraper, we command the nodejs backend to run the corresponding python script.
- **Python - dryscraper**
  - Dryscraper is a headless browser in python.
- **Python - reppy**
  - Reppy is a robots.txt parser
- **Python - pymongo**
  - Pymongo is a Python package that provides an API to interact with MongoDB

General Approach:

- We have a base class that handles most of the challenges of scraping and leaves only minimal work to the user.
- For every e-commerce site scraper, we need to inherit from this base class and override 3 functions.
  - next() -> Return the URL of the next product catalogue page
  - products(page) -> Given the beautifulSoup object of a product catalogue page, extract all product URLs from the page and return them
  - process(page) -> Given the beautifulsoup object of product page, extract all the information and do whatever you want to.

Key Challenges and Solution:

- Scrape sites that do not allow bots
  - We thought of mimicking the browser.
  - Earlier prototypes used Firefox User Agent string.
  - Later we moved to **dryscraper** which is a headless browser.
- Handle dynamic websites
  - Some websites load the product based on JavaScript or AJAX. As a result fetching the static HTML will yield incomplete/incorrect results.
  - The best way to do this is to load the complete page and let the JavaScript run. **Dryscraper** does this job and gives the modified page as output.
- Don’t get blocked by the target website - Crawl-delay.
  - To avoid bot overload on a site, site administrators set a minimum wait time between 2 consecutive requests. This minimum wait time is called Crawl-delay and is mentioned in a robots file, **robots.txt**.
  - The idea is to download this file, extract the information and put this minimum wait time constraint. We used **reppy** which is a parser for robots.txt file.
- Don’t get blocked by the target website - Follow Disallow rules.
  - Some sites don’t want bots to scrape specific parts of their website, either for search engine optimization or simple content privacy.
  - This is also mentioned in the **robots.txt** file and similar to the previous part, we use **reppy** to find this out.
- Scraping the entire website is slow!
  - We solve this problem by changing the design of our scrape process.
  - We never scrape the complete website! (Except for once when our dump is completely empty)

- We support 2 operations.
    - Look for and scrape new products only.
    - Update a single product.
- We leveraged our domain knowledge here, to optimize the process without giving up on accuracy or dump freshness.
- We hardly see a lot of new products being added at once, which makes the first process fast.
- Also, not too many products are updated everyday. Only a very small fraction of products are updated each day. As a result the second subtask needs to be run only on those products and hence achieves speed.

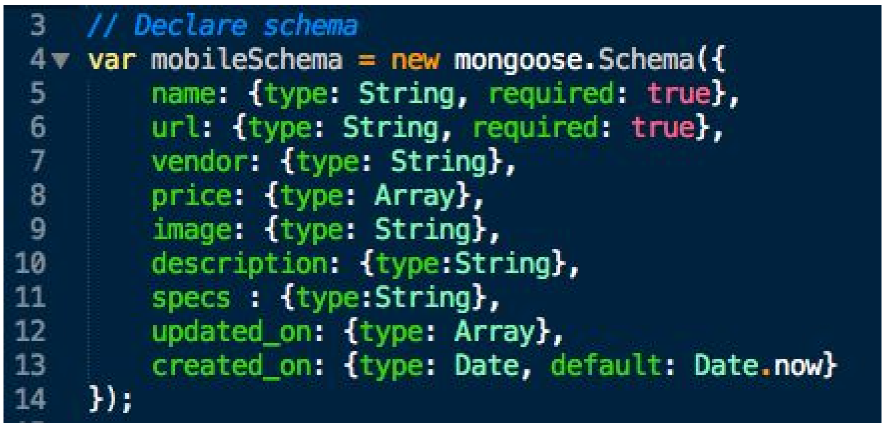
Web Application

Web application is implemented with following technologies :

- Node.js server
  - Core backend server for hosting the application
- MongoDB server
  - Database backend
- Mongoose MongoDB client for node
  - Node wrapper to connect to mongodb
- Twitter Bootstrap UI framework
  - Highly extensive web css framework from twitter
- Express MVC framework
  - Web MVC framework for nodejs
- Jade template engine
  - Templating engine plugin for nodejs
- Google Charts API
  - Chart plotting api from Google
- jQuery
  - Popular javascript library for DOM operations

MongoDB Database server configurations:

- MongoDB provides nosql like features which are best fit for this project data implementations
- Key challenges faced are that the data would be populating time and again with new values, so we had to think of an intelligent mechanism to populate all the updates.
- Following is the schema that we used to populate the master dump :



```
// Declare schema
var mobileSchema = new mongoose.Schema({
    name: {type: String, required: true},
    url: {type: String, required: true},
    vendor: {type: String},
    price: {type: Array},
    image: {type: String},
    description: {type:String},
    specs : {type:String},
    updated_on: {type: Array},
    created_on: {type: Date, default: Date.now}
});
```

- price and updated_on on attributes have been assigned to javascript array objects. This enables us to populate price varying on various updates in time
- So next time there is an upload by the scrapper for the same product, a new entire is not created, rather the existing entry is updated with new values for price and updated_on pushed in the respective arrays
- Given this model it is simple to retrieve the latest value by just popping the last value from the stack

Features of web application

- Web application enables the user to navigate, query scrapped data in a well defined user interface
- Interfaces include, search interface to search products by name, vendor and specification

- Search feature is implemented using jQuery ajax calls.

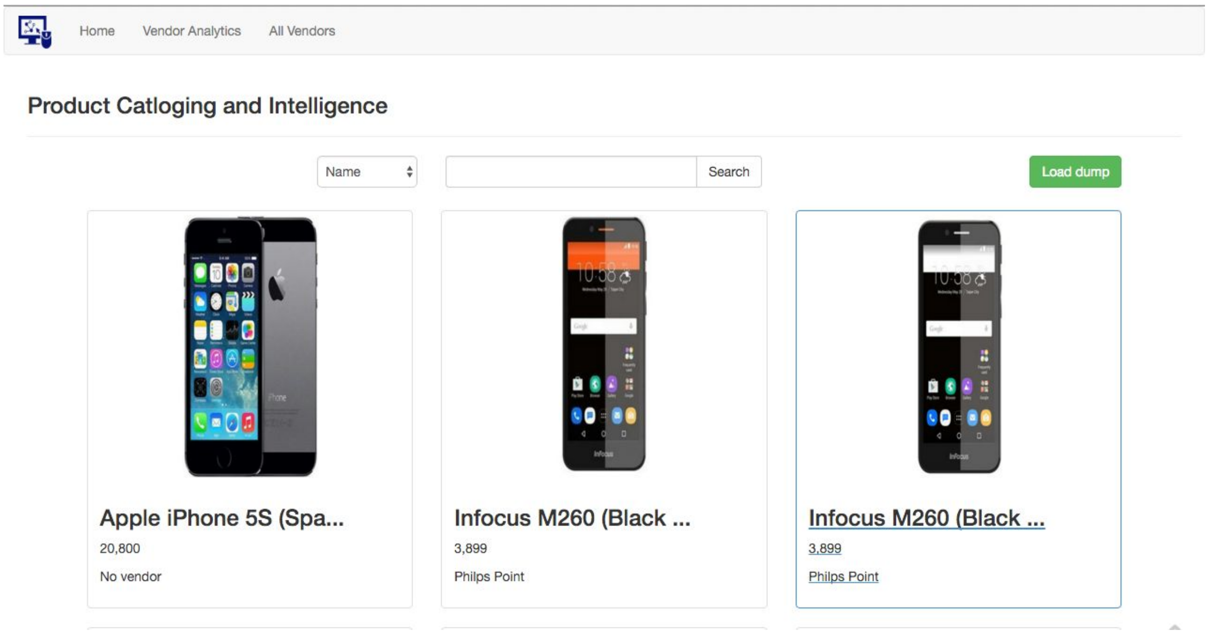
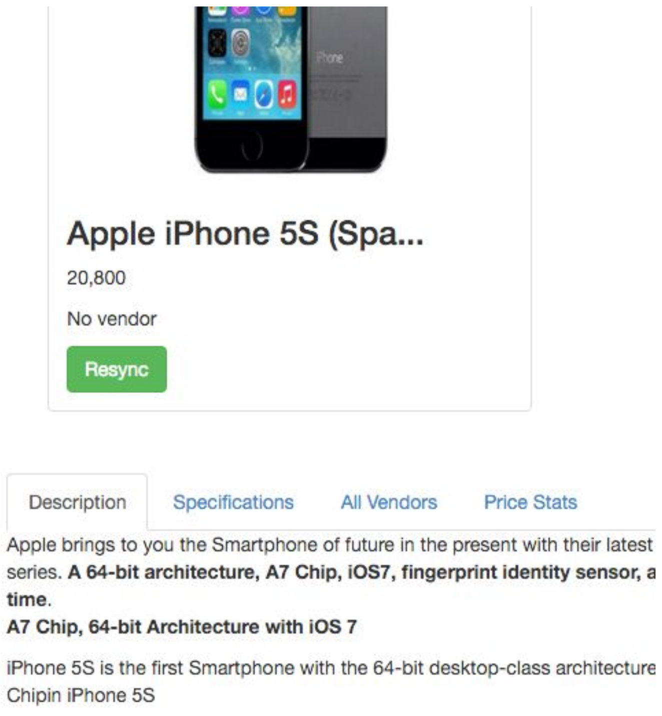


Fig. Main landing page

- Product view interface to see more information about a particular product
- Description
- Specifications
- All Vendors
- Price stats
- Resync feature is an important feature to have for the user, which when clicked initiates a dry scrape script to fetch in the latest information for that particular product
- This information is logged into the mongodb database can be verified in Price stats section



Apple brings to you the Smartphone of future in the present with their latest series. **A 64-bit architecture, A7 Chip, iOS7, fingerprint identity sensor, a time.**

A7 Chip, 64-bit Architecture with iOS 7

iPhone 5S is the first Smartphone with the 64-bit desktop-class architecture Chipin iPhone 5S

Fig: Product View

Price statistics

- Under product view we have a section for price statistics to show the variations in price for that particular product
- As we can see in the image various price variations are synced and displayed in a bar chart

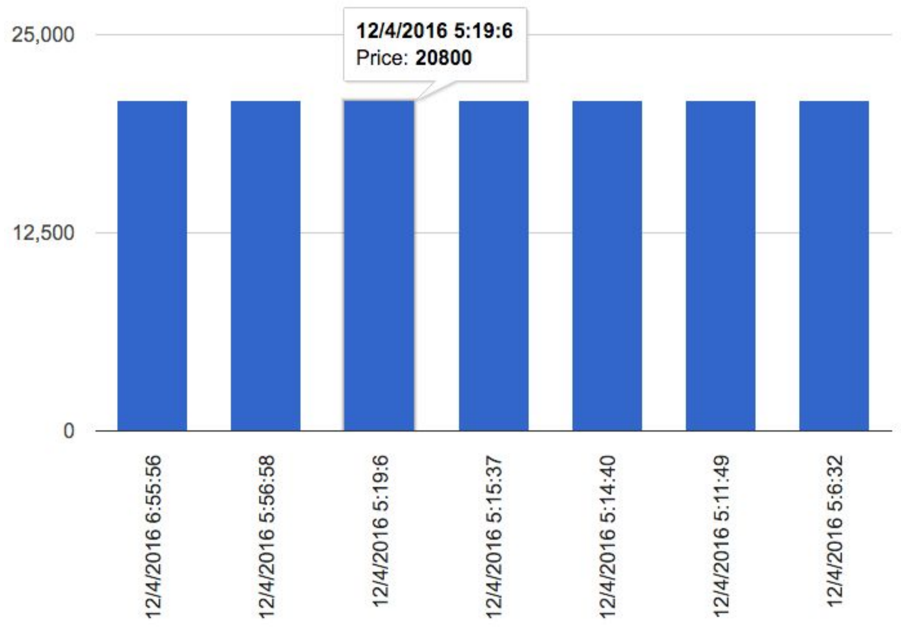
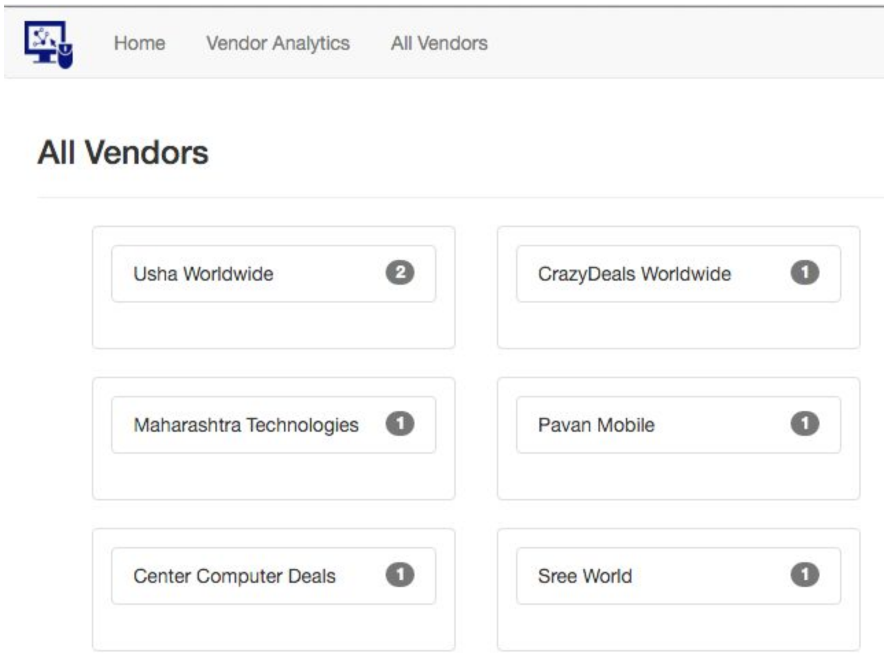


Fig. Price statistics for each product

All Vendors

- All vendors page, lists the total number of vendors that were scrapped from the website
- Also the number of products sold by each vendor is shown inside a badge
- On click of a vendor badge, the user is redirected to the list of products sold by that particular vendor



All Vendors

Usha Worldwide 2

CrazyDeals Worldwide 1

Maharashtra Technologies 1

Pavan Mobile 1

Center Computer Deals 1

Sree World 1

Fig. All vendors page

Vendor Analytics

- Vendor Analytics page, displays a bar chart representation of vendor data i.e Vendor vs Number of products sold by that vendor

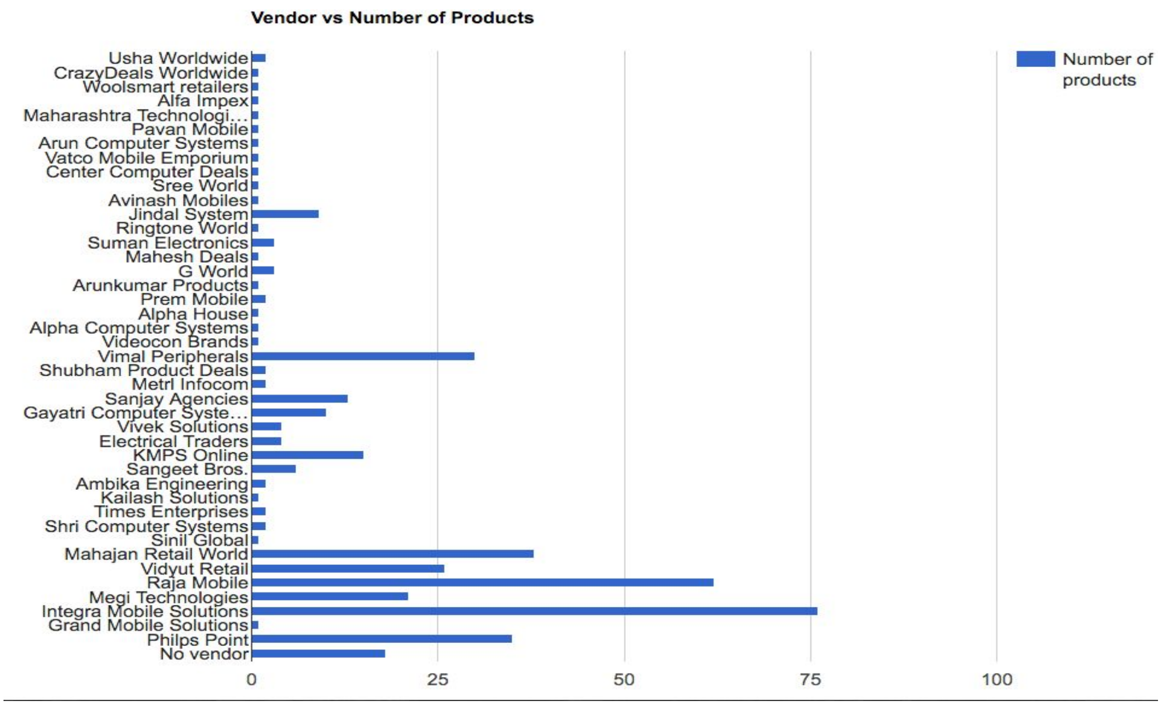


Fig. Vendor Analytics

Conclusion

- This project helped us learn a lot about scraping technologies, how to follow robots.txt specifications for a website to perform ethical scrapping.
- Scraping and web app system implemented gives an insightful display of products present on a website, the way their prices vary, new vendors joining the website, products sold by each vendor and so on.
- Resync feature further enhances the scrapping feature, by doing limited scrapping for each product rather syncing for all the products on the website that would be time consuming.

References

- Python beautiful soup : http://www.crummy.com/software/BeautifulSoup/
- https://en.wikipedia.org/wiki/Beautiful_Soup_(HTML_parser)
- http://www.seleniumhq.org/projects/webdriver/
- http://stackoverflow.com/questions/17975471/selenium-with-scrapy-for-dynamic-page
- PolyMaps: http://polymaps.org/ex/cluster.html
- LeafletJS: http://leafletjs.com/
- Google Maps API: https://developers.google.com/maps/
- Django Framework: https://www.djangoproject.com/
- Leaflet MarkerCluster: https://www.mapbox.com/mapbox.js/example/v1.0.0/leaflet-markercluster/
- https://github.com/Leaflet/Leaflet.markercluster