Software Documentation

For

Web Crawler for E-commerce Websites-IDP

Version 1.0

Prepared by Mohamad Ayad

TUM School of Management
Chair for Strategy and Organization

January 31, 2017

Table of Contents

Revision History

Version	Date	Name	Description
1	25-1-2017	Mohamad Ayad	Initial Document

1. Introduction

1.1 Purpose

The E-commerce web crawler is a web application that is developed to extract products information from e-commerce websites. The crawler analyzes the name, image, price, description and reviews for each product. Unlike other available tools such as SEO which is the process of affecting the visibility of a website in a web search en.

2. System Overview

2.1 Requirements

- Python 3.5+
- Django framework version 1.8.14

  high-level Python Web framework that encourages rapid development and clean, pragmatic design
- Beautifulsoup4

  Python library for pulling data out of HTML

2.2 Installation Instructions

2.2.1 Python Installation

Windows

Python 3.5+ Windows installer (Windows binary)
or
Python 3.5+ Windows AMD64 installer (Windows AMD64 binary)

Linux or Mac OS X

```
brew install python
```

2.2.2 Django Installation

pip install Django= =1.8.14

2.2.3 Beautifulsoup4 Installation

pip install beautifulsoup4

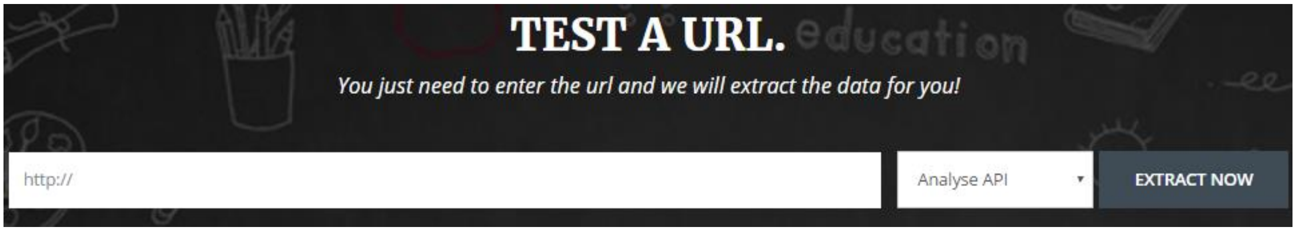
2.3 System Characteristics

- Scalable and easily maintainable in the future.
- Special back-up facilities to protect important data.
- Highly fault tolerant.

3. User and Technical Documentation

3.1 User Documentation

For the end-users the GUI of the web application made it very friendly and easy to use. First the user enters the link of the product inside the text box, then press the extract now button and the data will be extracted and visualized.



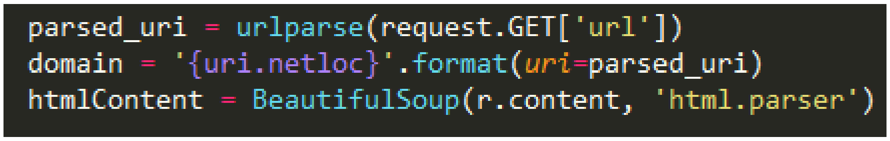
TEST A URL.

You just need to enter the url and we will extract the data for you!

3.2 Technical Documentation

3.2.1 Algorithm

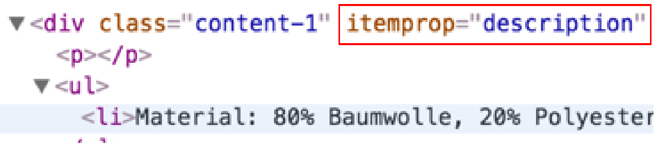
First of all, we need to parse the html of the webpage to get the html and it's all tags using that's why we are using Beautifulsoup.



```
parsed_uri = urlparse(request.GET['url'])
domain = '{uri.netloc}'.format(uri=parsed_uri)
htmlContent = BeautifulSoup(r.content, 'html.parser')
```

After parsing the html, the next step is to look for some patterns that may help us to find the information we need to extract. These patterns could be:
-**Metadata**: Up to 90% of the websites use metadata to store the title and the image of the product.

-**Itemprop:** is a global attribute used to add properties to an item. Every HTML element can have an itemprop attribute specified, and an itemprop consists of a name-value pair. Also up to 90% of the websites have this attribute for "price" and "priceCurrency".

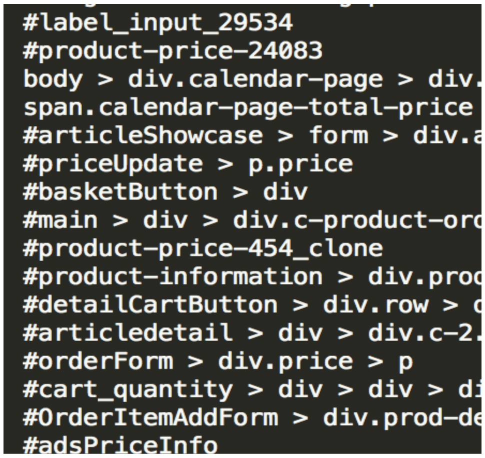


```
▼<div class="content-1" itemprop="description"
   <p></p>
  ▼<ul>
     <li>Material: 80% Baumwolle, 20% Polyester
```

-Tag id or class: search for ids or classes that match our target. As an example if we are looking for the price, we may find some ids or classes named "price, product-price, …."

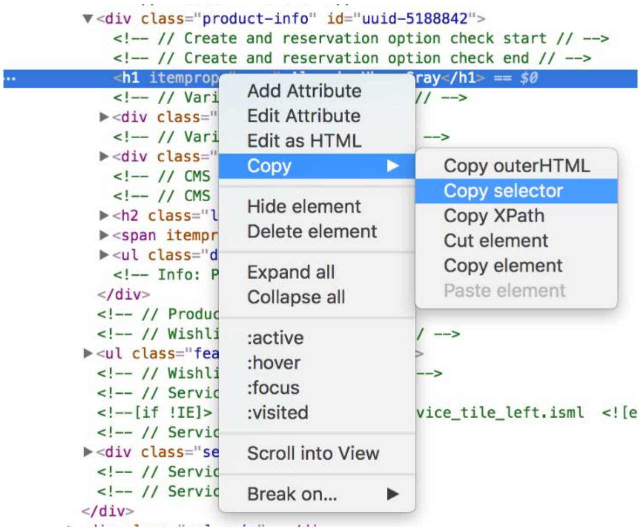
In order to increase the accuracy of the software, we have used some dictionaries to speed up the search and filter the results. (Dictionary for currency, ids and classes, for keywords…)

Also, we have added the static crawler feature based on "CSS Selectors". If the crawler didn't give the desired result, the CSS Selector of the missing information can be added to the specified attribute dictionary. In order to get this selector, in the browser right click on the information then inspect element, copy the selector and add it to the dictionary (e.g. For the price , the selector is added to priceDictionary). The more selectors added to the dictionary, the higher the accuracy is.



```
#label_input_29534
#product-price-24083
body > div.calendar-page > div.
span.calendar-page-total-price
#articleShowcase > form > div.a
#priceUpdate > p.price
#basketButton > div
#main > div > div.c-product-orc
#product-price-454_clone
#product-information > div.proc
#detailCartButton > div.row > c
#articledetail > div > div.c-2.
#orderForm > div.price > p
#cart_quantity > div > div > di
#OrderItemAddForm > div.prod-de
#adsPriceInfo
```

Dictionary of price selectors



```
▼<div class="product-info" id="uuid-5188842">
   <!-- // Create and reservation option check start // -->
   <!-- // Create and reservation option check end // -->
   <h1 itemprop                    ray</h1> == $0
   <!-- // Vari                      // -->
  ▶<div class="
   <!-- // Vari                      -->
  ▶<div class="
   <!-- // CMS
   <!-- // CMS
  ▶<h2 class="l
  ▶<span itempr
  ▶<ul class="d
   <!-- Info: P
  </div>
  <!-- // Produc
  <!-- // Wishli                     / -->
 ▶<ul class="fea                     >
  <!-- // Wishli                     -->
  <!-- // Servic
  <!--[if !IE]>                      vice_tile_left.isml  <![e
  <!-- // Servic
 ▶<div class="se
  <!-- // Servic
  <!-- // Servic
</div>
```

Copy the selector

3.2.2 Main functions

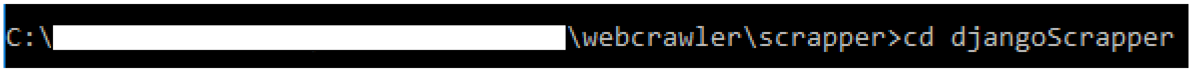
-getPrice() , getImage() , getTitle(), getDescription(), getReviews().

-filterArray(): This function filters the array of ids and classes and returns a new list of only the needed ids and classes that are used to extract the product information.

3.2.3 Run the application

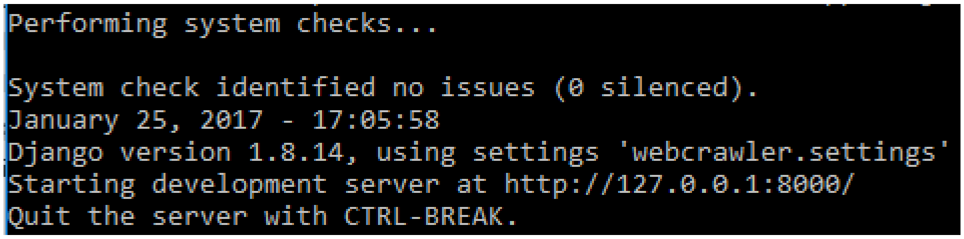
From the terminal or the command prompt

navigate to project folder/scrapper/djangoScrapper



```
C:\[redacted]\webcrawler\scrapper>cd djangoScrapper
```

Then execute this command: python manage.py runserver

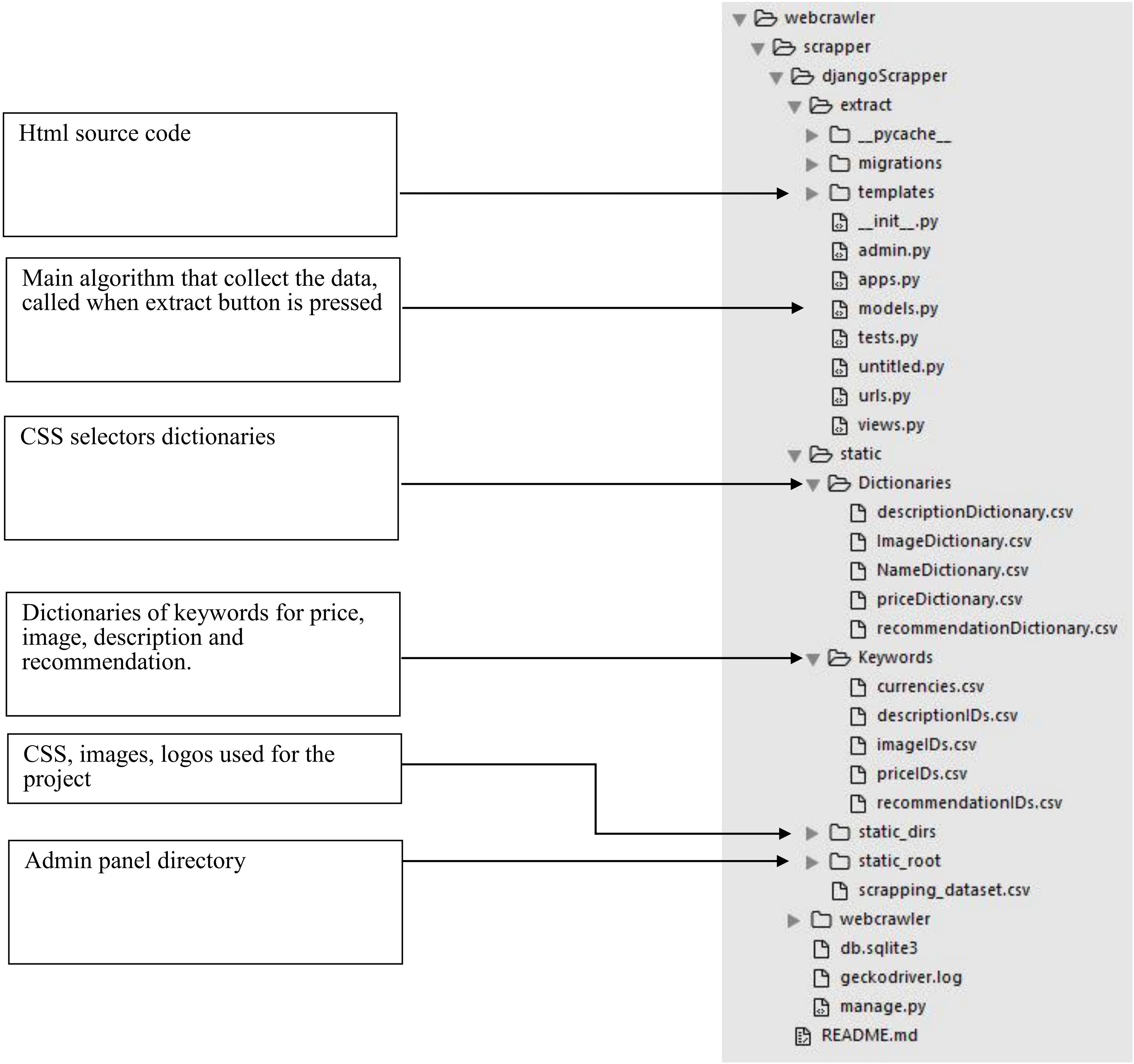


```
Performing system checks...

System check identified no issues (0 silenced).
January 25, 2017 - 17:05:58
Django version 1.8.14, using settings 'webcrawler.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

go to the browser and enter the following link
http://127.0.0.1:8000/extract/

3.2.4 Project Structure



Html source code

Main algorithm that collect the data, called when extract button is pressed

CSS selectors dictionaries

Dictionaries of keywords for price, image, description and recommendation.

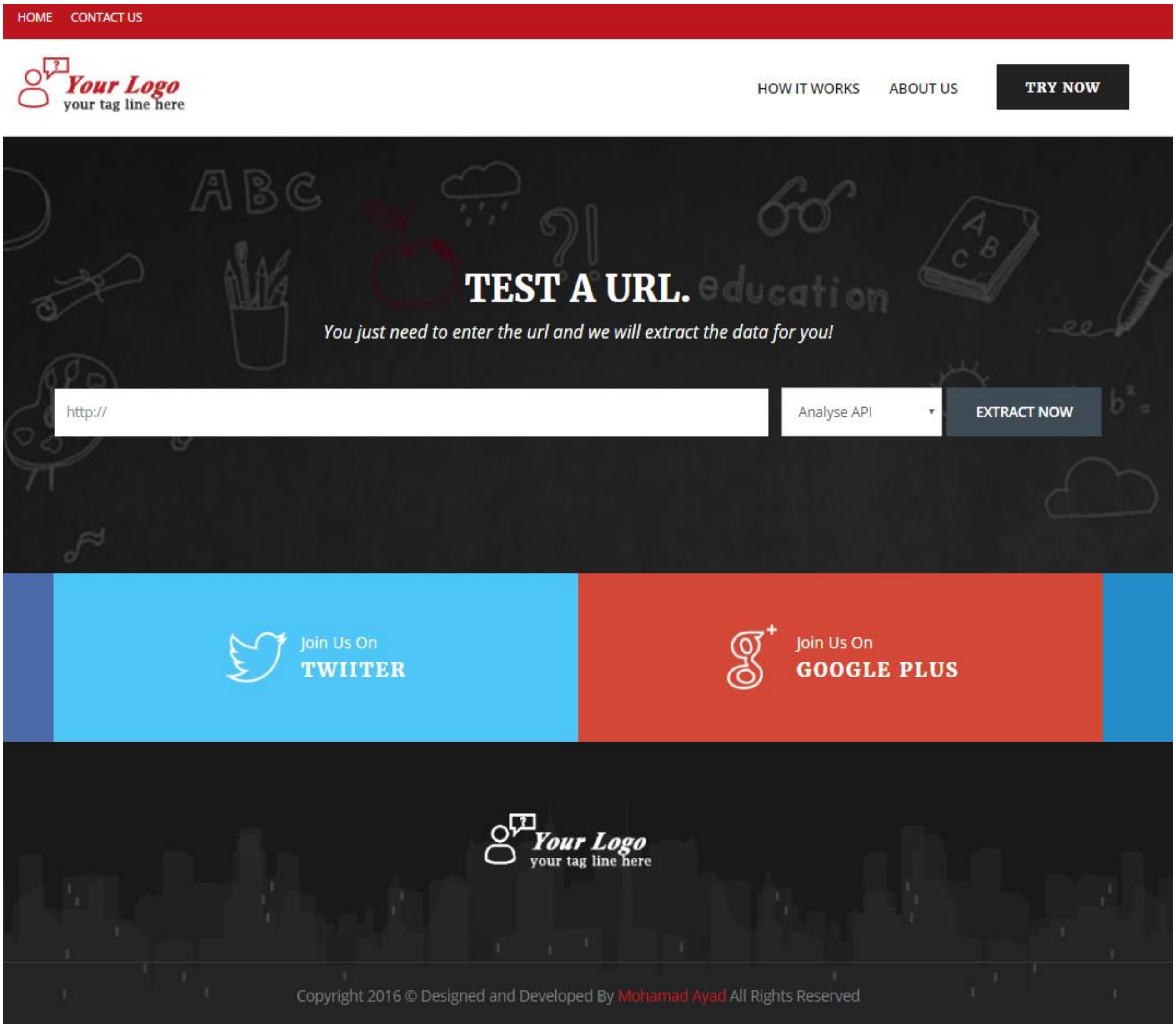
CSS, images, logos used for the project

Admin panel directory

```
▼ webcrawler
  ▼ scrapper
    ▼ djangoScrapper
      ▼ extract
        ▶ __pycache__
        ▶ migrations
        ▶ templates
          __init__.py
          admin.py
          apps.py
          models.py
          tests.py
          untitled.py
          urls.py
          views.py
      ▼ static
        ▼ Dictionaries
            descriptionDictionary.csv
            ImageDictionary.csv
            NameDictionary.csv
            priceDictionary.csv
            recommendationDictionary.csv
        ▼ Keywords
            currencies.csv
            descriptionIDs.csv
            imageIDs.csv
            priceIDs.csv
            recommendationIDs.csv
        ▶ static_dirs
        ▶ static_root
          scrapping_dataset.csv
      ▶ webcrawler
        db.sqlite3
        geckodriver.log
        manage.py
      README.md
```

4. Sample Run

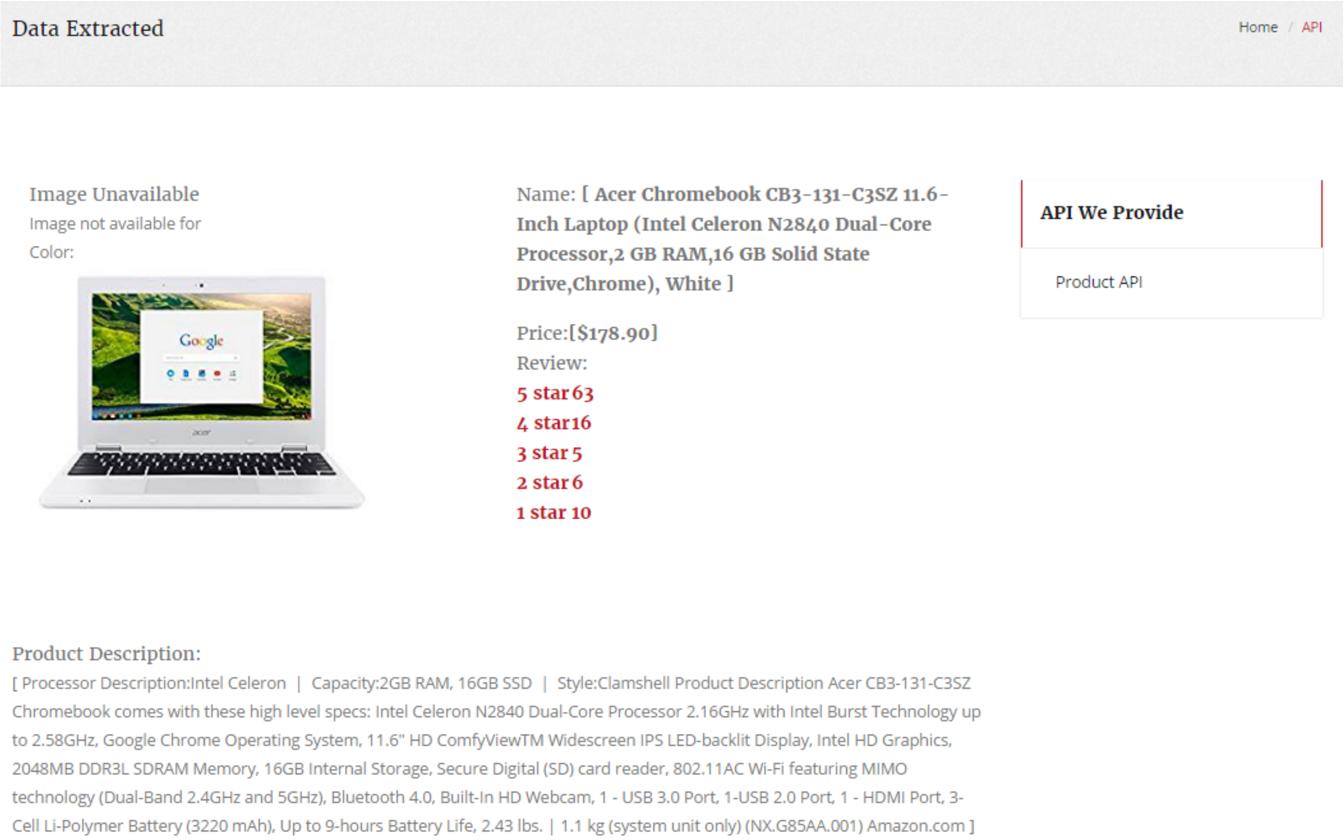
Below are the screenshots of visualizing the data for the end-user.
product: https://www.amazon.com/Acer-Chromebook-CB3-131-C3SZ-11-6-Inch-Dual-Core/dp/B019G7VPTC/ref=sr_1_5?s=pc&ie=UTF8&qid=1485360730&sr=1-5&keywords=laptop



TEST A URL.

You just need to enter the url and we will extract the data for you!

Copyright 2016 © Designed and Developed By Mohamad Ayad All Rights Reserved



Data Extracted

Image Unavailable

Image not available for Color:

Name: **[Acer Chromebook CB3-131-C3SZ 11.6-Inch Laptop (Intel Celeron N2840 Dual-Core Processor,2 GB RAM,16 GB Solid State Drive,Chrome), White]**

Price:[$178.90]

Review:

5 star 63

4 star 16

3 star 5

2 star 6

1 star 10

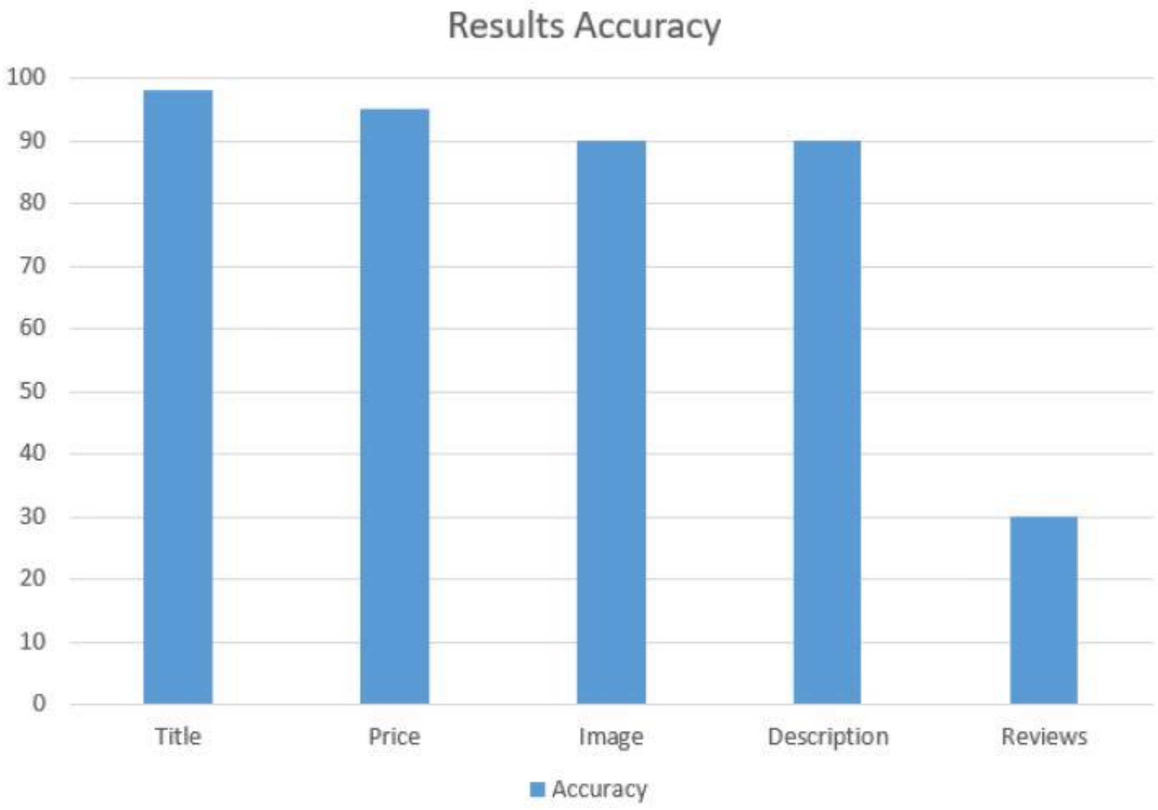
API We Provide

Product API

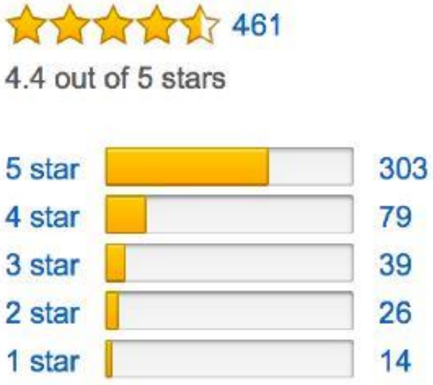
Product Description:

[Processor Description:Intel Celeron | Capacity:2GB RAM, 16GB SSD | Style:Clamshell Product Description Acer CB3-131-C3SZ Chromebook comes with these high level specs: Intel Celeron N2840 Dual-Core Processor 2.16GHz with Intel Burst Technology up to 2.58GHz, Google Chrome Operating System, 11.6" HD ComfyViewTM Widescreen IPS LED-backlit Display, Intel HD Graphics, 2048MB DDR3L SDRAM Memory, 16GB Internal Storage, Secure Digital (SD) card reader, 802.11AC Wi-Fi featuring MIMO technology (Dual-Band 2.4GHz and 5GHz), Bluetooth 4.0, Built-In HD Webcam, 1 - USB 3.0 Port, 1-USB 2.0 Port, 1 - HDMI Port, 3-Cell Li-Polymer Battery (3220 mAh), Up to 9-hours Battery Life, 2.43 lbs. | 1.1 kg (system unit only) (NX.G85AA.001) Amazon.com]

5. Results and Accuracy



For the reviews, the accuracy is very low because the websites that have statistics for customer ratings are very rare.



461

4.4 out of 5 stars

5 star	303
4 star	79
3 star	39
2 star	26
1 star	14

6. Future Work

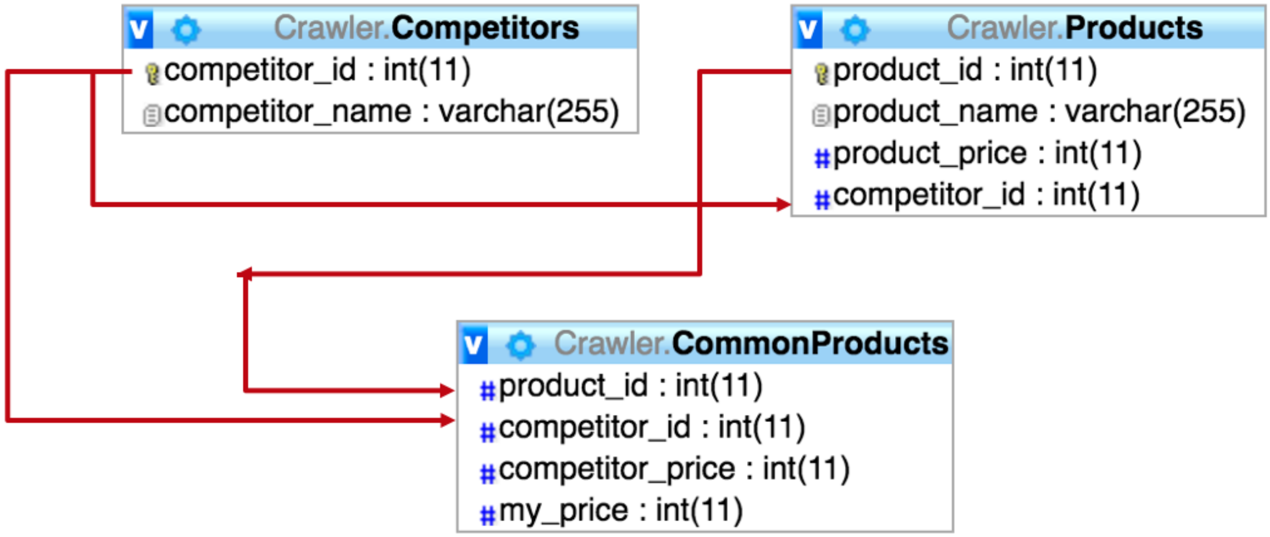
6.1 Machine Learning

Extending this project to a machine learning approach will lead to higher accuracy. Unfortunately, to be able to use machine learning a dataset of 100,000 entries approximately is needed which is currently not available. In order to label such a dataset, we need about 5000 hours. After labeling classification algorithm will fit this task by using some training models.

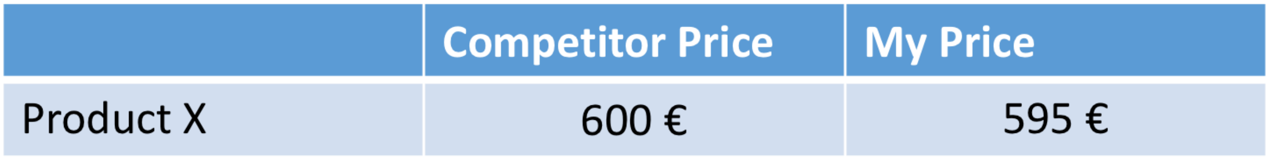
6.2 Track Competitors

Can be extended to track and trace user's competitors' prices and to store them in a database with by automating the crawler which can update this database weekly. Also by automating a feedback process by implementing a new filtering algorithm, the user will be able to get a feedback if his prices are very high or very low compared to competitors

6.2.1 Database Structure

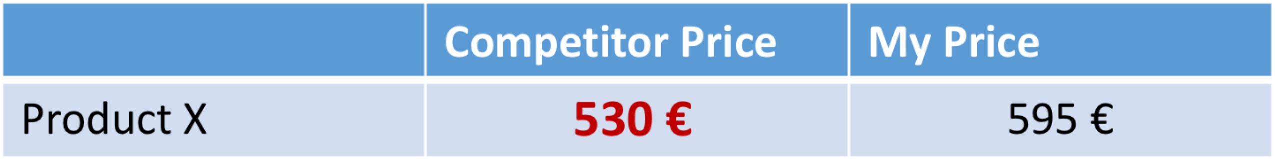


6.2.2 Feedback Example



	Competitor Price	My Price
Product X	600 €	595 €

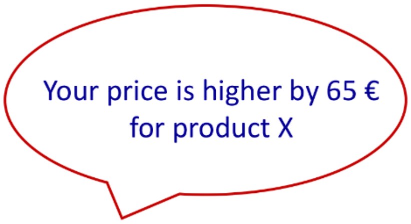
Initial Crawling results



	Competitor Price	My Price
Product X	**530 €**	595 €

Updated Results (After 1 week)

In this example, the competitor price of product X was 600 EUR and after 1 week the competitor has changed the price to 530 EUR. The automatic feedback script will notify the user that his price is now higher than his competitor's by amount X.



7. References

BeautifulSoup Documentation

- https://www.crummy.com/software/BeautifulSoup/bs4/doc/

Django Framework Documentation

- https://docs.djangoproject.com/en/1.10/

Python 3 Documentation

- https://docs.python.org/3/