Design of a Daily Brief Business Report Generator based on Web Scraping with KNN Algorithm

Gang Lin¹, Yanchun Liang¹, Xiaoyang Fu¹, Guangshun Chen² and Shaocan Cai²

¹Zhuhai Laboratory of Key Laboratory of Symbol Computation and Knowledge Engineering of Ministry of Education, Department of Computer Science and Technology, School of Computer Science, Zhuhai, 519041, Guangdong, China.
²Department of Computer Science and Technology, School of Computer Science, Zhuhai College of Jilin University, Zhuhai, 519041, Guangdong, China.

¹lingang19@jluzh.edu.cn

Abstract. In order to generate a report for an enterprise where there is neither the API supporting from their existing website systems nor the granted database access rights approval, a daily business report generator system based on web scraping with k nearest neighbor (kNN) classification algorithm is proposed in this paper. It covers the web crawler technology that is to access existing website system and extract business data. The kNN algorithm is applied to identify the verification code on the login page, and the brief daily report generating in a spreadsheet style grid. Compared with some OCR engine for image recognition, the system in Python can automatically generate the brief daily business reports by the kNN algorithm, which is better than some library with default training set on validating the verification code.

1. Introduction

A daily business report often assists managers to know about the daily situation of the enterprise, locate issues as soon as possible and make a sound business decision. It is a big data era, AI (artificial intelligence) era and Internet plus era, some modern enterprises introduced AI equipment and application with a competitive advantage. For these enterprises, daily business reports with a lot of colorful pictures, data grid can be generated very conveniently. However, as for some other enterprises, to build a daily report is a task of not only time consuming but needing more staffs to prepare data, collect information, write report, review report and so on.

In fact these enterprises often have their own website for usual official task assignment and some dynamic data or images show. These data or images are just what a daily report should contain. The website system is truly valid after a login success. Usually, staffs open the login page of the website system by the browser, input correct name, password and dynamic verification code, submit the form and then access data on other pages. It is possible that the website system seems a unique limited way to fetch data for daily report except that you want to pick up office phone, pen and notebooks. There is no published API since there might be not enough technical staffs to develop and maintain them, what is more, the existing database is too important for the whole system running to be used directly. Even if you apply for the database access rights, it always is disapproved with comments of a high risk of data modification or destruction, or some other similar. In that case, is there any other way to build a daily report? The answer is positive. Totally, there are 4 following key tasks to build a smart daily report generator system under the above circumstances.
1. Login the existing website
2. Identify verification code
3. Pass the login and extract some useful data
4. Generate daily business report.

2. Design on Business Report Generator System

2.1. Total Design

According to tasks above, the system consists of 4 parts, as shown in Fig. 1. The whole process is under the control of a global daily scheduler, which is triggered every work day, other holidays auto skipped. The report can be sent out to the manager only if the email address is configured in advance. The task #2 is a little bit more complicated, therefore it is listed as a separate item instead of a part of task #1 for easier illustration.

2.2. Web Scrapping

Besides daily report template, almost all of other data within daily report can be fetched from web pages with some necessary normalization and process. The process of extracting data from web pages is called as web scrapping, which is becoming increasingly useful as ever more information is available online [1]. Although web scraping is not a new term, in years past the practice has been more commonly known as screen scraping, data mining, web harvesting, or similar variations[2]. Web scrapping is used to solve task #1 and #3, prepare verification code screenshot for task #2, and data for task #4.

The program is required that systematically browses the World Wide Web in a methodical, automated manner, also known as a web spider, spider bot, or a web crawler. It is widely used in web indexing, automating maintenance tasks on a website, collecting specific types of data from web pages and so on. First of all, an initial list of URLs is required, also called the seeds. Along with visiting of seeds, a web crawler identifies all the hyperlinks and HTML code in the page, and add new URLs into the list to visit recursively. A web crawler crawls across the web, at the core is an element of recursion, and it retrieves page contents for a URL, examines that page for another URL, and retrieve that page [2]. These URLs are visited according to some policies.
Figure 2. Web crawler architecture for the System

The whole process will not stop until all URLs within the list are visited or other termination conditions are encountered such as all of useful data are extracted. Without separate verification code identification from page access, the web crawler architecture is shown in Fig. 2. Initially, only the seed set is contained within the URL frontier that include required URLs to traverse; as pages are fetched, their URLs are deleted from the frontier; for continuous crawling, the URL of a fetched page is added back into the frontier[3]. If all the URL addresses are with IP directly, *DNS should be ignored. From the professional point of view there are 3 basic crawler conventions to be followed [3].

1. Only one connection open to any host at a time
2. A few waiting minutes between successive requests to a host
3. Politeness restrictions

For plenty of enterprises, both business web server and client of daily report generator are within a company’s intranet. So the 3rd crawler convention is always no problem for the daily business report generator.

2.3. Verification Code Identification with KNN

The k nearest neighbor or kNN classification determines the decision boundary locally where k is a parameter [3]. The parameter k can be configured dynamically as different values for a scenario or some different scenarios. As for a scenario, different k may result in different result. That is to say, what k is sometimes depends on what result is expected, experience or knowledge.

The kNN is a kind of vector space classification method, which can deal with classes with non-spherical, disconnected or other irregular shapes well. On identifying a character within a verification code, the basic principle is that a test point has the same label as the surrounding points. kNN simply memorizes information of all characters in the training set and then compares the test character to them, which is also known as memory-based learning or instance-base learning[3]. kNN is also a supervised learning, which refers to working with a set of labeled training data[4]. It is also a kind of lazy learning [5]. For kNN, on the one hand, the larger training set is, the longer time comparison process is, the lower the efficiency is. On the other hand, if training set is too small to identify test verification code with a high recognition rate.

Figure 3. Example of identifying verification code by kNN
Preprocess of verification code image is the base of following image processing. After that, any pixel point within a character of identification code should only be one of black or white, no other values. The verification code with kNN is an instance of linear classifiers, 2-class classifiers. As shown in Fig. 3, 1 means black, 0 means white, what should the pixel point be which is marked as a question? The kNN algorithm may mark it as 1 since there are 3 neighbors marked as 1.

The process of verification code includes following steps. First of all, fetch the verification code from the login page on website. Second, read and preprocess verification code such as gray scale processing, noise eliminate, etc. Third, do image segmentation. Forth, identify characters within verification code. Finally, output and use the identification result.

2.4. Daily Report Building

Once the identified verify code is used on the login page, other pages can be accessed. Daily report can be built by extracting and using associated data from different pages. Different reports share a common spreadsheet template. Without the daily report generator, the boring task is copying certain data from one spreadsheet or website and pasting it into another one [6].

3. Implementation of a daily report generator

There are many program languages with different IDE tools to implement the daily report generator according to the design. Although Python is selected, C++, C#, Java, or some other languages can also be used. Here are some main steps listed in table I.

| No. | Item |
|-----|------|
| 1   | 1. Login website |
|     | 1.1 New web driver |
|     | 1.2 Access the login page |
|     | 2. Identify verification code |
|     | 2.1 Snapshot of current login page with identification code, and save image |
|     | 2.2 Crop verification code from snapshot, save. |
|     | 2.3 Read verification code file |
|     | 2.4 Grey scale processing |
|     | 2.5 Noise eliminate |
|     | 2.6 Locate boundary position for each character |
|     | 2.7 Image segmentation |
|     | 2.8 Identify characters within verification code |
| 2   | 3. Pass the login and Extract some useful data |
|     | 3.1 Login successfully |
|     | 3.2 More Data to extract? If yes, go to 3.3, otherwise, go to 4 |
|     | 3.3 Access new pages with URL list update |
|     | 3.4 Extract data from pages, go back to 3.2 |
| 3   | 4. Generate daily business report |
|     | 4.1 Fill data into template |
|     | 4.2 Output the daily report |

3.1. Login Website

| No | Item |
|----|------|
| 1  | Set Options for auto browser |
| 2  | Initialize URL list with URL of the login page |
| 3  | New web driver to integrate with selenium |
| 4  | Access the login page |

It is necessary to exploit a powerful web scraping tool as website access. Similar to auto website testing, writing codes to input username and password according to develop engineer’s indication. In
python selenium works by automating browsers to load the website, retrieve the required data, and even take screenshots or assert that certain actions[2], which is just what we want. Without its own web browser, selenium integrates with 3rd party browsers to run. Table II lists the logic of access login page.

3.2. Identify verification code
Although the intranet limits users who use the website, adding the verification code on the login page is always a valid measure to prevent unauthorized users to use brute-force attack to crack the administrator password. The verification code is different on each login, the brute-force attack becomes difficult and invalid. The verification code is widely used in the intranet network as well the Internet network. Even each request to verification code generating program, different verification code will be returned. That is to say, as for web scrapping, there is hardly other way to get the same verification with login page except for identifying the screenshot. So the first step of identifying verification code is to make a screenshot of login page, then extract value from it. The identified verification code will be typed automatically by the daily report generator and submit the login page.

The screenshot includes the verification code in a fixed area, so the verification code can be cut according to the position. For the further research about identifying verification code in the future from the verification code images instead of original website on, saving images of verification code are proposed. The benefit is from the loose coupling design [7, 8].

| No. | Item                                                                 |
|-----|----------------------------------------------------------------------|
| 1   | If standard library is not initialized, initialize it at the first time. |
| 2   | Score the similarity from 0 to 1 for each labeled item within standard library, 1 means complete matching, 0 for no match. |
| 3   | Sort all similarity scores in the reverse order, the first one is the biggest value, and its label is the character value. |

Table 3. Logic of character recognition

There are some open source OCR (Optical Character Recognition) engine libraries for identifying the whole verification code image without post-processing, which is a process to extract text from images [1]. The open source Tesseract OCR engine can be used and run after additional training. However, default training data set does not work well for our verification code images with a low recognition rate less than 50%. There are 2 choices, one is adding more test data into the training set, the other one is to develop the engine with more recognition rate. At the same time, a more convenient method to add training examples is wanted, so the latter one is the final selection.

Like consulting a dictionary for a new word, could we build a standard image library with labeled information? If the standard image library is built, the verification code identification issue is changed to identify each character after being extracted from the verification code image. The table III lists the logic about identifying characters within verification code. After all characters are identified, the whole verification code is recognized. The core process for identifying verification code is as shown in Fig. 4.
Ten standard folders are created for verification code with numbers only, named from 0 to 9, each folder contains the standard images labeled with the folder name, as shown in Fig. 5. Therefore, we can add standard image beyond codes. What’s more, auto building more standard images can be included easily with the limit of suit size of kNN training set. This kind of supervised learning method makes training set build very flexible based on file system.

There are 5000 random samples collected, where the first 200 verification codes are as training data, others are test data. These verification code images are converted to black and white pictures, as shown in Fig. 6, which is main to reduce the complexity of identifying process, since the black and white images only contains 2 kinds of color, only 0 and 1 can map them, much less than RGB.

For the convenience of check the result of identifying beyond the website environment, the same folder structure is used for identifying result. Besides, the result text file is also output, which can be compared with the result of Tesseract OCR engine, as shown in Fig. 8. All 4800 verification codes can be identified correctly by the daily report generator system, the correct rate is up to 100%, bigger than Tesseract with default training set only 443/5000 (13/200 for training set and 430/4800 for test set).
3.3. Extract Data to Generate Business Report

HTML code parser can find the element of web pages by the target name, tag name, class or combined of them. Data extracted can be stored in memory, disk, or the report directly.

4. Conclusion

Combination of kNN algorithm and web scraping, a good trend can be obtained to generate enterprise daily report, especially for those without API support and database rights. For identifying verification code, each character image is placed according to its label name, which is used to manager standard library and review the final results manually. The kNN algorithm display better performance than that of some library with default training set on validating the verification code.

Acknowledgments

This work was financially supported by Premier-Discipline Enhancement Scheme supported by Zhuhai Government (Grant No.2015YXXK02) and Premier Key-Discipline Enhancement Scheme supported Guangdong Government Funds (Grant No.2016GDYSZDXK036).

References

[1] Lawson, R. (2015). Web Scraping with Python. Packt Publishing Ltd., Birmingham, UK.
[2] Mitchell, R. (2018). Web Scraping with Python, 2nd ed.. No Starch Press, Sebastopol, CA, USA.
[3] Manning, C. D., Raghavan, P., & Hinrich Schütze. (2010). Introduction to Information Retrieval. Posts & Telecom Press, Beijing, RPC.
[4] Bell, J. (2015). Machine learning: hands-on for developers and technical professionals. John Wiley & Sons, Inc., Indianapolis, Indiana, USA.
[5] Zhihua Zhou. (2016). Machine Learning. Tsinghua University Press, Beijing, RPC.
[6] Sweigart, A. . (2015). Automate the boring stuff with Python: practical programming for total beginners. No Starch Press, San Francisco, CA, USA.
[7] Fowler, M. (2001). Reducing coupling. IEEE Software, 18(4), 102-104.
[8] Bieman, M. N. J. M. (2001). Coupling of design patterns: common practices and their benefits. Compsac, 574.
[9] Richard Lawson, Web Scraping with Python, Packt Publishing Ltd., Birmingham, UK, 2015, pp. v, 96.
[10] Ryan Mitchell, Web Scraping with Python, 2nd ed., No Starch Press, Sebastopol, CA, 2018, pp. ix, 33, 166.
[11] Christopher D. Manning, Prabhakar Raghavan, Hinrich Schutze, Introduction to Information Retrieval, Cambridge University Press, Sebastopol, CA, 2008, pp. 273-276, 405-419.
[12] Jason Bell, Machine Learning: Hands-On for Developers and Technical Professionals, John Wiley & Sons, Inc., Indianapolis, 2015, pp. 3.
[13] Al Sweigart, Automate The Boring Stuff With Python, No Starch Press, Inc., San Francisco, CA, 2015: 265-293.
[14] Martin Fowler, Reducing Coupling, IEEE SOFTWARE, 2001:101-103.
[15] William B McNatt, James M Bieman, Coupling of Design Patterns: Common Practices and Their Benefits, computer software and applications conference, 2001:1-6.Conference Name:ACM Woodstock conference