Web Science

Comparing recommendations in Bol.com and DeSlegte.com

Monday, June 6th, 2022

Version 4.1

Tom Denis (670940)

Robin Tollenaar (672023)

Marieke Morganwiese (670515)

Index

[**Introduction**](#_gx2gear87seh) **3**

[**Context**](#_qyhtpjn8e7qu) **3**

[Main research question](#_5osqsmdw6wm3) 3

[Sub questions](#_27fkypaujorz) 3

[**Methodology**](#_gr5kiag37ub7) **3**

[Research Strategy](#_wb0n9qk9v3ru) 3

[Research Design](#_5l4do0v9j6zp) 4

[**Results and analysis**](#_25w49nu8lr2e) **4**

[How will data be acquired from the websites using scraping tools?](#_w0leq4ogedlw) 4

[Deciding on a scraping tool.](#_y0bmk7xdf0f6) 4

[Using the scraping tool to obtain useful data](#_nbridajms7cb) 5

[Scraping with Scrapy](#_ahi1kcuzspq3) 5

[The data](#_tmx8xzx7osph) 8

[What graph theory methods can be applied in this situation?](#_sbzx3gnnwlv2) 9

[What do the recommendation graphs say about the websites?](#_e9al9nfgzhxp) 10

[Deciding on graphing tool](#_iv2z4w3zgdjk) 10

[Bol.com](#_vb1elmz5aa5x) 14

[DeSlegte.com](#_mdifq1d0qdnl) 15

[**Conclusion**](#_zb6a8deeo8dk) **17**

[How will data be acquired from the websites using scraping tools?](#_ic2zbsozpn6) 17

[What graph theory methods can be applied in this situation?](#_uqxob0svh57q) 17

[What do the recommendation graphs say about the websites?](#_d0cc196vqwb3) 17

[How do the graphs of the book recommendations of the websites Bol.com and DeSlegte.com compare in the English cookbooks in the price range of €10,- to €20,- ?](#_a8w5j5vxn1eo) 19

[**Discussion**](#_sq4rm16ipgtg) **19**

[**References**](#_zd1ika7k320u) **20**

[Appendix A-1 General scraping code](#_ar4bjvog63di) 20

[Appendix A-2 DeSlegte.com spider](#_5brspa1qdq92) 22

[Appendix A-3 Bol.com spider](#_vy1a6fq6z7a3) 23

[Appendix B Bol.com filters](#_mif9q18bk0qg) 25

[Appendix C Bol.com recommended section](#_vl8ckk64rr3e) 25

[Appendix D DeSlegte.com filters](#_y0tn0cr5xsx9) 26

[Appendix E DeSlegte.com recommended section](#_z456lfc2m7ir) 26

[Appendix F-1 Bol.com table snippet](#_lnxbz9) 27

[Appendix F-2 DeSlegte.com table snippet](#_nbc8u089cvj4) 27

[Appendix G-1 Bol.com retrieved analysis values](#_v7u9s7od1rq4) 28

[Appendix G-2 DeSlegte.com retrieved analysis values](#_1ypbrfz6vbj7) 28

# Introduction

Online purchasing is gaining popularity. This led to the development of an increasing number of web stores. However, how can you select the best one? Apart from speed, affordability, and design, web stores may be compared based on their approach to cross-selling. This is a feature that most online stores provide in order to entice customers to view and buy additional products, by showing recommended products *(Bol.com: Appendix C, DeSlegte.com: Appendix E)*. Two websites will be compared in this project on the basis of just that. Both Bol.com and DeSlegte.com sell books. This report will discuss the variations in recommendations for English cookbooks in the price range of €10,- to €20,-. The research aim will be expanded upon in this report. The technique for answering the questions, outlined in the goals of the project, will also be discussed.

# Context

The aim of the project is to analyse the recommendations of cooking books from the websites Bol.com and DeSlegte.com. The use of scraping tools and graphing tools will be used to better understand and apply graph theory concepts to the recommendations of the websites. The scraping tools will be used in order to get a large amount of data from the websites and facilitate the analysation of the link structure of the collected data.

## Main research question

How do the graphs of the book recommendations of the websites Bol.com and DeSlegte.com compare in the English cookbooks in the price range of €10,- to €20,- ?

## Sub questions

1. How will data be acquired from the websites using scraping tools?
2. What graph theory methods can be applied in this situation?
3. What do the recommendation graphs say about the websites?

# Methodology

## Research Strategy

In order to understand the problem of the main question, a qualitative research in the form of a literature study is performed. In order to investigate the two web pages, data must be collected. The findings are explained in the first sub-question, “How will data be acquired from the websites using scraping tools?”. This will offer a better understanding of the first part of the solution to the main question. Qualitative research was applied in order to better understand what scraping is and how it can be applied in this research. The result of this study is to get tools to gain data of the web pages.

To better understand the necessary graph theory concepts of the data collected on both websites Bol.com and DeSlegte.com, another literature study is done. In order to analyse the data collected, a graphing tool API will be used. This API must provide the necessary tools to analyse the data and show the behaviour of the recommendation graph. The findings are shown and explained in the sub-question “What graph theory methods can be applied in this situation?”.

After the literature studies are done, the data can be analysed properly. These findings are shown and discussed in the sub-question “What do the recommendation graphs say about the websites?”.

## Research Design

1. **How will data be acquired from the websites using scraping tools?**

A literature study is performed in order to determine which scraping tools are the most suitable in order to analyse the data of the websites. A high performance scraping tool will be necessary as a large amount of data will be analysed from the websites in order to get the link structure. The tool will be applied to the data pool of the websites.

1. **What graph theory methods can be applied in this situation?**

A literature study will be performed to determine what graph theory concepts can prove meaningful in analysing the recommendations.

1. **What do the recommendation graphs say about the websites?**

A suitable tool will be chosen via literature study. This tool is meant to gain understanding from the data of sub-question 1 by applying the found mathematical methods in sub-question 2. This will be done through an experiment. The experiment will show the behaviour of the recommendation graphs, which will offer a deeper understanding of the recommendations on the websites.

# Results and analysis

## How will data be acquired from the websites using scraping tools?

### Deciding on a scraping tool.

In order to apply mathematical analysis, data has to be collected first. The data for each website is stored in the HTML of the web pages. It is possible to go over all items required and get the necessary information from the HTML by hand. This would take a lot of time and effort. In order to gain data from the chosen web pages, Bol.com and DeSlegte.com, web-scraping is efficient and quick. Web scraping is an automated method to gain information out of the HTML code of a web page. The tool is able to get the HTML code for a given web page and navigate to the requested item. The tool chosen to scrape the websites must be free, efficient and not too complicated. This is because the amount of web pages that must be scraped is at least 1000 pages. The complexity of the tool can also not be too much because there is not enough time to learn to apply very complex code or applications. There are applications and websites that offer scraping as a service, but most demand payment like Zyte(6), this is a company that offers scraped data for a fee, but an appointment must be made first. Other applications available for this project have some restrictions to them. For example, Octoparse(7) is a free alternative but is slow and not versatile, this can give unexpected troubles when in use. There are also libraries for programming languages that support scraping. Libraries are collections of pre-written code that users can use to optimize tasks. The language selected for the objective is Python, as the group members are the most comfortable coding in Python. Scrapy(5) is a free, fast and rather simple scraping library specially made for Python. This tool is selected because it fits within the budget, has up-to date documentation and tutorials online, and works efficiently

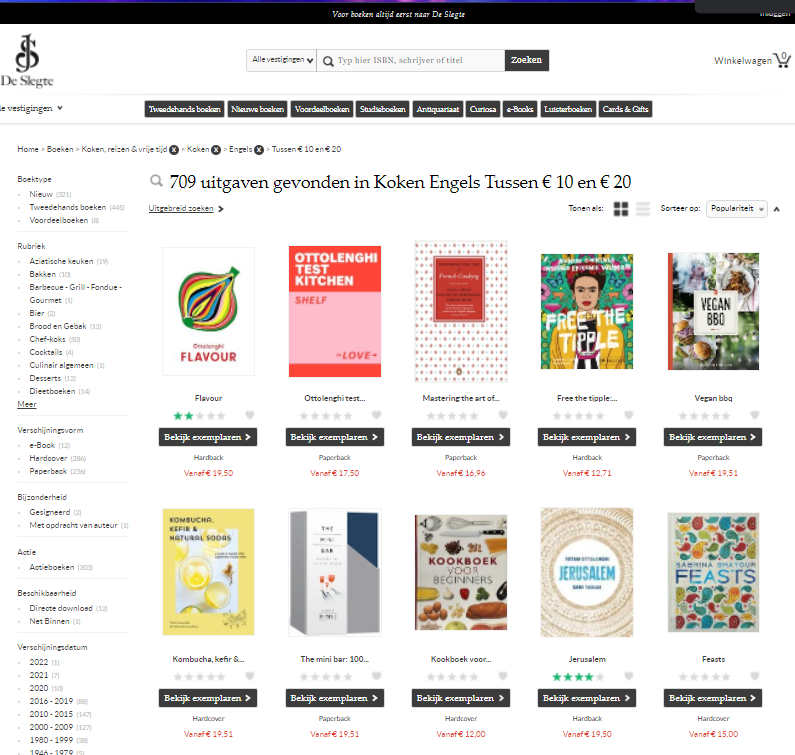
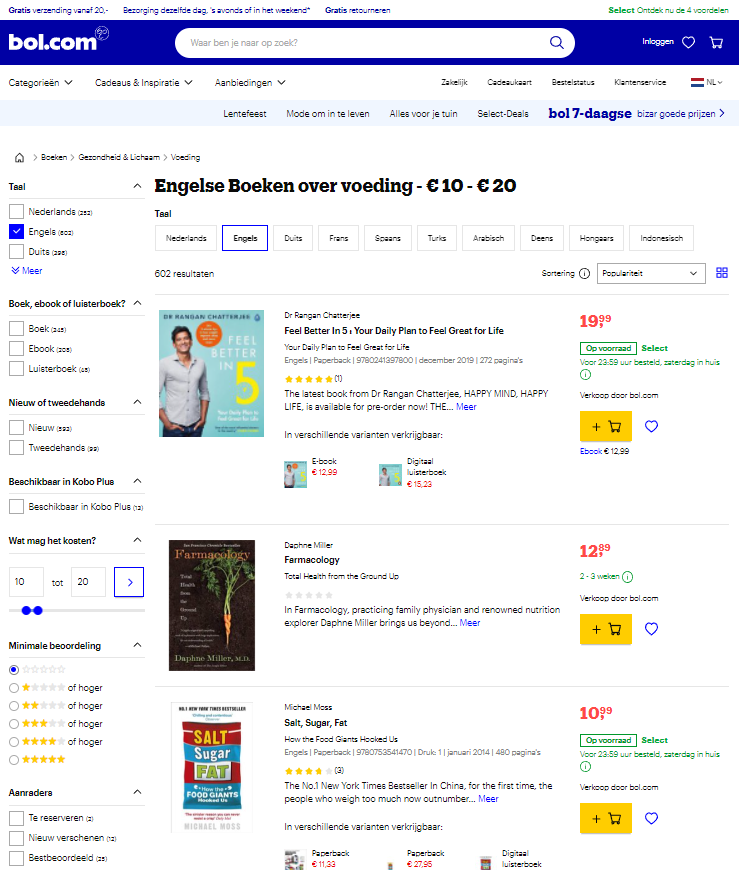
### Using the scraping tool to obtain useful data

To use Scrapy, a program has to be coded in the coding language Python *(Appendix A-1)*. Scrapy does not have a user interface, and its sole purpose is to only obtain data from the scraped websites.

#### Scraping with Scrapy

Spiders are classes that define how a website is scraped. This includes how to ‘crawl’ a website and collect structured data from its pages. Crawling is the process through which a spider reads information from a website.(1)

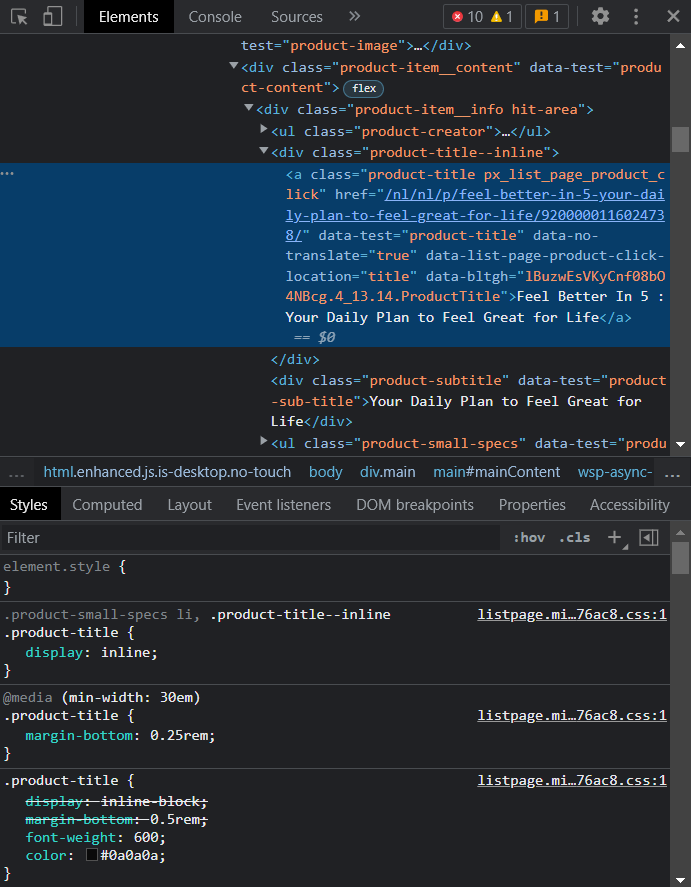
A workspace is established in order to construct the appropriate spider for gathering the data required for this study. This workspace is secured by GitHub version control. In the workspace, there is a folder containing the spiders. Scraping the websites Bol.com and DeSlegte.com yields two distinct spiders, resulting in two distinct files. The code for the two spiders is very similar, but because of the difference in HTML code layout, a few variables differ *(DeSlegte.com Appendix A-2, Bol.com Appendix A-3)*. A parser class is written in a separate location and imported into the spider files to eliminate as much duplicate code as feasible (*Appendix A-1)*. This parser class gathers all necessary data and is formed early in the spider's life.



*Figure 1: The Bol.com book list Figure 2: DeSlegte.com booklist*

The spider is first guided to the URL holding the book list. Figure 1 and 2 show these book lists. A list of books is collected by scraping a list of URLs maintained on the website for each page that has the correct filters. For Bol.com this is 26 at the moment of writing this, and for DeSlegte.com this is 18. When requesting data with Scrapy, the program returns HTML code. This code is too big to use as data in later stages. In order to navigate the specific data needed, XPath is used. The XPath for every book in the list of Bol.com is as follows: /html/body/div[1]/main/wsp-async-browse/div/div/div[3]/div[3]/div[1]/div/div[4]/div/ul**/li**/div[2]/div/div[1]/a**/@href**. For DeSlegte.com, this is: /html/body/div[2]/div[2]/div/div[3]/ul**/li**/div/div/div[2]/h3/a**/@href**. The XPath works similar to how folders are saved in a pc, with every ‘/’ the path goes into a ‘folder’ or element in this case. The way that the XPaths are obtained is through Chrome DevTools, this is a fast way to look at the HTML code of a page. Chrome DevTools also offers a way of copying the XPath of a singular item, this is used to obtain the needed paths. A picture of the interface of Chrome DevTools is offered in Figure 3. The XPaths aren’t always ready to use right away, a small part needs to be added in this first case. ‘@href’ is an attribute signifying a hyperlink. This means that in every case that a link is desired, ‘@href; needs to be appended to the path. Another modification made is in the middle of the path, namely ‘li’. When copying the path directly from the Chrome DevTools, the path only leads to the single item chosen, but the whole list of books is needed. Originally, the XPath contained ‘li[x]’, with x being the index of the book. To make the XPath usable for the entire list of books, the ‘[x]’ is removed.

The original XPath for DeSlegte.com is /html/body/div[2]/div[2]/div/div[3]/ul/li[1]/div/div/div[2]/h3/a .

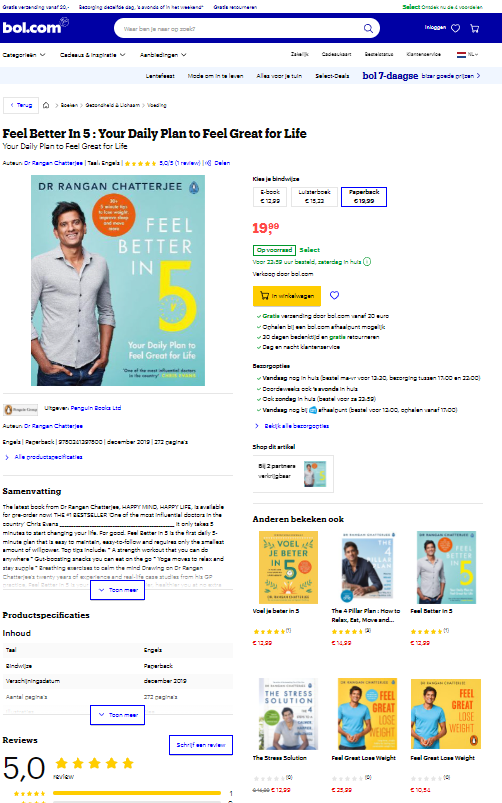
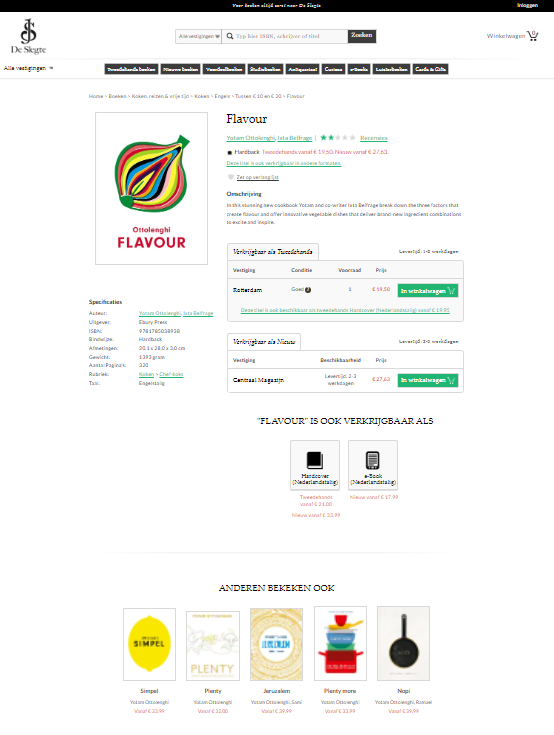


*Figure 3: Interface of Chrome DevTools*

The filters are important for this research as it reduces the amount of items that are analysed in the experiment, explained later in the report. The filters applied to both websites are: the items have to be books about cooking and/or food; the items’ price range is between €10,- and €20,-, and ; the books must be written in English *(Bol.com: Appendix B, DeSlegte.com: Appendix D).*

When a list of books is gathered, the program goes through all items in the list, scraping every link within those. When the program has finalized scraping these, it continues on to the next page for the next booklist. From the obtained HTML of a single book, a list of recommended books is gathered using a different XPath. The URLs of those books are saved. The spider then proceeds to the next book or page. The XPath used for Bol.com is: //\*[@id="mainContent"]/div/div[1]/div[5]/div[2]/div[2]/ul**/li[x]**/div/div[2]/a**/@href**.

The XPath for DeSlegte.com is: /html/body/div[2]/div[2]/div/div[2]/div/div[2]/div[5]/ul**/li[x]**/div[2]/h3/a**/@href**. In this case, the ‘li[x]’ is not removed. This is because no more than 5 recommended books are scraped. Because DeSlegte.com sometimes has less than 5 books to suggest, only the top five recommendations on Bol.com will be taken into account. X can be a number from 1 to 5, meaning that there are 5 XPaths for both websites. Again, the ‘@href’ is added at the end because the data collected is a link. Pictures of the individual book pages can be found in Figure 4 and 5 for reference.



*Figure 4: DeSlegte.com recommended section Figure 5: Bol.com recommended section*

### The data

All data collected is stored in a comma separated file (CSV) of that website *(Bol.com Appendix F-1, DeSlegte.com Appendix F-2)*. One row in that CSV consists of the items: item, recommended1, recommended2, recommended3, recommended4 and recommended5. All these items are links to individual books. This makes the data more consistent in order to reach a more fair conclusion.

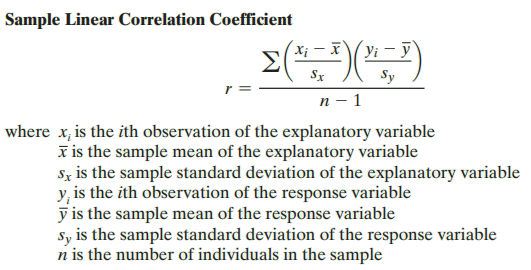
There are two motives for storing the data in CSV files first, instead of using the data directly in mathematical code. Using the data directly for calculations can be time inefficient when testing and building the program. The second motivation is that scraping a lot of data from a website in a relatively short amount of time can trigger a web scraping prevention. This is a defence mechanism some websites have put in place against web scraping attacks (10). The scraping used in this project might be flagged as an attack, and revoking access to the page for a certain time. In order to prevent being blocked for a certain amount of time, the scraping is done in multiple runs. Firstly, the file is created and the first few book lists are scraped. When this is done running, the code is changed in order to scrape the next upcoming pages. This continues until all pages are scraped. After the CSV files are filled with scraped data, it can be analysed with mathematical concepts through Python code. The mathematical concepts that will prove useful for analysing these networks first have to be researched.

## What graph theory methods can be applied in this situation?

The data gained previously can be interpreted as two directed graphs. One for Bol.com and one for DeSlegte.com. The nodes in these graphs are representations of the books from the original pool (the books that comply with the filters that were originally set, (*Appendix B and D).* The edges, or arcs since they are directed graphs, denote which book recommends which other book. If these books recommend books outside the pool, these are added as vertices too. The out degree, which denotes how many books one book recommends, is five for every node that is present in the original pool at Bol.com. Bol.com has at least five or more recommendations for every original book on the website. For uniformity, the first five is taken each time. The nodes that were not present in the original pool will have an out degree of zero, since there are no recommendations taken into account from books that are not within the used filters.

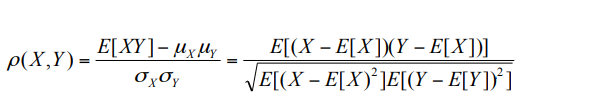
Adegree distribution displays the frequency of the degrees and the distribution of the degrees. Plotting big graphs may not give us any clear indication of the difference between graphs. The degree distribution makes the differences of graphs more evident (3).

The linear correlation coefficient (Pearson correlation coefficient)returns a value between -1 and 1. A coefficient of 1 implies a perfect positive linear relation between two variables. A coefficient of -1 implies a perfect negative linear relationship. The closer the coefficient is to one of these maximum values, the stronger the evidence of a positive or negative linear association between the two variables. If the coefficient value is closer to 0, it indicates that there is no linear correlation between the two variables. The coefficient is defined by:

(9)

The coefficient isimpressionable by the amount of books referred to that are outside the original pool. More books outside the original pool results in more books with an out degree of 0, and an in degree of mostly one (unless recommended more often) and thus a degree of 1 in total. This influences the degree distribution of the graph as a whole, since the books present in the original data pool have an out degree of 5, and an in degree of either 0 or more, depending on if and how frequently they are recommended. Comparing the linear correlation coefficient of both websites, could illustrate how the websites compare in recommending books within or outside the same set of filters put in place.

Assortativity is also a computation that returns a coefficient between -1 and 1. This value represents to what extent nodes in a network associate with other nodes, being of similar sort or being of opposing sort. *“A network is said to be assortative when high degree nodes are, on average, connected to other nodes with high degree and low degree nodes are, on average, connected to other nodes with low degree. A network is said to be disassortative when, on average, high degree nodes are connected to nodes with low(er) degree and, on average, low degree nodes are connected to nodes with high(er) degree”* (4)*.* The definition of a linear correlation between two values is defined as:

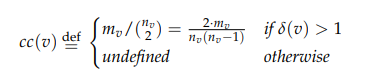
**

*“Where µ X and µY are the mean of X and Y respectively, E XY −µ X µY [ ] is the covariance of X and Y and σX and σY are their respective standard deviations”* (4)*.*

The relations between popular and unpopular books can be analysed, which explains if relatively popular or unpopular books are recommended more often. The result from this test will give more comprehension about how the websites’ algorithm works and how they possibly differ from each other.

Examining aclustering coefficientcould also produce interesting results. This is a measure of the degree to which vertices tend to cluster together. The clustering coefficient illuminates to which extent the neighbours of a specific node are also neighbours of each other within a group (3). The formula for the clustering coefficient is defined as follows:

*“Consider a simple connected, undirected graph G and vertex v ∈ V(G) with neighbour set N(v). Let nv = |N(v)| and mv be the number of edges in the subgraph induced by N(v), i.e., mv = |E(G[N(v)])|.”*



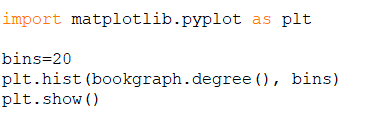
With the clustering coefficient cc(v) and vertex v with degree δ(v) (3) . The coefficient is a value between 0 (neighbourhood is not connected at all) and 1 (neighbourhood is perfectly connected). Values closer to 0 could prove that there is no relation between books being recommended to each other. A value closer to 1 means that certain books recommend each other more often. A possible explanation for this could be the same author, same (partial) title or a different characteristic. To define these, further research is required.

## What do the recommendation graphs say about the websites?

### Deciding on graphing tool

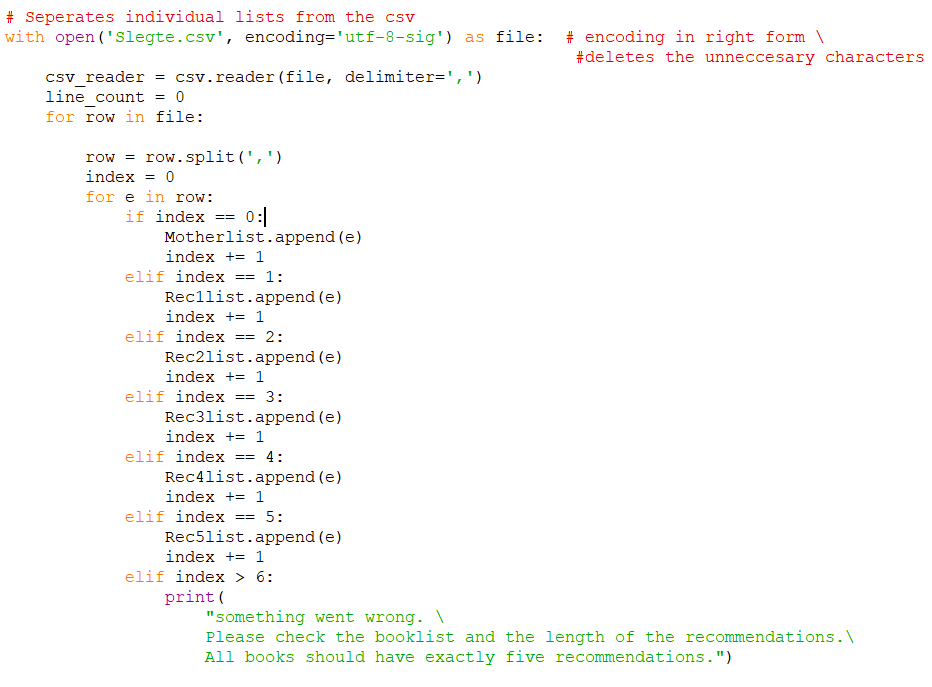
Working with the coding language Python has also narrowed down the available libraries for analysing the data. The available graph analysis libraries encountered for Python are NetworkX, iGraph and Graph-Tool (11). The library is selected upon the following criteria: The library has to be available for the operating system of the computer used, it has to have an up-to-date and clear documentation, and it has to contain the graph theory analysis methods that are estimated to be useful for this project. Graph-Tool has a very clear syntax, but is only available for the operating systems GNU, Linux and MacOS. Using a Windows computer for this research excluded the availability of this library. IGraph and NetworkX have similar and clear syntax and documentation.

To further aid the implementation of these libraries, the libraries Matplotlib and CSV were implemented. Matplotlib aids in visualizing graphs, such as implemented in Figure 6.



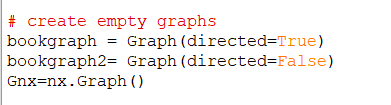
*Figure 6: Code snippet of Matplotlib library implementation*

The CSV library is useful for efficient reading of comma separated value files. First, the correct file has to be chosen. The library has functions to automatically divide the file by comma (or different symbol), named the delimiter. Each row read in the file is divided into the columns per row, and separate lists are made from each column (Motherlist, Rec1list, Rec2list, Rec3list, Rec4list, Rec5list) as displayed in Figure 7.



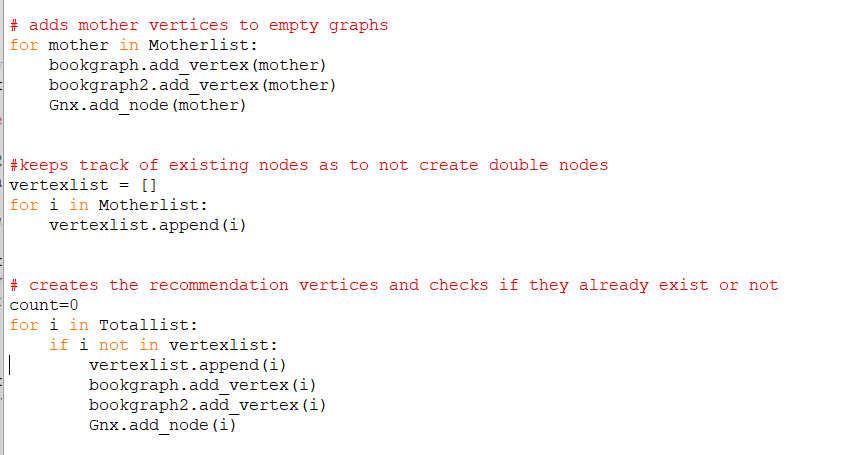
*Figure 7: Code snippet of CSV library implementation*

IGraph is used for the majority of the project. First, an empty graph is created (Figure 8). Three graphs are created to implement all analysis methods.



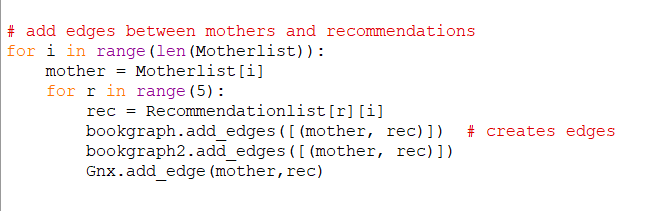
*Figure 8: Creating empty graphs*

For each graph, all books in the original data pool (dubbed ‘Mothers’) are added to the node list. Additionally, each book from the recommendation columns is also added to the node list, if it is not already present in the node list (Figure 9).



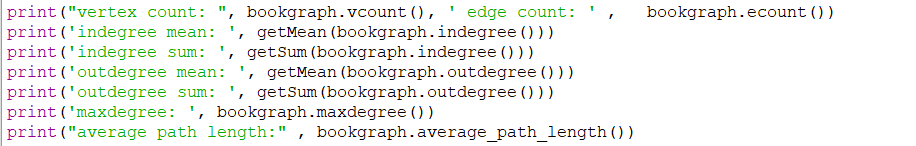
*Figure 9: Adding nodes to the graphs*

To finalize the graphs, all edges denoting recommendations between books need to be registered. Each original book in ‘Motherlist’ is on a separate row, denoted by ‘i’ (Figure 10). The recommendation book will be on row ‘i’ and column r, which is a loop that goes from 0 till 5 and thus passes each column and each recommendation that an original book has. For each recommendation book, an edge is registered for the graphs.



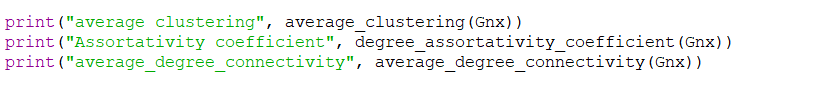
*Figure 10: Record all edges from original books to recommendation books*

Once the graphs are set up successfully, the library analysis methods can be used to print statements (Figure 11).



*Figure 11: retrieving analysis through print statements*

NetworkX is also used for additional analysis methods (the average clustering, the assortativity coefficient and the average degree connectivity) which were unavailable on IGraph (Figure 12)



*Figure 12: Retrieving analysis through NetworkX library functions*

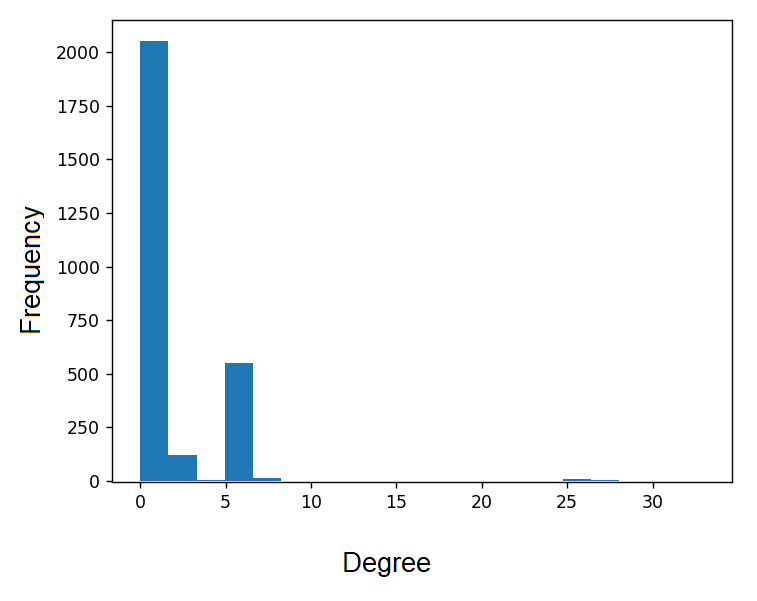
The values are printed and ready to partake in the research *(Bol.com: Appendix G-1, DeSlegte.com: Appendix G-2).*

### Bol.com

The total vertex count of the scraped data is 2762, of which 571 original books. Thus, 2191 new books are added to the graph. These new books are from outside the original data pool, so the books that are within a different price range, language, or genre. The graph has 2855 edges, which means that 2855-2191= 664 books (24.0% of the total amount of books) are either chosen multiple times as a recommendation or were present in the original pool. The amount of nodes has grown with 383.7% in comparison to the books in the original pool.

The degree distribution (Figure 13) shows the frequency of each degree. The most nodes, approximately, 2100, have a degree of 1. This corresponds to the previous calculation that states that, 2191 new books are added. As explained earlier, the new books will more often than not have a degree of 1, unless multiple books recommend the same book. If that is the case, the degree 2 and 3 belong to those books with a frequency of about 125 and 30 respectively.

The next high frequency is a degree of five, which belongs to about 500 nodes. These are most of the 571 original books. A degree of 6, 7, 8 or 9 also exists, although with very low frequency. These are books that are in either the original data pool and are recommended a degree-5 times, thus 1, 2, 3, or 4 times respectively. It is also possible books from the recommended data pool have a degree of 6, 7, 8, or 9. This however means that the book is recommended the same amount as the degree, and this is estimated as implausibly high. The same estimation is for the very high outliers with a degree of 25, 26 and 27.



*Figure 13: Degree distribution of Bol.com*

The Pearson correlation coefficient of the in and out degree is -0.805. The closer the coefficient is to -1, the stronger the evidence of negative association between the two variables (9). There is a notable relationship between the degrees, where on an average the in degree declines when the out degree increases. This could be due to the fact that books outside the original data pool more often than not have an in degree of only one, and always have an out degree of 0. The original books from Bol.com have an out degree of five, and leading to books with a degree about one influences the correlation coefficient.

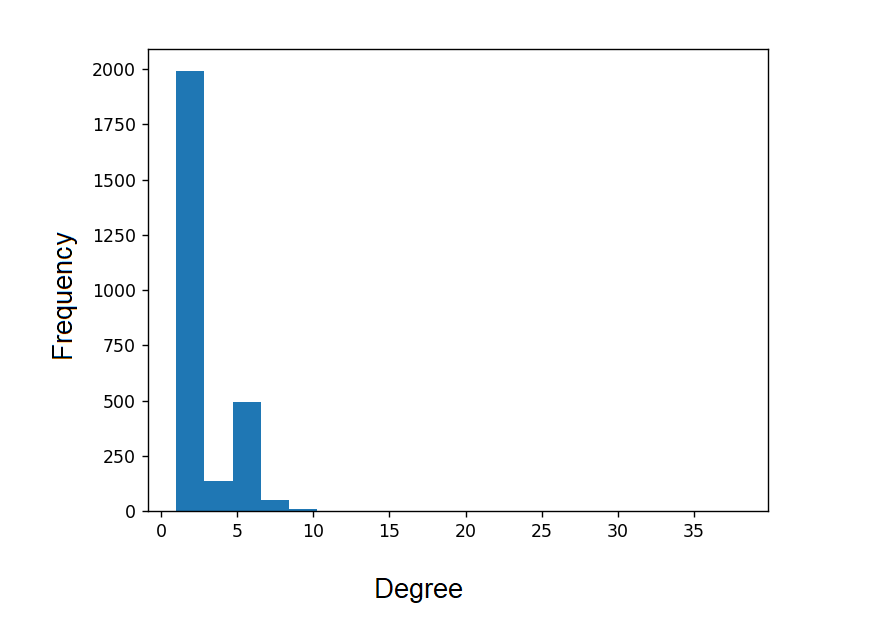
The assortativity coefficient is also -0.805. This value is rounded up to three decimals, but is not exactly equal to the Pearson coefficient. The assortativity ranges from –1 to +1. Positive values indicate an inclination for nodes of similar degrees to connect to each other. Negative values mean that large-degree nodes tend to attach to low-degree nodes. The value -0.805 means that large (in)degree nodes (relatively popular books) tend to attach to low-(in) degree books (relatively unpopular books).

The clustering coefficient is 0.023. This implies that the neighbourhood is not connected at all, as the value is very close to 0. In the context of the project, this seems sensible. Common recommendations are relatively rare, as only 24% of the books are recommended once or more.

### DeSlegte.com

For this website, not every book from the original pool contains five recommendations. The total vertex count is 2682, of which 636 in the original data pool. This means that 2046 new book are added to the graph. The graph has 2944 edges, which indicates that 2944-2682= 262 books (8.9% of the total amount of books) are either chosen multiple times as a recommendation, or books from the original data pool are also referenced. The amount of nodes has grown with 421.7% in comparison to the book in the original pool.

The degree distribution (Figure 14) displays the degree distribution of DeSlegte.com. A degree of 1 is most common, referring to the books outside the original data pool that are only recommended once. The second-biggest degree frequency is 5, with about 500 nodes, referring to the original books. A small part of the total books, again expected to be from the original books, have a degree of 7, 8, or 9, estimated to be about 300 books altogether.



*Figure 14: Degree distribution of DeSlegte.com*

The Pearson correlation coefficient, using the in and out degree as the variables, is -0,032. This value is fairly equally positioned between the maximum values -1 and 1, thus no linear correlation can be interfered about this network. This indicates that there is no (linear) relation between the in and out degree. The website tends to use some books as recommendation, more often. The absence of an equal amount of recommendations can result in unfair data, intervening with the correlation coefficient. This however does not imply an absence of correlation, it simply implies no linear correlation(9).

The assortativity coefficient value is also -0,032. This value is rounded up to three decimals, and is not exactly equal to the Pearson coefficient if rounded up to 8 decimals. Again, there is no indication that there is a notable correlation between the nodes with a higher degree connecting to nodes with a lower degree. A possible cause is if DeSlegte.com does take the relative popularity of books into account when picking books to recommend, but rather pick other characterizations to recommend books. To further define these possible characterizations, more research is required. A different cause could be because the network from this website does not create five recommendations for each original node, and this interferes with the data analysis

The clustering coefficient is 0.036, which is very close to 0. This implies that the neighbourhood is not connected. For Slegte.com, about 8.9% of the books are recommended more than once, and the clustering coefficient reflects that too.

# Conclusion

The objective of this report is to research how the recommendations of the websites Bol.com and DeSlegte.com compare to each other. For this, the stock of English cookbooks within the price range of €10,- to €20,- is compared using web scraping and network analysis.

To establish this, steps needed to be taken first.

## How will data be acquired from the websites using scraping tools?

In order to acquire the data, a literature study was performed in order to find an appropriate scraping tool. Scrapy was chosen in order as it is fast, free and relatively easy library to be used in order to scrape websites

A workspace has been created in GitHub in order to construct a custom spider to crawl the data and to have a version control of the code produced. The data is scraped with two different spider files for each of the websites, as the HTML code structure for both differs. The overall structure of the python code is similar.

Once the data was fully scraped, the data is stored in a CSV file which ensures a good efficiency as scraping websites takes time and can be confronted to the issue of scraping defences.

## What graph theory methods can be applied in this situation?

The original books from the pool are the nodes and the edges, which are directed to the recommendations. The recommendations have an out degree of 5, as the maximum recommendation given by Bol.com is 5. The graph theory applied to the context consists namely of degree distribution, linear correlation coefficient, and clustering coefficient.

## What do the recommendation graphs say about the websites?

The mathematical methods chosen in the previous sub-question were implemented using the network analysis libraries NetworkX and iGraph. From values returned from the analysis show that for Bol.com, 24.0% of the books are recommended more than once. For DeSlegte.com, only 8.9% of all books are recommended more than once. This indicates DeSlegte.com recommends a wider variety of books in comparison to Bol.com, which recommends the same books more often. A possible explanation is because Bol.com takes bestsellers into account and recommends these more often. To verify this, more research is needed.

DeSlegte.com recommends more books outside the original data pool than Bol.com. For DeSlegte.com the amount of nodes has grown with 421.7% in comparison to the book in the original pool, and for Bol.com this growth is 383.7%. Since the filters for the original data pool were relatively strict, it is unknown if the websites recommend books from different genres, different languages, or simply books outside the price range set.

The degree distribution of both websites has a subtle difference. DeSlegte.com has more frequency in degrees between 1 and 5, where Bol.com does not have any degrees of 3 and 4, but does have high outliers with a degree of 25, 26 and 27 *(Figure 13 and 14)*. The books with these extremely high degrees could be explained by bestsellers, which DeSlegte.com does not take into account, but this is merely a theory.

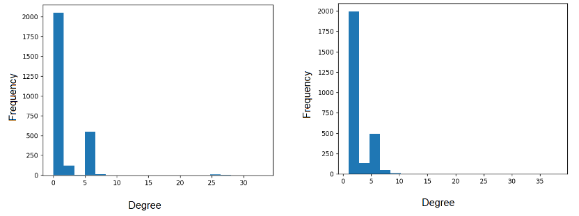


Figure 13: *Degree distribution of Bol.com* Figure 14: *Degree distribution of DeSlegte.com*

The Pearson correlation is calculated for the in and out degree of both websites. For Bol.com, this shows a strong negative association between the in and out degree. For DeSlegte.com, it indicates no (linear) correlation. Bol.com has an out degree of five for all original books, and seldom an in degree, namely only if the original book is recommended by others. This creates the relation of big out degrees often having small in degrees on average. A possible explanation for why DeSlegte.com does not produce similar results, may be the difference in out degree of original books. Due to the lack of consistent recommendation amounts on the original books from DeSlegte.com, original books recommend less than five books more often than Bol.com, transposing the Pearson coefficient.

The assortative value for Bol.com indicates that high degree nodes tend to connect to low degree nodes. For DeSlegte.com, there is no correlation perceived. Assortativity represents to what extent nodes in a network associate with other nodes, being of similar sort or being of opposing sort. It can be interfered that Bol.com connects to opposing sort more often, and DeSlegte.com does not have a set preference.

The clustering coefficient is for both graphs close to zero. This makes sense for the context of this project. Most books are recommended only once, so clustering is proven to be rare and difficult in the current context of the project. The clustering coefficient could possibly be bigger if the recommendations of the books outside the original data pool were also taken into account, creating a more complete graph with more edges, and more chance to have multiple edges leading to the same books. This could create more clustering. However, this is outside the reach of this project.

## How do the graphs of the book recommendations of the websites Bol.com and DeSlegte.com compare in the English cookbooks in the price range of €10,- to €20,- ?

DeSlegte.com recommends a wider variety of books in comparison to Bol.com, which recommends the same books more often. DeSlegte.com also recommends more books outside the original data pool. DeSlegte.com has more degrees ranging from 5 to 10, whereas Bol.com has less of these degrees but big outliers of books with 25 to 27 degrees.

For Bol.com, this shows a strong negative association between the in and out degree. For DeSlegte.com, it indicates no (linear) correlation. This can be a result of DeSlegte.com not having a consistent amount of out degrees (recommendations) per original book. The assortative value for Bol.com indicates that high degree nodes tend to connect to low degree nodes. For DeSlegte.com, there is no correlation evaluated.

# Discussion

When comparing the websites, the aim is to retrieve the data as fair and consistent as possible. When scraping DeSlegte.com, the data showed that not all books have the same amount of recommendations. Bol.com does offer a consistent amount of recommendations. When applying graph theory methods, the analysis is influenced by the amount of data and the consistency. This makes the conclusion less representative.

It is unknown if the books that are recommended outside the original data pool differ in genre, language, or price range. It could prove valuable to repeat the research with fewer filters on the data, such as no constraints on price range and language. This could result in showing which website recommends books in different genres (more frequently), and if an underlying reason can be found, such as the same author. More insight in the differences between the books could tell us more about the difference in recommendations from Bol.com and DeSlegte.com.

The current program offers no insight into which books belong to which degrees. Having this information could explain more about why Bol.com has a degree of 25 for some books and why DeSlegte.com does not. Including this in the project was however outside the current scope.

Because the graph created in this project is a subgraph from the total book graph from Bol.com and DeSlegte.com, certain analysis methods did not have any added value for this project. This could change by using a bigger subgraph. Analysing a bigger dataset, which is obtained by either relaxing the constraints on the original data pool or including recommendations of books outside this pool. This could shed more light on analysis methods, such as the clustering coefficient.

The size of the group members has an influence on the speed and quality of the research. After roughly 3 weeks, one group member was removed from the group. This member has created a delay themselves by bad communication and withholding deliverables. The loss of this member has removed some workforce, but did make the group more reliable and communicate more clearly.

# References

1. Scrapy at a glance — Scrapy 2.6.1 documentation. (2022). Scrapy. Retrieved on May 23rd, 2022, from <https://docs.scrapy.org/en/latest/intro/overview.html>
2. iGraph – Network analysis software. (z.d.). IGraph. Retrieved on May 23rd, 2022, from <https://igraph.org/>
3. Van Steen, M. (2010). Graph Theory and Complex Networks (1st edition, Vol. 2010).
4. Noldus, R., & Van Mieghem, P. (2015, 27 January). Assortativity in Complex Networks. Oxford Academic. Retrieved on May 20th, 2022, from [https://academic.oup.com/comnet/article/3/4/507/775579\](https://academic.oup.com/comnet/article/3/4/507/775579%5C)
5. Scrapy | A Fast and Powerful Scraping and Web Crawling Framework. (z.d.). Scrapy. Retrieved on May 25th 2022, from <https://scrapy.org/>
6. Zyte - #1 Web Data Extraction Services. (z.d.). Zyte (Formerly Scrapinghub) #1 Web Scraping Service. Retrieved on June 1st, 2022, from <https://www.zyte.com/data-extraction/>
7. Web Scraping Tool & Free Web Crawlers | Octoparse. (z.d.). Octoparse. Retrieved on June 1st 2022, from <https://www.octoparse.com/>
8. Introduction — NetworkX 2.8.3 documentation. (z.d.). NetworkX: Network Analysis In Python. Retrieved on June 1st, 2022, from <https://networkx.org/documentation/stable/reference/introduction.html>
9. Sullivan III, M. (2017). Statistics, informed decisions using data (5th edition). Pearson. Retrieved on May 23d, 2022, from <https://www.pearson.com/us/higher-education/program/Sullivan-Statistics-Informed-Decisions-Using-Data-plus-My-Lab-Statistics-with-Pearson-e-Text-Access-Card-Package-5th-Edition/PGM2398746.html>
10. Datadome.co. (z.d.). DataDome. Retrieved on june 6th 2022, from [https://datadome.co/learning-center/scraper-crawler-bots-how-to-protect-ythe-website-against-intensive-scraping/](https://datadome.co/learning-center/scraper-crawler-bots-how-to-protect-your-website-against-intensive-scraping/)
11. PythonGraphLibraries - Python Wiki. (2022, 25 maart). Python. Retrieved on may 23rd 2022, from <https://wiki.python.org/moin/PythonGraphLibraries>

### Appendix A-1 General scraping code

import scrapy

import csv

#the parser class handles the aquisition of the data

class ParserClass():

def \_\_init\_\_(self, paths, urlBuild,filen):

self.paths = paths

self.urlBuild = urlBuild

self.page = 0

self.file = open(filen, "w",newline="")

self.writer=csv.writer(self.file)

self.writer.writerow(["item", "recommended1", "recommended2", "recommended3","recommended4","recommended5"])

async def parse\_all(self,response):

links = [response.xpath(self.getMainPath()).getall()] #get all the links of this page

for link in links[0]: #for every link

yield scrapy.Request(self.getUrlBuild(2)+link, self.parse\_single)

if self.getPage()<10: #Bol.com has (about) 25 pages in this category, just scraping the first 3 for faster testing

self.page+=1 #increase the page number

url= self.getUrlBuild(0)+str(self.getPage())+self.getUrlBuild(1) #create the link for the next page

print('-----------------------now on to page: '+str(self.getPage())+'-----------------------')

yield scrapy.Request(url, self.parse\_all) #send the request to the parse function again

else:

print('-----------------------done scraping-----------------------')

async def parse\_single(self, response):

#try:

recommended1= self.getUrlBuild(2)+response.xpath(self.getSinglePath(0)).get()

recommended2= self.getUrlBuild(2)+response.xpath(self.getSinglePath(1)).get()

recommended3= self.getUrlBuild(2)+response.xpath(self.getSinglePath(2)).get()

recommended4= self.getUrlBuild(2)+response.xpath(self.getSinglePath(3)).get()

recommended5= self.getUrlBuild(2)+response.xpath(self.getSinglePath(4)).get()

data=[response.url,recommended1,recommended2,recommended3,recommended4,recommended5]

self.write(data)

print(data)

print('\n\n')

def getResponse(self):

return self.response

def getMainPath(self):

return self.paths[0]

def getSinglePath(self,nr):

return self.paths[1][nr]

def getUrlBuild(self,nr):

return self.urlBuild[nr]

def getPage(self):

return self.page

def write(self,text):

return self.writer.writerow(text)

def addLink(self,link):

self.allLinks.append(link)

### Appendix A-2 DeSlegte.com spider

*from \_\_future\_\_ import absolute\_import*

*import scrapy*

*from scrapy.crawler import CrawlerProcess*

*#if any outside class is made, put it in items and import it like this here*

*from tutorial.items import ParserClass*

*#to run: scrapy runspider sleg\_spider.py*

*class SlegSpider(scrapy.Spider):*

*name = "sleg" #the name that the spider will be called*

*def start\_requests(self): #on startup, the spider will start here*

*yield scrapy.Request(start, slegParser.parse\_all) #send out the first request, the code leaves this class immediately*

*#yield scrapy.Request(start, slegParser.parse\_single())*

*process = CrawlerProcess(settings={ #Some settings for the crawler*

*"FEEDS": {*

*"sitems.json": {"format": "json"},*

*},*

*})*

*start='https://www.deslegte.com/boeken/koken-reizen-vrije-tijd/koken/engels/10-20-euro/?p=1&sc=popularity&so=desc'*

*paths= ['/html/body/div[2]/div[2]/div/div[3]/ul/li/div/div/div[2]/h3/a/@href',['/html/body/div[2]/div[2]/div/div[2]/div/div[2]/div[5]/ul/li[1]/div[2]/h3/a/@href','/html/body/div[2]/div[2]/div/div[2]/div/div[2]/div[5]/ul/li[2]/div[2]/h3/a/@href','/html/body/div[2]/div[2]/div/div[2]/div/div[2]/div[5]/ul/li[3]/div[2]/h3/a/@href','/html/body/div[2]/div[2]/div/div[2]/div/div[2]/div[5]/ul/li[4]/div[2]/h3/a/@href','/html/body/div[2]/div[2]/div/div[2]/div/div[2]/div[5]/ul/li[5]/div[2]/h3/a/@href']]# first rec link added/ same for every'*

*urlBuild=['https://www.deslegte.com/boeken/koken-reizen-vrije-tijd/koken/engels/10-20-euro/?p=','','https://www.deslegte.com/']*

*#creates a parser object, go to items.py to see/edit the parser class*

*slegParser = ParserClass(paths,urlBuild,"sleg.csv")*

*process.crawl(SlegSpider) #puts the spider in the crawler*

*process.start() #Runs the spider, the script will block here until the crawling is finished*

*print('-----------------------done-----------------------\n\n\n\n\n\n\n\n') #this is for the user to know where the spider finishes*

*slegParser.file.close() #closes the CSV file*

### Appendix A-3 Bol.com spider

*from \_\_future\_\_ import absolute\_import*

*import scrapy*

*from scrapy.crawler import CrawlerProcess*

*#if any outside class is made, put it in items and import it like this here*

*from tutorial.items import ParserClass*

*#to run: scrapy runspider Bol\_spider.py*

*class BolSpider(scrapy.Spider):*

*name = "Bol" #the name that the spider will be called*

*def start\_requests(self): #on startup, the spider will start here*

*yield scrapy.Request(start, BolParser.parse\_all) #send out the first request, the code leaves this class immediately*

*process = CrawlerProcess(settings={ #Some settings for the crawler*

*"FEEDS": {*

*"items.json": {"format": "json"},*

*},*

*})*

*start='https://www.Bol.com/nl/nl/l/engelse-boeken-over-voeding/41026/8292/?page=1&12194=10-20'*

*paths= ['/html/body/div[1]/main/wsp-async-browse/div/div/div[3]/div[3]/div[1]/div/div[4]/div/ul/li/div[2]/div/div[1]/a/@href',['//\*[@id="mainContent"]/div/div[1]/div[5]/div[2]/div[2]/ul/li[1]/div/div[2]/a/@href','//\*[@id="mainContent"]/div/div[1]/div[5]/div[2]/div[2]/ul/li[2]/div/div[2]/a/@href','//\*[@id="mainContent"]/div/div[1]/div[5]/div[2]/div[2]/ul/li[3]/div/div[2]/a/@href','//\*[@id="mainContent"]/div/div[1]/div[5]/div[2]/div[2]/ul/li[4]/div/div[2]/a/@href','//\*[@id="mainContent"]/div/div[1]/div[5]/div[2]/div[2]/ul/li[5]/div/div[2]/a/@href']]*

*urlBuild=['https://www.Bol.com/nl/nl/l/engelse-boeken-over-voeding/41026/8292/?page=','&12194=10-20','https://www.Bol.com']*

*#this is information specific to the Bol.com website*

*BolParser = ParserClass(paths,urlBuild,"Bol.csv")*

*#creates a parser object, go to items.py to see/edit the parser class*

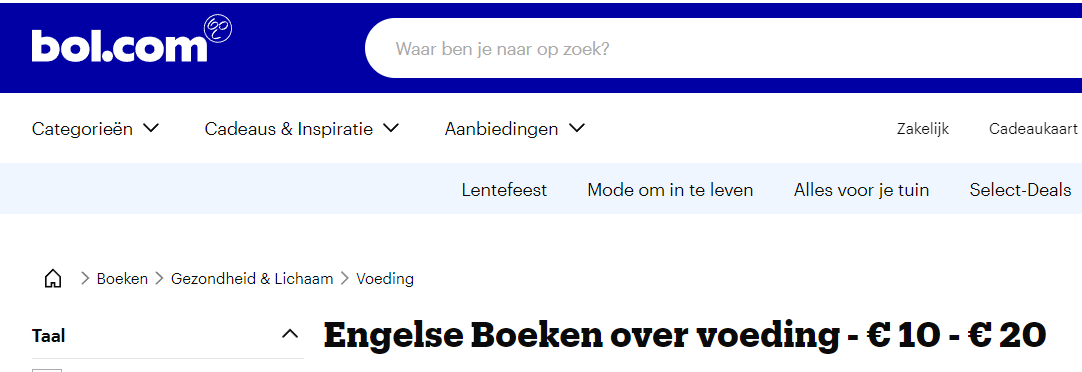
*process.crawl(BolSpider)*

*process.start() # the script will block here until the crawling is finished*

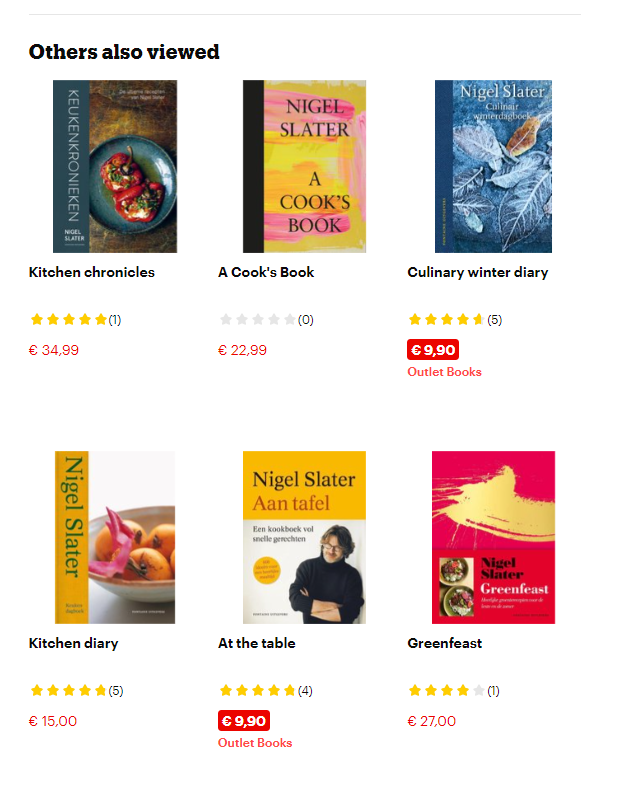
*print('-----------------------done-----------------------\n\n\n\n\n\n\n\n')*

*BolParser.file.close()*

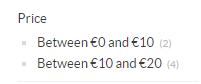
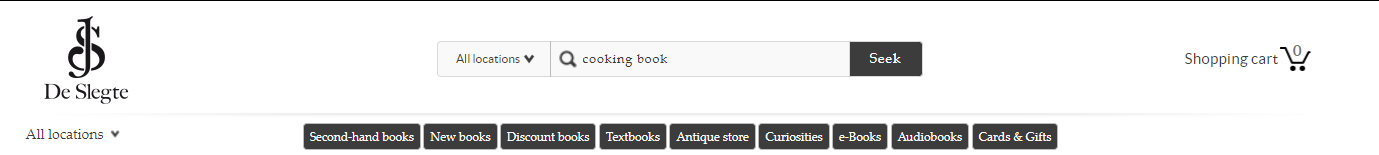
### Appendix B Bol.com filters

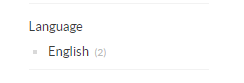


### Appendix C Bol.com recommended section



### Appendix D DeSlegte.com filters

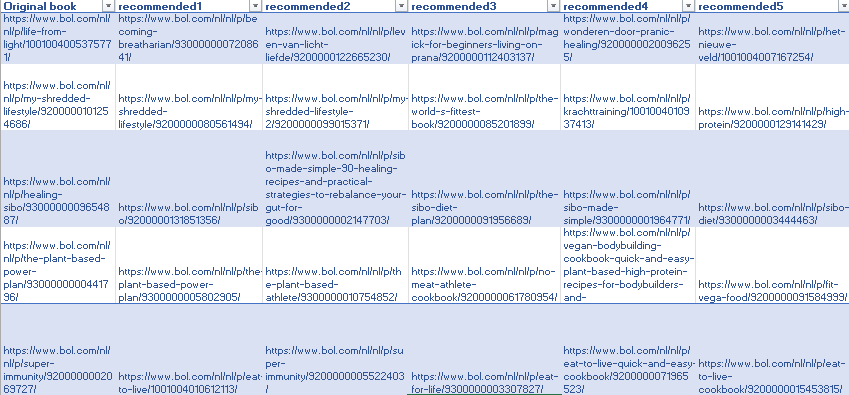




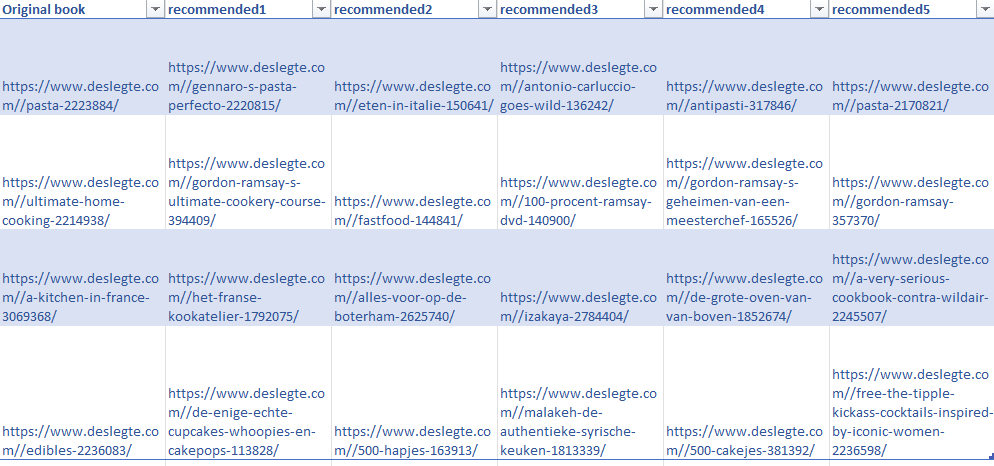
### Appendix E DeSlegte.com recommended section



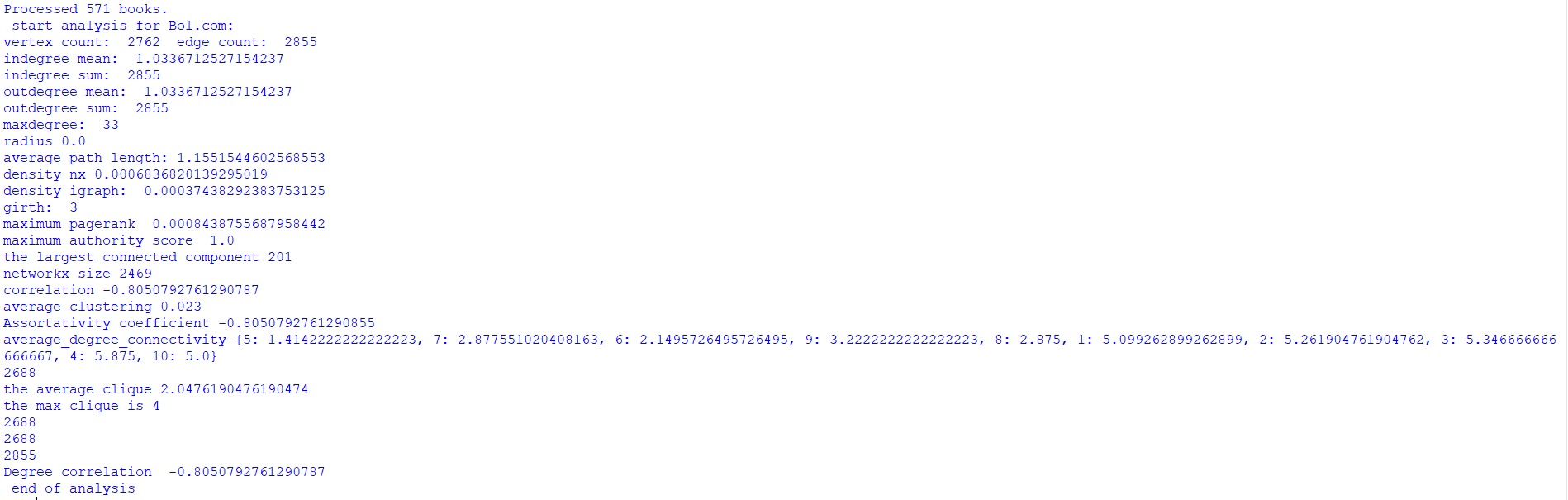
### Appendix F-1 Bol.com table snippet



### Appendix F-2 DeSlegte.com table snippet



### Appendix G-1 Bol.com retrieved analysis values



### Appendix G-2 DeSlegte.com retrieved analysis values

