Web Scraper Revealing Trends of Target Products and New Insights in Online Shopping Websites

Habib Ullah[1], Zahid Ullah[2], Shahid Maqsood[3], and Abdul Hafeez[4]
Department of Computer Science and IT, University of Engineering and Technology, Jalozai Campus, Pakistan[1,4]
Department of Electrical Engineering, CECOS University of IT and Emerging Sciences, Peshawar, Pakistan[2]
Department of Industrial Engineering, University of Engineering and Technology, Peshawar, Pakistan[3]

***Abstract*—Trillions of posts from Facebook, tweets in Twitter, photos on Instagram and e-mails on exchange servers are overwhelming the Internet with big data. This necessitates the development of such tools that can detect the frequent updates and select the required information instantly. This research work aims to implement scraper software that is capable of collecting the updated information from the target products hosted in fabulous online e-commerce websites. The software is implemented using Scrapy and Django frameworks. The software is configured and evaluated across different e-commerce websites. Individual website generates a greater amount of data about the products that need to be scraped. The proposed software provides the ability to search a target product in a single consolidated place instead of searching across various websites, such as amazon.com, alibaba.com and daraz.pk. Furthermore, the scheduling mechanism enables the scraper to execute at a required frequency within a specified time frame.**

***Keywords*—Django QuerySet (DQS); e-commerce; hamming distance algorithm (HDA); Levenshtein distance algorithm (LDA); scraper; scheduling mechanism**

I. Introduction

E-commerce is a mechanism of selling/purchasing products/services over the Internet. It is like a virtual product store where products are available and customers can browse and add products/services to the shopping carts. The customers are required to complete the transaction requirements by filling the transaction form with the required information, such as complete address, number of products, and the credit card number. Following the successful completion of the transaction requirement, an e-mail notification is sent to the customer.

The commercial use of Internet has been increasing exponentially day by day. In the modern era, shopping over Internet is becoming a common trend [1]–[5]. The pervasiveness of e-commerce has enabled an increasing number of transactions over the internet and thus, framing a compelling case for more and more business to turn online. Behavior by teenagers and older adults using smart-phone-based online shopping is becoming significant [1]. Recent work shows that emails, social media and smart-phone based advertising have been an established medium between the customers and businesses. In order to create and maintain efficient communication between customers and business, e-marketing techniques has been developed [6].

Data scraping pertains to the process of extracting data from online files through computer scripts. Such extracted data exists in the form of tables and lists. The interface between the script and the Internet for extracting data is basically a set of commands, i.e., an application programming interface (API). These APIs can be trained and used to extract data for search results across a group of websites. Automating web searches and extracting data from multiple pages for search results, merely requires users to input search items rather than navigating and searching websites individually.

The use and impact of web scraping examples are enormous. Tracking of pricing activities between different competitors can be accomplished via web scraping on a single or group of websites with minimal human intervention. Similarly, web scraping has enabled efficient searching of multiple websites and an increased transparency in research (scholar.google.com). Web scraping is pretty common in academic databases, such as Scopus, web of science and Inspec. Information that is not readily portable, such as author list and the corresponding author information can be extracted efficiently using web scraping. Subsequently, automating web searches is beneficial in other scenarios where web search is time consuming.

From server's perspective that is hosting a website can endure a remarkable strain in case of scraping a bulk of pages from a single or scraping huge volume of pages across multiple websites in a short span of time. In spite of that, acquiring few thousands of search results through scraping can hardly have deleterious startle on the server's performance. In nutshell, web scraping provides a resource-efficient search and transparency with minimum additional efforts.

Individual buyers or small organization can benefit from open-source and free web scraper available over the Internet. Additional developments will make the web scraper even better and easier to use and a well-trained API will benefit the prevailing networks.

However, the customers have limited knowledge about the trends of the target products between different e-commerce websites, such as daraz.pk, alibaba.com, and olx.com. These websites often have different rates for the same product. Finding the best price for a given product thus becomes a daunting task due to a variety of shopping websites. Customers have to search different online websites manually in order to find an optimal price for a target product. Therefore, a specific tool is required that can show the trends of a particular product in online markets and e-commerce websites.

We propose a scarping algorithm for detecting marketing trends in online shopping websites. Specifically our contribution is the pioneer work on using web scraping for extracting best price of the target products from multiple websites rather

than a single website. This work entails the products from the top online websites. When the user wants to buy a product, he/she can search the product in one consolidated website and the search results are pulled up from fabulous marketing websites in just one consolidated place. Instead of using datasets from Amazon and Google, etc. our scraping method can be adapted for a variety of other shopping websites.

Rest of the paper is organized as follows: Section II entails the related work. Section III captures modeling of the proposed system. Section IV discusses the results and Section V concludes the paper.

II. Related Work

Numerous scrapers have been written in various programming languages and frameworks are being used for retrieving web data, such as beautiful soup, scrapy, Java, and Ruby. Beautiful soup is used to extract banner ads from different websites [7]. The problem of keyword suggestion was implemented according to the keywords entered by the user using clustering bipartite advertiser-keyword graphs [8]. The clustering submarkets were recovered with the help of advertisers, depicting a usual bidding behavior and the sets of keywords with an affinity to the same market place.

The data collected from booking commercial and large apartments does not reflect the latest market activity and thus, hides the recent knowledge of rental markets in the US. Scraper has been designed and developed to bridge this gap [9]. Scraper has been developed to distill most important news from large amount of news data [10]. Methods have been developed to quantify and predict the feedback from customers on a given product. This can further help marketing and investors to refine their decision making for addressing customer requirements precisely [11]. Web bots have been used for modelling traffic patterns generated by different internet bots [12]. Neural Network has been used to detect buying or non-buying sessions from user sessions that involves only human intervention instead of those carried out by internet bots [13]

To avoid complexity, a simplified version of a scarper has also been implemented [14]. A framework was developed for scraping and retrieving the trendiness of YouTube content and viewers statistics−their watching time and shares [15]. Nonetheless, the YouTube APIs do not allow third parties to easily scrap such information. A framework naming YOUStatAnalyzer enables researchers to create their own data sets based on a variety of search criteria [15]. The framework has also the capability to analyze the created data sets for extracting useful features and distinguishing statistics.

The scheduling of jobs/tasks on processor is the most important and challenging task. Time slicing deals with the switching of context within the processor. However, space slicing specifies the ways for how to map processes onto the processor [16]. In order to achieve an optimal scheduling for processes, a general mathematical framework, resource task networks, was formulated [17]. In another scheme, scheduling of batch jobs based on first-come-first-serve was discussed on large parallel processor [18]. Using gang scheduling, initiated only by embarrassingly parallel jobs, helps preventing severe fragmentation. Furthermore, operating system support was provided in order to provide robust parallelism in addition to hardware-level parallelism [19], [20].

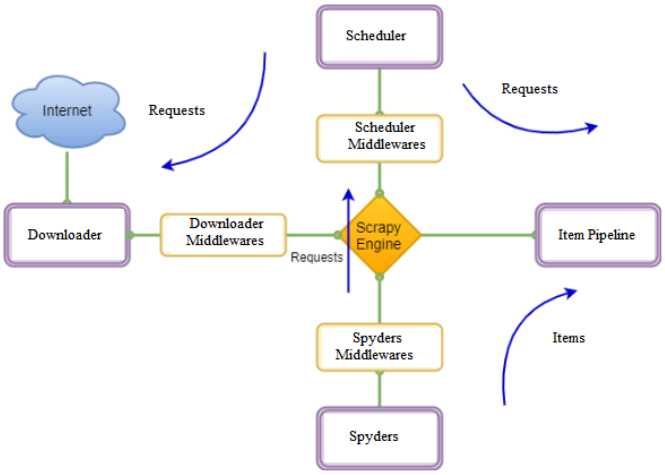

Fig. 1. The proposed system model. The proposed system contains five basic units, which are scheduler, downloader, scrapy engine, item pipeline, and spypder.

Interestingly, a gamut of searching algorithms are more efficient than straight forward searching techniques such as Hamming, Needleman Wunsch, Smith Waterman, Knuth Morris Pratt, Boyer Moore, and RabinKarp. A comparative study of different types of string matching algorithms, observation on their time and space complexities, and corresponding efficiency and performance has been tested with different biological sequences [21].

The entire samples from a finite number of keywords in a given string of text were computed by a simple and efficient algorithm [22]. An algorithm has been designed by constructing a finite state pattern-matching machine from the keywords, which was further used to process the text string into a single pass. The time taken by constructing the pattern matching machine was proportionate to the total sum of spans of the keywords. The speed up achieved by the algorithm was used to accelerate the search in the library bibliography by a factor of $5 - 10$.

III. Modeling of the Proposed System

The proposed system model, given in Fig. 1, has five basic components and are discussed here. In order to build a rapid prototype, the system has been implemented in Python. The implementation modules, details and their relationships are delineated in Fig. 1. As shown, the system is composed of the following four modules: 1) downloader; 2) scheduler; 3) item pipeline; and 4) spyders.

1) **Scheduler**: This component is responsible for scheduling all the requests and responses in Scrapy. This further queues up all the requests that are received from the engine and passes these to the downloader.
2) **Downloader**: The job of this module is to download all the required pages that are passed by the Scrapy engine and to send the downloaded pages back to the Scrapy engine through download middleware.

3) **Scrapy engine**: Scrapy engines are used to scrap large scale data. The heart of the system is the Scrapy engine. The goal is to control all the processes in Scrapy and the entire requests and responses passed through it from one component to another.
4) **Item pipeline**: The functionality of this module is to filter the data. It validates the data and checks to see whether the data is scrapped and also, clean data.
5) **Spyder**: An abbreviation for Scientific Python Development Environment. Spyder integrates important libraries, such as NumPy, Matplotlib, and SciPy and is an open-source tool for scientific programming. It is a class of python that defines how to extract the required data and the target page to be crawled [23]. It generates a request that will be sent to downloader through Scrapy engine. Items are scraped and stored whenever the desired response is received to the spyder.

The subsequent sub sections explains web scraping, scheduling of scrapers, and the search mechanism used for target products.

A. Web Scraping

Web scraping is also called web data extraction. It is a process, which is used to extract large amount of data from websites and to store the extracted data into the local storage in different formats. Web scraping is used for different purposes such as research, analysis of market and comparison of price, collection of opinion of public in business, jobs advertisements, and collection of contact detail of required business.

The data of websites are only shown in the web browser. If we need to check all the data of any website, we cannot do this without going to every page and cannot copy the data for the personal use owing to the longer time it takes to be copied. Web scraping is technique that provides to copy the data from websites in a reasonable amount of time instead of copying manually. It automates the manual copying process using a web crawler and bot.

The web scraping software is connected to the website through hypertext transfer protocol (HTTP). It fetches the page and extracts data from that page and swaps among the multiple pages of websites according the requirement to extract data. When the data is extracted, it will then be exported into different format such as CSV and JSON according to the needs.

B. Scheduling of Scrapers

It is a method in which specified, arranged work or processes are assigned to the resources to complete it. Virtual elements of the work such as threads and processes are scheduled to hardware resources like processors and expansion cards. The goals of the scheduler is to keep the entire resources busy and share them effectively in order to maximize the CPU usage and quickly switch processes onto CPU for time sharing to get a desired output.

Operating system entails a variety of schedulers − long-, medium-, and short-term schedulers. For the purpose to schedule running scrapers, a scraper is using Django-celery. Celery is an asynchronous task queue and supports distributed message passing. Task queues are used in order to distribute the workload among the given processors.

Input to the celery daemon is the task orders and then the execution of corresponding tasks is performed sequentially in order to complete the entire job, which ensures that none of the tasks is lost; even if the system is over burdened. In our proposed work, celery works as a job replacement tool that can be controlled by Django admin interface.

Algorithm 1 Hamming Distance Algorithm

```
Input: S1 and S2
Output: dist
if S1 ≠ S2 then
    Raise value error
    statements...
end if
dist = sum(S1(x,y != S2(x,y) for S1(x,y) and S2(x,y) in zip(S1, S2))
return dist
```

Algorithm 2 Levenshtein Distance Algorithm

```
Input: S1 and S2
Output: Levenshtein distance (LD)
if len(S1) ≠ 0 then
    return len(S2)
end if
if len(S2) ≠ 0 then
    return len(S1)
end if
if len(S2)- 1 = 0 then
    LD = 0
else
    LD = 1
end if
A1 = LD(S1, len(S2) − 1, S2, len(S1)) + 1
A2 = LD(S1, len(S2), S2, len(S1) − 1) + 1
A3 = LD(S1, len(S2) − 1, S2, len(S1) − 1) + LD
Minimum(A1,A2,A3)
```

C. Search Mechanism for Target Products in Websites

The goal of string matching algorithm is to find one or more than one patterns within a larger string or text. In this research work, we have implemented two types of searching mechanisms − Hamming distance and Levenshtein distance techniques.

1) Hamming distance: The Hamming distance counts the difference at positions between any two strings (S_1 and S_2) of an identical length. In other words, this means the least number of interchanges required to convert a string S_1 into S_2, as illustrated in Algorithm 1.

2) Levenshtein distance: There are three operations that are used to transform one string into another in order to find the similarities between strings [24]–[27]. The Levenshtein distance between any two strings *C* and *D* of length |C| and |D|, respectively can be formulated by $lev_{C,D}$ (|C|,|D|) as given

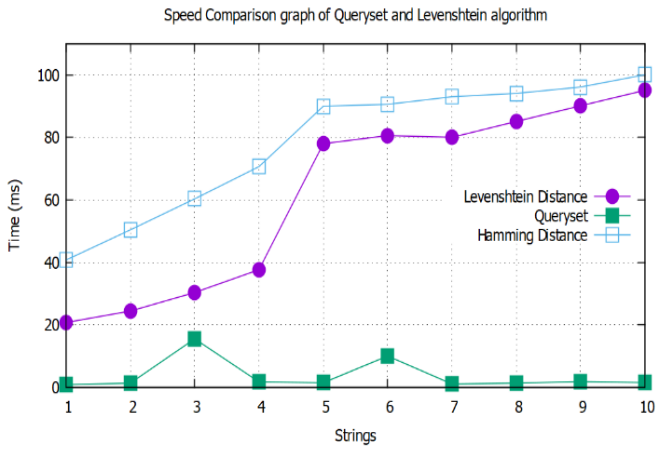

Fig. 2. The speed comparison of Django QuerySet (DQS), Levenshtein distance algorithm (LDA) and Hamming distance algorithm (HDA). DQS takes lesser time compared to LDA and HDA in string comparison.

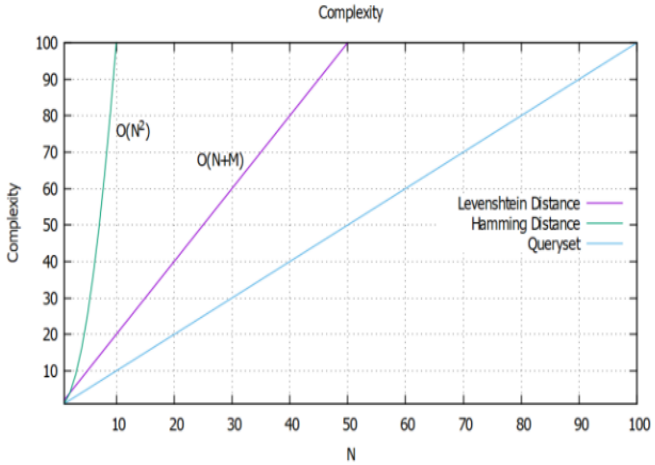

Fig. 3. Complexity curve of Levenshtein distance, Hamming distance and Queryset. As clear from the graph that HDA is more complex compared to LDA while the LDA complexity is much higher compared to DQS.

in (1).

$$lev_{(C,D)}(i,j) = \begin{cases} max(i,j) & \text{if } min(i,j) = 0 \\ min \begin{cases} lev_{(C,D)}(i-1,j)+1 \\ lev_{(C,D)}(i,j-1+1) \\ lev_{(C,D)}(i-1,j-1)+1_{(C_i \neq D_j)} \end{cases} & \text{otherwise} \end{cases} \quad (1)$$

Where ($C_i \neq D_j$) is the indicator function that string *C* and *D* are entirely different and hence, equal to 0. Furthermore, C_i = D_i means equal to 1, where $lev_{(C,D)}(i,j)$ corresponds to the span between the initial character of of *C* and *D*.

It should be noted that the first element in the minimum corresponds to the deletion from *C* to *D*, while the second pertains to the insertion and the third is related to either similarity/dissimilarity of the corresponding symbols. Algorithm 2 shows the Levenshtein distance algorithm.

IV. Experimental Setup and Discussion on Results

A. Experimental setup

The experimental setup entails Toshiba laptop with a main memory of 8 GB and a processing capability of 2.5 GHz with a storage capacity of 1 TB. The operating system used was Microsoft Windows 10. The scrapers were written in Python using Django framework.

We used Scrapy framework for the web scraping to write crawler. The crawler is configured and tested to scrap data from different websites. We scraped 15000 products in approximately 11 minutes. Finally, we compared three types of searching algorithms– Django Queryset, Hamming distance, and Levenshtein distance [28]. We compared these techniques based on the speed, complexity, and the throughput.

B. Discussion on Performance

The speed comparison of Django QuerySet (DQS), Levenshtein distance algorithm (LDA) and Hamming distance algorithm (HDA) is shown in Fig. 2. DQS consumes lesser time than LDA and HDA in string matching process. Each query is executed ten times and then averaged. The average time taken to execute a query has a greater value for LDA and HDA than DQS. The reason for preferring LDA over HDA and DQS is the large number of comparisons.

The LDA filters out the most accurate results for searching algorithm. As the string size increases, the time taken to process also increases in both HDA and LDA. The reason is that both algorithms perform complex operations of insertion and deletion during matching process against the database the database. Increasing string size is proportional to the number of operations.

The complexity of Levenshtein distance algorithm (LDA), Django QuerySet (DQS), and Hamming distance algorithm (HDA) is captured in Fig. 3. It is obvious that HDA is more complex than LDA, while the LDA complexity is greater than DQS. The complexities of LDA, HDA and DQS is O(N+M), $O(N^2)$ and O(N), respectively.

It means that the complexity of LDA is higher than that of the DQS. LDA performs complex operations of insertion, deletion, and substitution; however, DQS only checks whether the database values contain the search phrase. HDA measures the least number of errors required to alter one string into another.

Fig. 4 shows the throughput achieved by LDA, HDA, and DQS. The throughput of DQS is greater than both LDA and HDA; however, LDA is preferred owing to its accuracy (approximately 70 %) and versatility. Experiments have shown that the accuracy of HDA is about 81 %, which is better than LDA; however, LDA is still preferred because the complexity of $O(N^2)$ is higher compared to O(N+M).

Moreover, the major problem with HDA is that the length of both of the strings must be the same; otherwise, the algorithm will not work. The LDA performs more operations on strings than HDA and DQS; it filters most accurate result for our matching algorithm than DQS. For a query set of size

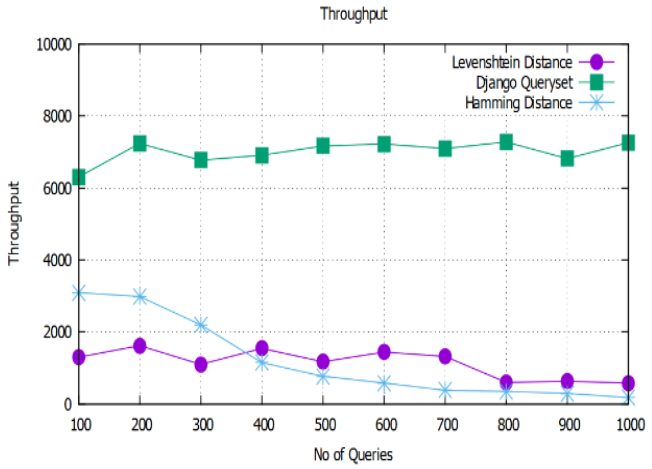

Fig. 4. Throughput comparison among Levenshtein distance algorithm (LDA), Hamming distance algorithm (HDA), and Queryset (DQS). The throughput of DQS is higher compared to both LDA and HDA; however, LDA is preferred because of its accuracy (approximately 70 %) and versatility.

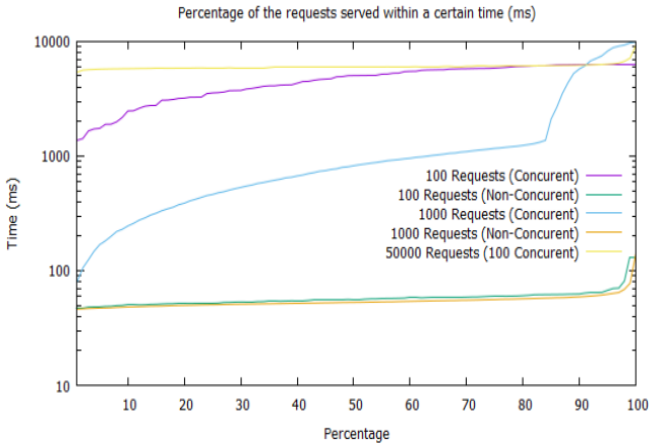

Fig. 5. Time taken to serve an increasing number of requests. A large number of users requests are taken such as 100, 1000, and 5000 against both concurrent and non-concurrent requests.

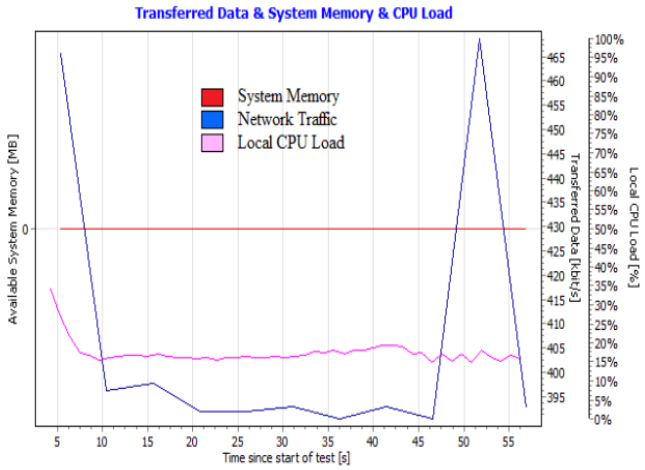

Fig. 6. Results of simulation for a span of one-minute using the Web Server Stress Tool for transferred data, system memory, and CPU load.

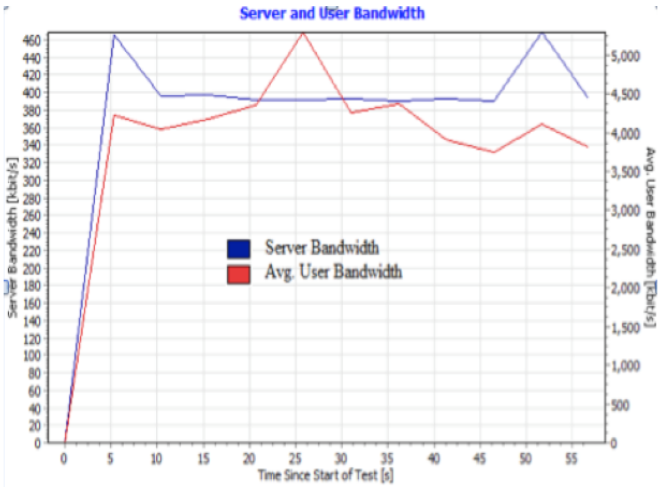

Fig. 7. Server and user bandwidth. Whenever the requests are arrived at the system, there is an increase in bandwidth requirements of both the server and user, which means the server bandwidth is 460 Kb/s and the user bandwidth is 370 Kb/s.

200, all of the three algorithms produce higher value due to the fact that system was busy performing other tasks as well.

The server response against user requests is delineated in Fig. 5. In our proposed system, two types of user requests were evaluated: concurrent and non-concurrent requests. The response of the web server hosted on a local server was evaluated against user requests.

In order to measure the percentage of the requests served with a certain time frame, the parameter of interest was concurrency. It is observed that the requests having concurrency were served quickly. Howbeit, significant system resources will be allocated in case large number of requests is arrived at once.

Fig. 6 shows the results of simulation for a span of one-minute using the Web Server Stress Tool in order to measure the data transferred, the system memory used, and the CPU load. The network traffic is the amount of data transferred over the network at a given point of time.

The peaks in network traffic was observed at time stamp 5 sec and at 52 sec. That is, the traffic reached up to 456 Kbps, mainly because of higher number of requests. Comparatively, the network traffic was lower during time span of 10 sec 45 sec. The reason is lower number of requests generated against the server.

The benchmark results of server and user bandwidth in Kbps are shown in Fig. 7. In the beginning as there is zero number of requests generated against the server, therefore, the average bandwidth of user and server is 0. However, with an increasing number of requests received by the system, the bandwidth of the server as well as user goes higher. The peak bandwidth achieved by the server and user requests is 400 Kbps and 370 Kbps, respectively.

V. Conclusions and Future Work

In this work, we proposed a filtering web pricing system that exploits web scraping techniques in order to extract trends

and suggest best price of a target product from top of the line commercial websites such as amazon.com, alibaba.com, and daraz.pk. The designed framework incorporates Scrapy framework for web crawling and scraping. Celery is used to schedule scraper in order to analyze the crucial pages in the target websites and distill the required information against a given product.

For the sake of string matching between the users typed search and the online products, Levenshtein, Hamming, and QuerySet are used. The results show an improved accuracy and an accelerated response for retrieving search results while using Levenshtein distance. Albeit, throughput of QuerySet is much higher than Levenshtein and Hamming method. To the best of our knowledge, this is the first attempt to filter knowledge about best pricing of a product from top of the line websites.

As of future work, we aim to enable the proposed framework to suggest relevant and non-relevant items based on a factor k. Furthermore, future research directions include integration of the proposed work into social media, such as Google and facebook to suggest best prices about the products based on the user preferences. Ultimately, the goal is to enable users to search for the best price from top of the line website, whether it may be finding best and cheap hotels, or finding the cheapest airfare while traveling or finding the best deal for jewelry at wedding ceremonies and the list continues to increase.

ACKNOWLEDGMENT

The authors would like to offer thanks to all the anonymous reviewers for their nice and valuable suggestions, which increased quality of the work.

REFERENCES

[1] S.-M. Kuoppamäki, S. Taipale, and T.-A. Wilska, "The use of mobile technology for online shopping and entertainment among older adults in finland," *Telematics and Informatics*, vol. 34, no. 4, pp. 110–117, 2017.

[2] J. Ogilvie, K. Lindsey, K. Reynolds, and W. M. Northington, "Examining reactive customer engagement strategies in online shopping cart abandonment: A regulatory fit perspective," in *Rediscovering the Essentiality of Marketing*. Springer, 2016, pp. 755–756.

[3] W. Hong, K. Y. Tam, and C. K. B. Yim, "E-service environment: Impacts of web interface characteristics on consumers online shopping behavior," in *E-Service: New Directions in Theory and Practice*. Routledge, 2016, pp. 120–140.

[4] M. D. Griffiths and J. Parke, "The social impact of internet gambling," *Social Science Computer Review*, vol. 20, no. 3, pp. 312–320, 2002.

[5] M. Griffiths, "Internet addiction: does it really exist?" 1998.

[6] H. Kresh, A. Laible, M. Lam, and M. Raisinghani, "Online advertising: Creating a relationship between businesses and consumers," in *Global Business Value Innovations*. Springer, 2018, pp. 47–61.

[7] E. Vargiu and M. Urru, "Exploiting web scraping in a collaborative filtering-based approach to web advertising," *Artificial Intelligence Research*, vol. 2, no. 1, p. 44, 2012.

[8] J. J. Carrasco, D. C. Fain, K. J. Lang, and L. Zhukov, "Clustering of bipartite advertiser-keyword graph," 2003.

[9] G. Boeing and P. Waddell, "New insights into rental housing markets across the united states: web scraping and analyzing craigslist rental listings," *Journal of Planning Education and Research*, vol. 37, no. 4, pp. 457–476, 2017.

[10] F. Hamborg, N. Meuschke, and B. Gipp, "Matrix-based news aggregation: exploring different news perspectives," in *Digital Libraries (JCDL), 2017 ACM/IEEE Joint Conference on*. IEEE, 2017, pp. 1–10.

[11] S. Wang, "From social media to innovation and marketing intelligence: A simulation to forecast online review and rating performance," *Journal of Digital & Social Media Marketing*, vol. 4, no. 3, pp. 251–262, 2016.

[12] G. Suchacka and D. Wotzka, "Modeling a session-based botsarrival process at a web server," in *Proceedings of the 31st European Conference on Modelling and Simulation (ECMS17)*, 2017, pp. 605–612.

[13] G. Suchacka and S. Stemplewski, "Application of neural network to predict purchases in online store," in *Information Systems Architecture and Technology: Proceedings of 37th International Conference on Information Systems Architecture and Technology–ISAT 2016–Part IV*. Springer, 2017, pp. 221–231.

[14] R. B. Penman, T. Baldwin, and D. Martinez, "Web scraping made simple with sitescraper," 2009.

[15] M. Zeni, D. Miorandi, and F. De Pellegrini, "Youstatanalyzer: a tool for analysing the dynamics of youtube content popularity," in *Proceedings of the 7th International Conference on Performance Evaluation Methodologies and Tools*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2013, pp. 286–289.

[16] R. M. Bryant, H.-Y. Chang, and B. S. Rosenburg, "Operating system support for parallel programming on rp3," *IBM Journal of Research and Development*, vol. 35, no. 5.6, pp. 617–634, 1991.

[17] G. Schilling and C. Pantelides, "A simple continuous-time process scheduling formulation and a novel solution algorithm," *Computers & Chemical Engineering*, vol. 20, pp. S1221–S1226, 1996.

[18] U. Schwiegeishohn and R. Yahyapour, "Improving first-come-first-serve job scheduling by gang scheduling," in *Workshop on Job Scheduling Strategies for Parallel Processing*. Springer, 1998, pp. 180–198.

[19] T. Jones, S. Dawson, R. Neely, W. Tuel, L. Brenner, J. Fier, R. Blackmore, P. Caffrey, B. Maskell, P. Tomlinson *et al.*, "Improving the scalability of parallel jobs by adding parallel awareness to the operating system," in *Proceedings of the 2003 ACM/IEEE conference on Supercomputing*. ACM, 2003, p. 10.

[20] A. Gupta, A. Tucker, and S. Urushibara, "The impact of operating system scheduling policies and synchronization methods of performance of parallel applications," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 19, no. 1. ACM, 1991, pp. 120–132.

[21] P. Pandiselvam, T. Marimuthu, and R. Lawrance, "A comparative study on string matching algorithms of biological sequences," in *International Conference on Intelligent Computing*, 2014, pp. 1–5.

[22] A. V. Aho and M. J. Corasick, "Efficient string matching: an aid to bibliographic search," *Communications of the ACM*, vol. 18, no. 6, pp. 333–340, 1975.

[23] M. Schrenk, *Webbots, spiders, and screen scrapers: A guide to developing Internet agents with PHP/CURL*. No Starch Press, 2012.

[24] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," in *Soviet physics doklady*, vol. 10, no. 8, 1966, pp. 707–710.

[25] L. Yujian and L. Bo, "A normalized levenshtein distance metric," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 6, pp. 1091–1095, 2007.

[26] B. Gujral, H. Khayrallah, and P. Koehn, "Translation of unknown words in low resource languages," 2016.

[27] L. C. Guillen, J. Domenico, K. Camargo, R. Pinheiro, and C. Coeli, "Match quality of a linkage strategy based on the combined use of a statistical linkage key and the levenshtein distance to link birth to death records in brazil." *International Journal for Population Data Science*, vol. 1, no. 1, 2017.

[28] P. E. Black, "Dictionary of algorithms and data structures," Tech. Rep., 1998.