Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 51-17 (2018) 778–783

Integration and Analysis of Agricultural Market Information Based on Web Mining

Jianghui Zhou[1], Chunming Cheng[1], Li Kang[1*], Ruizhi Sun[1,2]

1 College of Information and Electrical Engineering, China Agricultural University,
Beijing 100083, China (e-mail: {zjh1994, ccm, kangli, sunruizhi} @cau.edu.cn);
2 Key Laboratory of Agricultural Information Acquisition Technology,
Ministry of Agriculture, Beijing 100083, China.

Abstract: Agricultural big data can be used to guide agricultural production, forecast agricultural market demands, and support agricultural decisions. How to effectively extract and use the information on the Internet, which contains a large amount of agricultural information, has become a huge challenge. This paper proposes three kinds of automatic data acquisition strategies based on (focused, incremental, custom) Web crawler technologies, which are better suited to different types of agricultural websites than traditional Web crawlers. In addition to solving asynchronous processing, dynamic page rendering, distribution, and data-persistent problems encountered during data acquisition, this paper also proposes to combine the Aho-Corasick algorithm to improve the text matching efficiency. Finally, the acquired agricultural market data was visually analyzed by using key technologies of Web mining. This study takes Chinese agricultural official websites, agricultural products wholesale market websites, and e-commerce websites as examples to integrate, process, visualize, and analyze the data acquired by using the three automatic data acquisition strategies proposed in this paper.

Keywords: Agricultural big data; Decision support; Web crawler; Web mining; Data visualization

1. INTRODUCTION

Agricultural market information can objectively describe economic activities and changes of the agricultural market. It is a general name of targeted and cost-effective knowledge, news, data, intelligence that can be used for agricultural production, management, and market forecasting. From the perspective of the agricultural market demands, agricultural big data can be used to guide agricultural production, forecast agricultural market demands and support agricultural decisions, so as to achieve the desired goals of avoiding risks, increasing income, and managing transparently. Monitoring agricultural market information can ensure the balance of supply of agricultural products, effectively alert the unstable factors of the agricultural market, and ensure the sustainable and stable development of the agricultural market.

With the rapid development of new-generation information technologies such as the Internet, cloud computing, and big data, various types of massive data have been rapidly formed, providing effective ways to solve the difficulties and problems faced by the development of agricultural big data. The main purpose of Web information search is to discover Web information resources by using a technology called Web crawler to automatically roam on the Internet and find target content as much as possible. Ramakrishna (2010) summed up some Web mining ideas. It makes sense to apply traditional data mining methods and Web mining ideas, to agricultural market websites and extract interesting, potential, useful patterns, and hidden information from agricultural market network resources and agricultural market network activities. The excavated information can be used for agricultural information management, decision support, process control as well as for the maintenance of agricultural data itself.

Finding information on Web is a difficult and challenging task because of the extremely large volume of data and noise (Waldherr 2017). Due to inconsistent data structures used by the Web, it becomes difficult to acquire data automatically. The problem is amplified when Web crawlers have to deal with semi-structured and unstructured data. In addition, most of the current webpages are dynamically generated by JavaScript, as are agricultural market websites. Without a JavaScript rendering engine, the general Web crawlers cannot complete dynamic page acquisition. It means that they cannot be applied well to the agricultural market websites. So how to design crawlers suitable for different types of agricultural market websites and integrate the acquired data is worth studying.

2. DATA ACQUISITION STRATEGIES

Data acquisition as the first step in Web mining is commonly achieved by using web crawler technology. A Web crawler is also called a Web spider or Web robot (Spetka 1994). It is a software program that traverses the hyperlink structure of the Web automatically to locate and retrieve information. It starts from a certain page of the website, reads the content of the webpage, finds other hyperlinks in the webpage, and then finds the next webpage through these hyperlinks. Keep going until all webpages on the Internet are crawled (Cho 2001).

2.1 Classification

So far, there is no standard classification for the Web crawlers,

2405-8963 © 2018, IFAC (International Federation of Automatic Control) Hosting by Elsevier Ltd. All rights reserved.
Peer review under responsibility of International Federation of Automatic Control.
10.1016/j.ifacol.2018.08.101

and Kumar (2017) gave a broad one. In this study, we mentioned four types of Web crawlers as follows:

1) Universal Crawler: These crawlers are not limited to webpages of a particular theme or domain. They follow links endlessly and get all webpages they encounter.
2) Focused Crawler: These crawlers are used for searching information related to some specific theme from the Web (Chakrabarti 1999). Focused crawling not only specifies theme of interest, but also provides some labeled examples of relevant and irrelevant webpages (Yu 2010).
3) Custom Crawler: These crawlers are a special form of universal crawlers. They pre-select target websites, analyze the DOM structure of the pages, locate elements, and obtain structured information. The advantage of these crawlers is that they can accurately obtain the information we need, but once the DOM structure of the page changed, the code may need to be rewritten.
4) Incremental Crawler: The Web is dynamic and data on the webpages keep on changing frequently. These crawlers are used to maintain the index database up-to-date. Badawi (2013) proposed a method to reduce the network traffic and keep the index up-to-date.

2.2 Reference components

In order to adapt to the acquisition of massive agricultural data, we must consider not only the efficiency of Web crawlers but also the applicability of agricultural market websites. The three crawlers in this study combine asynchronous processing capabilities of Scrapy, distributed and data-persistent functionality of Redis, dynamic page rendering functionality of Splash. All of them are efficient dynamic page crawlers.

(1) Asynchrony

Scrapy is an application framework for crawling websites and extracting structured data. It is written with Twisted, a popular event-driven networking framework. Thus, it is implemented using a non-blocking (aka asynchronous) code for concurrency. Fig. 1 shows an overview of the Scrapy architecture and an outline of the data flow that takes place inside the system (shown by the arrows).

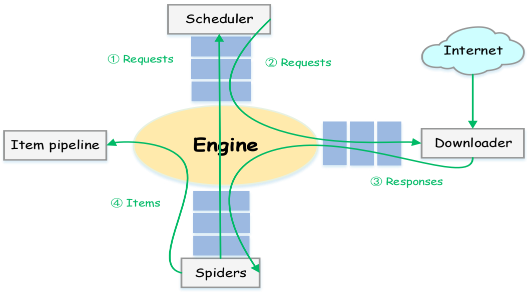


Fig. 1. Data flow diagram of Scrapy

(2) Distribution and Data-persistent

As shown in Fig. 2, we call own core server as master, call machine which is used to run the Web crawlers as slave. First, we build a Redis database on the master and open up a separate list for each type of website that needs to be crawled. By setting Scrapy-Redis on the slaves, master URLs are inherited by slave URLs. Due to the queue mechanism of Scrapy-Redis, links acquired by slaves do not conflict with each other. After each slave has completed the crawling task, the obtained results will be aggregated to the server.

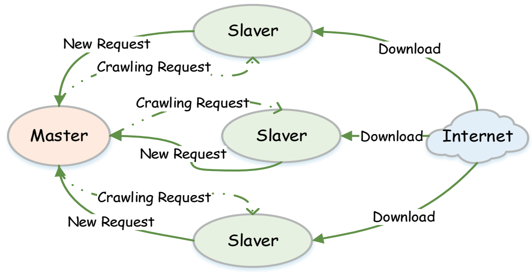


Fig. 2. Data flow diagram of Scrapy-Redis

(3) Dynamic page rendering

Currently, in order to speed up the loading of webpages, many parts of the page are generated by JavaScript. It becomes a big problem for Scrapy to crawl dynamic pages generated by JavaScript without a JavaScript engine. In other words, only static pages could be crawled by Scrapy. Splash is a JavaScript rendering service which needs to be installed in Docker. It is a lightweight Web browser with an HTTP API, implemented in Python using Twisted and QT5. The (twisted) QT reactor is used to make the service fully asynchronous allowing to use webkit concurrency via QT main loop.

3. FOCUSED CRAWLER: AGRICULTURAL DATA

Agricultural market websites can find a lot of valuable information. Though there are a large number of pages on the Internet, only a few are our concern. The goal of universal crawlers is to crawl as many pages as possible. In this process, they do not care much about the order of page acquisition and theme of the acquired pages. This will consume a lot of system resources and network bandwidth, and the consumption of these resources cannot be in exchange for higher utilization of the acquisition page. However, focused crawlers analyze the hyperlinks and download Web content according to the theme given in advance. They predict the next URL to be crawled and the relevance of the current webpage to ensure crawling and downloading theme-related pages as much as possible, unrelated pages as few as possible. The focused crawler adds a sorting module (Cho 1998), and a theme establishment module based on the universal crawler. This study adopts focused crawler technology to acquire agricultural market information from agricultural information publishing websites and agricultural news websites. Fig. 3 is the basic flow of a focused crawler, and the process is as follows:

1) The crawl module acquires webpages;
2) The correlation analysis module analyzes the relevance of the webpage;
3) The crawl module performs corresponding processing according to the different results of the analysis;
4) The sorting module sorts the crawl queue by webpage weight;
5) The crawl module obtains a waiting URL from the crawl queue and continues to execute.

6) Loop to the first step until the crawl queue is empty.

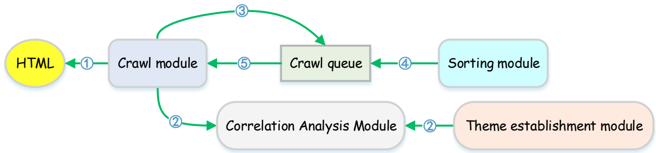


Fig. 3. Basic flow of a focused crawler

3.1 Thematic similarity judgment

This study uses the thematic similarity judgment method of Best First Search. The idea of this method is to analyze and sort the crawl queue, and crawl the best URL preferentially. A keywords set of agricultural market information has been designed to describe crawling theme. The weight of each keyword is represented by TF*IDF.

TF (Term Frequency) refers to the frequency of occurrence of a word in text, is used to increase the weight of keywords. It is defined as

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}. \tag{1}$$

$n_{i,j}$ refers to the number of occurrences of word t_i in file d_j. $\sum_k n_{k,j}$ refers to the sum of the occurrences of all words in the file d_j.

IDF (Inverse Document Frequency) is used to reduce the theme weight of public words. It is defined as

$$idf_i = \log \frac{|D|}{\left|\left\{j : t_i \in d_j\right\}\right| + 1}. \tag{2}$$

$|D|$ is the total number of files in the corpus. $\left|\left\{j : t_i \in d_j\right\}\right|$ is the number of documents containing the word t_i.

TF*IDF represents the weight of the word, is used for weight sorting and obtain Top-N as the keywords of the text. It is defined as

$$w_{i,j} = tf_{i,j} \times idf_i. \tag{3}$$

The relevance of the theme is calculated using the vector space model algorithm. Take the number of keywords n as the dimension of the vector space. The weight w_i of each keyword k is the value of each dimension. f_{kq} refers to the weight of the keyword k in the theme q. The theme is defined as

$$\alpha = \sum_{k \in q} f_{kq} = \left(w_1, w_2, \cdots, w_n\right). \tag{4}$$

Analyze page p, count the frequency of keyword occurrences, and find the ratio of frequencies. The keyword with the highest frequency of occurrence is used as a reference, and its frequency is represented by $x_i = 1$. Through the frequency ratio, find the frequency x_i of other keywords. The value of each dimension of this page's corresponding vector is represented by $x_i w_i$. The page theme is defined as

$$\beta = \sum_{k \in p} f_{kp} = \left(x_1 w_1, x_1 w_2, \cdots, x_1 w_n\right). \tag{5}$$

Refer to (6), the theme relevance of pages is calculated by using Harvest rate (the ratio of the number of theme-related pages to the total number of extracted pages).

$$\begin{aligned} Sim(q,p) &= \frac{\sum_{k \in q \cap p} f_{kq} f_{kp}}{\sqrt{\sum_{k \in q} f_{kq}^2}\sqrt{\sum_{k \in p} f_{kp}^2}} = \cos\langle \alpha, \beta \rangle \\ &= \frac{x_1 w_1^2 + x_2 w_2^2 + \cdots + x_n w_n^2}{\sqrt{w_1^2 + w_2^2 + \cdots + w_n^2}\sqrt{x_1^2 w_1^2 + x_2^2 w_2^2 + \cdots + x_n^2 w_n^2}} \end{aligned} \tag{6}$$

Set the threshold r according to the actual situation. When $\cos\langle \alpha, \beta \rangle \geq r$, it can be considered that the page and the theme are related.

3.2 Keyword multimode matching

Aho (1975) invented a string searching algorithm based on finite state automata (FSA) multi-pattern matching algorithm called Aho-Corasick. The Aho-Corasick algorithm constructs a finite pattern matching state machine for all keywords to be matched before matching and only needs to scan the text once to complete the matching work. The usual string searching algorithm needs to fall back to the start position every time a match is made, resulting in very low execution efficiency. The Aho-Corasick algorithm can avoid the waste of efficiency caused by rollback. For this reason, it was used in this study to match keywords. The algorithm needs to implement three functions during its execution: Goto, Output, and Fail. The pseudocode is as follows:

```
q := INIT_STATE;  // root
for $i$ := 1 to $m$ do
        while $g(q, T[i]) = \emptyset$ do
        $q := f(q)$;   // follow a fail
        $q := g(q, T[i])$;   // follow a goto
        $node := q$;
        while $node \neq$ root do
        if $flag(node) \neq \emptyset$ then print $i$, out($node$);
        $node := f(node)$;   // backtracking
end for;
```

T refers to the target string and its length is *m*. *q* refers to the node pointer of the dictionary tree. *g* returns the next node pointer that arrives from node *q* through path *T*[*i*]. *f* returns the node's backtracking node pointer. *flag* determines if the node is a flag node.

3.3 Theme crawling process

First, we give three definitions as follows:

(Def 1) Theme: $T = \left\{k_1, k_2, k_3, k_4, \ldots\right\}$, k_i refers to keyword;

(Def 2) Theme-Weight table: $K - W = \{<\mathrm{k}_1, w_1>, <\mathrm{k}_2, w_2>, \ldots\}$, w_i refers to the corresponding weight of the keyword;

(Def 3) Crawl queue: $Q = \{<\mathrm{u}_1, r_1>, <\mathrm{u}_2, r_2>, ...\}$, u_i refers to URL, r_i refers to the relevance score.

Fig. 4 is the flow chart of the focused crawler in this study, and the process is as follows:

1) Initialize, define topic set *T*, threshold, crawl depth, desired number, initial URL set, etc.
2) If the crawl queue *Q* is not empty, then take out an URL and put it in the splash for dynamic rendering and get the HTML text.
3) Extract all the href attributes from the pages in the initial URL set, perform URL normalization, and finally add them to the crawl queue.
4) Obtain the cleaned text by encoding, decoding, denoising the pages in the non-initial URL set and extract the text.
5) Make a thematic similarity judgment on the extracted text and store theme-related pages.
6) Normalize and deduplicate URLs on the new page, then add to the crawl queue.
7) Prioritize the crawl queue by the *K-W* table.
8) Repeat until the crawl queue is empty, or reach the maximum depth, or reach the desired number.

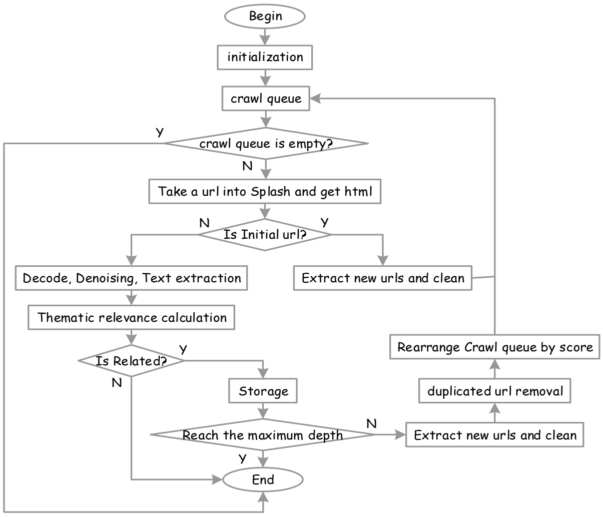

Begin
initialization
crawl queue
Y
crawl queue is empty?
N
Take a url into Splash and get html
N
Is Initial url?
Y
Decode, Denoising, Text extraction
Extract new urls and clean
Thematic relevance calculation
Y
Is Related?
N
Rearrange Crawl queue by score
Storage
duplicated url removal
N
Reach the maximum depth
Extract new urls and clean
Y
End

Fig. 4. Flow chart of the focused crawler

4. INCREMENTAL CRAWLER: CORN PRICE DATA

Corn is an important food crop, and corn prices are a direct reflection of market development and supply balance. By extracting and analyzing market price information, it not only can visually demonstrate the development trend of the corn trading market, but also is conducive to providing data for structural reform on the supply side, providing theoretical basis for relevant departments to formulate production development decision-making, providing data for research on pricing mechanism of agricultural products, monitoring, early warning, and forecasting.

In this study, corn price data was crawled from the official website of agriculture and large wholesale market sites according to specific rules for crawling. After cleaning and integrating, it was stored in the database. Then periodically revisit the site based on the site update cycle and obtain the incremental content based on the time attribute. These websites do not notify the Web crawler of new changes, so we need to periodically poll the sources to maintain the copies up-to-date. Cho (2003) studied how to maintain local copies of remote data sources "fresh," when the source data is updated autonomously and independently. The incremental crawler strategy of this study can be applied to construct agricultural products price data warehouses and data marts.

4.1 Crawling strategy

As shown in Fig. 5, the process is as follows:

1) Initialize the seeds queue to be crawled.
2) Add the seeds queue to the crawl queue.
3) Extract a URL from the crawl queue and send it to splash, send http request, and render the page dynamically.
4) Pass the HTML text returned by the response to the parser, use the CSS selector, regular expression, Xpath, etc. to perform element positioning and extraction on the newly added information.
5) Send the parsed text to pipelines for data cleaning and multi-source data integration.
6) Store the processed data in the database.
7) Parse new URLs that meet the requirements and add them to the crawl queue, and enter the new round of crawling.
8) Exit the program until the crawl queue is empty.
9) Periodic revisit by site update cycle.

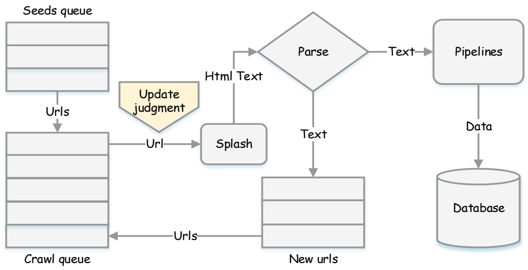


Fig. 5. Data flow diagram of corn price crawler

4.2 Web mining and visualization

This study visualized the corn price crawled from a wholesale market website in the past 800 days by plotting a K-line chart (Fig. 6 left) to display the price fluctuations. By plotting a time-price scatter plot, we found that a Non-linear regression model can be used for short-term trend predictions.

Assuming that *x* refers to independent variable, *y* refers to dependent variable, and ε refers to the sum of the effects of various random factors *y* (random disturbance terms), $\varepsilon \sim \mathrm{N}\left(0, \sigma^2\right)$, β_i refers to the regression coefficient. Then the nonlinear regression curve model can be defined as:

$$\mathrm{y} = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \varepsilon. \tag{7}$$

Make $x_i^2 = x_i', \cdots$ in (7), then the equation changes to (8) and the matrix form refers to (9).

$$\mathrm{y} = \beta_0 + \beta_1 x + \beta_2 x' + \cdots + \varepsilon \tag{8}$$

$$Y = XB + \varepsilon \tag{9}$$

We can get (10) by using the least squares method.

$$\min\left(E'E\right)=\left(Y-XB\right)'\left(Y-XB\right) \tag{10}$$

Then the estimated value of regression coefficient vector can be defined as

$$\hat{B}=\left(X'X\right)^{-1}\left(X'Y\right). \tag{11}$$

Finally, we drew a regression fitting curve to forecast the short-term price trend. (Fig. 6 right)

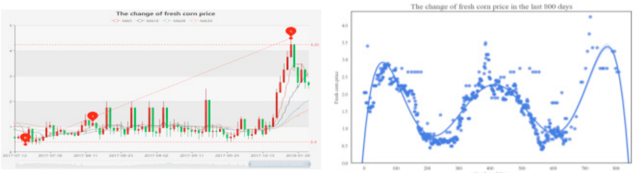


Fig. 6. The change of fresh corn price in the last 800 days

5. CUSTOM CRAWLER: E-COMMERCE DATA

Due to the rapid development of information technology and e-commerce, more and more people are beginning to enjoy the convenience of online transactions. A large number of excellent online business portals are experiencing rapid growth of transaction volume. The resulting network data is also growing exponentially. It contains a lot of important sales-related information, such as market supply and demand information and potential user needs, etc. Therefore, it is of great significance to mine useful information in massive commodity data.

This study takes an e-commerce website in China as an example to design a crawler to crawl information whose keywords is "agricultural products" and obtain the product selling price, sales volume, and shop address. After cleaning and formatting, the data was stored in the database and used to analyze the sales of various products along the time series and the user's consumption scenarios.

5.1 *Crawling strategy*

The difference in crawling such websites is that we can analyze the URL pattern of webpages to be crawled in advance. Set a crawl queue once, and then crawl data like the universal crawlers. Refer to Fig. 7 for the process.

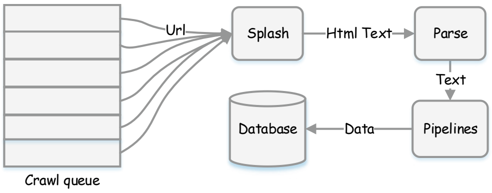


Fig. 7. Data flow diagram of e-commerce Web crawler

5.2 *Web mining and visualization*

We selected 4,000 acquired agricultural products information for mining and conducted the following visual analysis:

First, we segmented the product titles and established a set of stop words. Then we filtered each participle to eliminate unnecessary words. In order to make each segment of the titles unique, we removed the duplicates in the segmentation result. As shown in Fig. 8, the words we get were finally visualized. Through analysis, we found that coarse grain, food crops, rice, millet, batata, and shiitake appeared most on the e-commerce website. It can be seen that these products are more respected by the business.

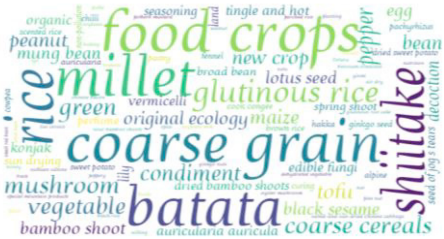
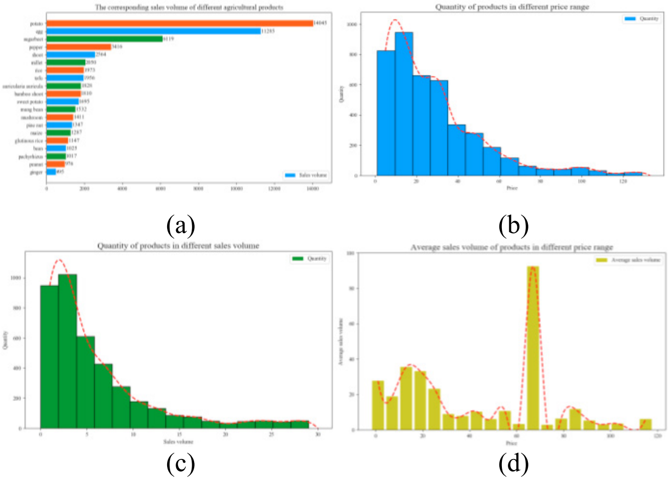


Fig. 8. The word cloud of product titles

We selected 20 products and calculated the corresponding sales volume of different agricultural products (Fig. 9.a). In order to make the visual effect more intuitive, here we chose products with a price below ¥130, and counted the number of products of different price ranges (Fig. 9.b). Then We chose products with the sales volume less than 30, and calculated the quantity of products (Fig. 9.c). In addition, we selected items priced below ¥120, and counted the average sales volume of products of different price ranges (Fig. 9.d).



(a) (b)

(c) (d)

Fig. 9. Bar charts of Quantity-Price-Sales

We selected products with a price below ¥200 and drew a scatter plot of product price and sales volume distribution (Fig. 10.a). In addition, we selected products with a price below ¥120 and plotted a diagram about the total product selling price of different price ranges. Then we found that a linear regression model can be built as

$$y_i = a + bx_i. \tag{12}$$

x_i refers to price, y_i refers to total selling price, *a* and *b* are regression coefficients, y_i is an estimate of the mean $E(y_i)$ of the dependent variable that corresponds to the value of the independent variable x_i. And we can get (13) by using the least squares method.

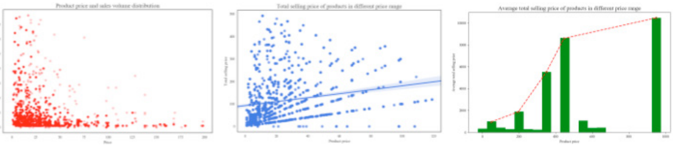
$$\sum\left(y_i - y_i\right)^2 = \sum\left(y_i - a - bx_i\right)^2 \tag{13}$$

The regression coefficients of the model are defined in (14) and (15).

$$\hat{b} = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{n\sum x_i^2 - \left(\sum x_i\right)^2} \tag{14}$$

$$\hat{a} = \frac{\sum y_i}{n} - \hat{b}\frac{\sum x_i}{n} \tag{15}$$

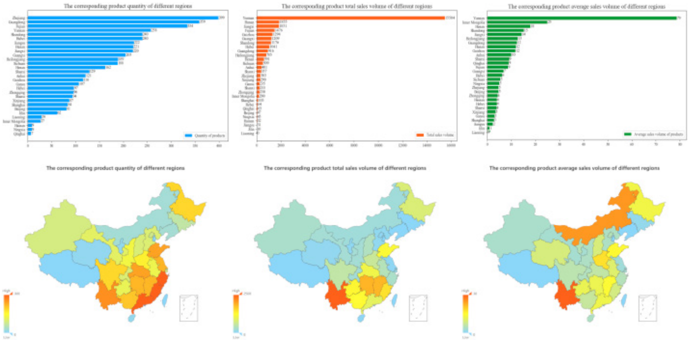
A diagram about the total product selling price of different price ranges with a linear regression was drawn by the above equations (Fig. 10.b). We further analyzed the average total sales, selected products with a price below ¥1000, and plotted a more intuitive diagram about average total product selling price of different price ranges (Fig. 10.c).



(a) (b) (c)

Fig. 10. Price and Income

We also counted the product quantity (Fig. 11.a), total sales volume (Fig. 11.b), and average sales volume (Fig. 11.c) of different provinces, and plotted the bar charts and heat maps.


The corresponding product quantity of different regions
The corresponding product total sales volume of different regions
The corresponding product average sales volume of different regions

(a) (b) (c)

Fig. 11. Regional distribution differences

We can conclude from these diagrams: (1) Potatoes are the hottest products on this e-commerce website. (2) Most of the products are between ¥10 and ¥20. (3) The sales volume of most products is between 2 and 4. (4) Products with a price of ¥65 to ¥70 are best sold. (5) The selling price is inversely proportional to the quantity and the sales volume of the product, and is proportional to the total selling price. (6) Active agricultural products merchants are concentrated in the southeast coastal areas. (7) The product sales volume of Yunnan is the highest in China.

6. CONCLUSIONS

This study applies Web mining technologies to agricultural big data. Three types of crawlers that support distributed and dynamic pages were designed to acquire data according to the massive data of agricultural market information. This study has improved some aspects of existing Web crawlers, making them more suitable for the acquisition of agricultural market information. And the acquisition results of agricultural information publishing websites, agricultural news websites, agricultural product marketing websites, and agricultural product e-commerce websites have proved this. In addition, the efficiency of the text extraction module and the theme relevance evaluation module in the focused crawler has also been effectively improved by using the Aho-Corasick algorithm to perform keywords matching. Through these data acquisition strategies, the acquired data was finally integrated and analyzed by Web mining and visualization technologies. Using key technologies of big data to mine agricultural market information, to a certain extent, can provide references for the construction of agricultural data warehouse, market information monitoring, agricultural products price forecasting and warning, farmer production decisions, agricultural academic researches, etc.

REFERENCES

Aho, A.V., and Corasick, M.J. (1975). Efficient string matching: an aid to bibliographic search. *Communications of the ACM*, 18(6), 333-340.

Badawi, M., Mohamed, A., Hussein, A., and Gheith, M. (2013). Maintaining the search engine freshness using mobile agent. *Egyptian Informatics Journal*, 14(1),27-36.

Chakrabarti, S., Van den Berg, M., and Dom, B. (1999). Focused crawling: a new approach to topic-specific Web resource discovery. *Computer networks*, 31(11-16), 1623-1640.

Cho, J., Garcia-Molina, H., and Page, L. (1998). Efficient crawling through URL ordering. *Computer Networks and ISDN Systems*, 30(1-7), 161-172.

Cho, J. (2001). Crawling the web: discovery and maintenance of large-scale web data. *A Thesis Nov*.

Cho, J., and Garcia-Molina, H. (2003). Effective page refresh policies for web crawlers. *ACM Transactions on Database Systems (TODS)*, 28(4), 390-426.

Kumar, M., Bhatia, R., and Rattan, D. (2017). A survey of Web crawlers for information retrieval. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 7(6).

Ramakrishna, M. T., Gowdar, L. K., Havanur, M. S., and Swamy, B. P. M. (2010, February). Web mining: Key accomplishments, applications and future directions. *In International Conference on Data Storage and Data Engineering (DSDE)*, 187-191.

Spetka, S. (1994, October). The TkWWW robot: beyond browsing. *In Proceedings of the second WWW conference*, 94.

Waldherr, A., Maier, D., Miltner, P, and Günther, E. (2017). Big data, big noise: the challenge of finding issue networks on the web. *Social Science Computer Review*, 35(4), 427-443.

Yu, H. L., Bingwu, L., and Fang, Y. (2010, July). Similarity computation of web pages of focused crawler. *In International Forum on Information Technology and Applications (IFITA)*, 2, 70-72).