Development and Application of a Document Delivery Robot Based on the Selenium Framework

Authors: Shen Jinian, Li Jing, Liu Xinxia, Shen Jinian

Date: 2023-12-28T00:00:00+00:00

Abstract

Purpose/Significance The author is engaged in document delivery services at the Jiangsu Provincial Engineering Literature Information Center platform, where mechanically repetitive operations constitute a significant proportion, severely encroaching upon working hours. Furthermore, literature requests occur around the clock, and as a librarian on duty, document delivery can only be performed within limited time frames, making it difficult to guarantee delivery timeliness. Therefore, the author attempted to develop a document delivery robot to automate the completion of document delivery tasks. Method/Process] This paper takes the Jiangsu Provincial Engineering Literature Information Center platform as an example, designs and develops a document delivery robot based on the Selenium framework, and implements the entire process of logging in, checking in, obtaining orders, searching, downloading, uploading, and then continuing to wait for new orders after completing document delivery. Results/Conclusion] Through analysis of one month of operation, although the document delivery robot cannot completely replace manual delivery, it can effectively address the large proportion of mechanically repetitive work in the document delivery process, improve delivery efficiency, and simultaneously extend service hours, allowing delivery librarians to devote more energy to solving difficult-to-locate documents such as special literature. Finally, the document delivery robot also has reference value for the document delivery field and other related scenarios.

Full Text

Development and Application of a Document Delivery Robot Based on the Selenium Framework: A Case Study of the Jiangsu Engineering Technology Literature Information Center

Shen Jinian[1], **Li Jing**[1], **Liu Xinxia**[2]
[1](Library of China Pharmaceutical University, Nanjing 211198, China)
[2](Library of Nanjing University of Finance and Economics, Nanjing 210023, China)

Abstract

[**Purpose/Significance**] The authors are responsible for document delivery services on the Jiangsu Engineering Technology Literature Information Center platform, where mechanically repetitive operations account for a large proportion of tasks, severely encroaching on work hours. Moreover, literature requests occur 24 hours a day, yet as on-duty librarians, we can only process deliveries during limited time periods, making it difficult to guarantee timely delivery. Therefore, we attempted to develop a document delivery robot to automate the entire document delivery workflow. [**Method/Process**] This paper takes the Jiangsu Engineering Technology Literature Information Center platform as a case study, designing and developing a document delivery robot based on the Selenium framework to automate the complete process of logging in, checking in, obtaining orders, searching, downloading, uploading, and then waiting for subsequent order assignments. [**Result/Conclusion**] Through one month of operational analysis, although the document delivery robot cannot completely replace manual delivery, it can effectively address the large proportion of mechanically repetitive tasks in the document delivery process, improve delivery efficiency, and simultaneously extend service hours, allowing delivery librarians to devote more energy to locating difficult-to-find materials such as special documents. Finally, the document delivery robot offers valuable insights for the document delivery field and other related scenarios.

Keywords: Document Delivery; Document Delivery Robot; Selenium; Jiangsu Engineering Technology Literature Information Center
Classification Number: G252.6

Document delivery refers to a service model where libraries or other information institutions transfer literature materials to users according to their requests. Various delivery methods exist, including postal mail, email, and online transmission. Currently, document delivery has become an essential component of library digitalization initiatives, with numerous university and public libraries offering such services while continuously strengthening the construction and updating of digital literature resources and improving their service offerings.

1. Research Status

Extensive research has been conducted on document delivery services, primarily focusing on service models, workflows, and service quality. In 2014, Xu Chun et al. introduced the service model and workflow of the "Jiangsu Engineering Technology Literature Information Center Platform" in their study on the joint reference consultation service model based on a buzzer mechanism [1]. In the same year, Xu Chun et al. also provided a detailed analysis of the platform's operation from perspectives including document delivery volume, user demand, resource distribution, platform operation mode, and librarian contribution levels in their empirical study on regional joint reference consultation platform service models [2].

Service quality represents one of users' foremost concerns in document delivery. Much research has concentrated on improving response speed and accuracy, as well as enhancing user satisfaction and overall service quality. In 2014, Qin Xia conducted interviews with 21 senior document delivery staff in her study on user evaluation and behavior across multiple platforms [3], summarizing factors that influence user evaluations and analyzing their correspondence with four types of user behaviors: platform selection, information retrieval, request submission, and document acquisition. The study concluded with recommendations for document delivery platform management institutions. In 2020, Lu Yao et al. introduced association analysis mining techniques to analyze document delivery request data from different disciplinary and institutional user groups, enabling effective and targeted literature recommendation services [4]. In 2019, Xu Feng et al. investigated the underlying causes of pseudo-demand in document delivery and proposed countermeasures to reduce such occurrences, aiming to improve service efficiency and help users obtain literature resources more effectively [5].

With the rapid development of internet technology, emerging technologies have been applied to document delivery services, improving not only efficiency and quality but also expanding service scope and depth. In 2019, Zhu Yuqiang developed a document delivery robot using the WeChat public platform—without building third-party servers or utilizing advanced WeChat interfaces—that enables users to obtain full-text documents by simply sending a message [6]. However, research on such innovative applications remains relatively scarce. Therefore, we attempt to apply the Selenium framework to document delivery work, using Python to call its interfaces and achieve automation of document delivery services.

2. Introduction to the Jiangsu Engineering Technology Literature Information Center Platform

With the advancement of internet technology, various document delivery systems have emerged in China. These systems leverage internet-based distributed technology to enable unified management while facilitating tiered access for

individual libraries and convenient submission and retrieval for users. Representative systems include the China Academic Library & Information System (CALIS) and the China Academic Social Sciences and Humanities Library (CASHL), which primarily serve university users. In addition to these national systems, regional library alliances have developed numerous delivery systems, with typical examples being the Beijing Academic Library & Information System (BALIS) and the Jiangsu Engineering Technology Literature Information Center.

The "Jiangsu Engineering Technology Literature Information Center" (hereinafter referred to as "the Platform"), established in 2004, is one of Jiangsu Province's four major public science and technology infrastructure service platforms and serves as a regional literature information guarantee service platform for scientific and technological innovation. The platform integrates existing engineering technology literature information resources from ten institutions across Jiangsu's science and technology, culture, and education systems, including the Provincial Institute of Scientific and Technical Information, Provincial Academy of Agricultural Sciences Information Institute, Provincial Technical Supervision Information Institute, Nanjing Library, Nanjing University, Southeast University, Nanjing Agricultural University, China Pharmaceutical University, Nanjing Medical University, and Nanjing Tech University. Through resource sharing cooperation with the National Science and Technology Library and the Yangtze River Delta regional literature information resources, the platform constructs a literature information resource guarantee service system based on shared knowledge, resources, and construction, providing joint services to the entire province.

The document delivery process involves three parties: users with literature needs, the document delivery platform, and on-duty librarians. The basic framework is illustrated in Figure 1 [Figure 1: see original paper]. Users submit literature requests, which the platform distributes to on-duty librarians. Librarians retrieve document information and conduct searches based on their professional knowledge and available resources. After obtaining the resources, they submit the documents through the platform, which ultimately presents them to users.

To ensure efficient operation, the platform implements an "automatic order dispatch" system that automatically assigns orders to on-duty librarians who queue to receive them on a first-come, first-served basis with time-limited completion. Participating librarians log into the platform and click the "check-in" button to join the service queue sequentially. User-submitted orders form an order queue, and the system assigns orders from this queue to librarians in the service queue, popping up an order assignment reminder window on the webpage. Upon seeing the reminder, librarians can click "accept" or "abandon." Clicking "accept" moves the librarian to the end of the service queue, while clicking "abandon" assigns the order to the next librarian in line and moves the abandoning librarian to the queue's end. To maintain stable, orderly, and efficient operation, the platform limits Chinese order completion to 5 minutes and foreign language

orders to 15 minutes. Orders abandoned by three different individuals become difficult orders [7].

Currently, document delivery relies on on-duty librarians, and their processing timeliness determines delivery effectiveness. Literature demand occurs 24 hours a day, yet librarians can only provide service during limited hours, making timely delivery difficult to guarantee. Librarians engaged in document delivery generally work part-time, and prolonged delivery tasks encroach upon their work time. Moreover, most document delivery tasks involve mechanical repetition that can be completely automated through programming. These considerations constitute the design rationale for the document delivery robot.

3. System Design and Implementation

3.1 Selenium Framework Selection

Selenium is an automated testing tool that can drive browsers to perform specific actions such as opening webpages and extracting data. Automation code calls the Selenium framework to create corresponding HTTP requests and send them to the webdriver browser driver, which parses the requests and controls the browser. The execution process is illustrated in Figure 2 [Figure 2: see original paper].

Compared with mainstream web scraping tools such as BeautifulSoup and Scrapy, Selenium differs fundamentally in that it is a tool for testing web applications. By calling browser driver programs, Selenium simulates actual user operations. Scraping tools, conversely, primarily collect data by constructing HTTP request headers to crawl specified pages. Websites with strict anti-scraping measures can directly identify and block such scrapers. Journal databases implement particularly strict anti-scraping measures, making scraping tools easily restricted. Selenium is not a scraping tool but a simulation testing framework that runs directly in browsers to mimic human operations. Its access frequency and methods do not impose additional burdens on target servers and are less likely to be restricted by target websites, making it ideal for websites requiring user interaction, such as login and submission pages.

The primary research problem addressed in this paper is automating the entire document delivery workflow, including obtaining tasks from the platform website, searching multiple retrieval sites, downloading documents, and uploading them. Therefore, the Selenium framework represents the optimal choice.

3.2 System Function Design

Based on the platform's document delivery workflow, the document delivery robot must implement the entire process of platform login, check-in, queuing, order acquisition, searching, downloading, uploading, and re-queuing, all without human intervention and fully automated. The detailed workflow is illustrated in Figure 3 [Figure 3: see original paper].

3.3 Function Modules

(1) Initialization and Platform Login
Initialization primarily involves setting up the program's runtime environment, including logging, download paths, runtime environment paths, order dictionaries, browser drivers, and pre-opened working windows. The platform requires librarian login to receive tasks, so platform login functionality must be implemented after program initialization.

(2) Check-in
After logging into the platform, on-duty librarians must check in and queue for order assignment. For first-time order acceptance or subsequent orders after delivery completion, the system must first detect check-in status. If not checked in, the process initiates check-in; if already checked in, it directly enters the queue to wait for order dispatch. A function is defined to retrieve check-in status, returning the status upon successful acquisition or False upon failure. Key code implementation:

```
def check_{{{sign}}_{{status}}}():
    locator = (By.XPATH, '//*[@id="signBut"]/a')
    WebDriverWait(driver, 10, 0.3).until(EC.presence_{{{of}}_{{element}}}_{located}(locator)
    sign_{text} = driver.find_{element}(By.XPATH, '//*[@id="signBut"]/a').text
except Exception as e:
    return False
else:
    return sign_{text}
```

(3) Waiting for Order Dispatch
If already checked in, the order acceptance function waits for order dispatch using a retry function that attempts every 5 seconds until an order is obtained. Key code implementation:

```
@retry(wait_{fixed}=5000)
def order_{prompt}():
    driver.find_{element}(By.XPATH, "//*[@id='orderPrompt']/p[2]/a[1]").click()
except Exception as e:
```

If not checked in, the check-in function processes sign-in before waiting for order dispatch.

(4) Order Information Acquisition
Before obtaining orders, a dictionary is defined to store order information including order number, title, literature type, author, source, status, order acquisition time, and processing completion time. The dictionary header is:

```
header = ['Order Number', 'Order Title', 'Author', 'Literature Type', 'Source', 'Status', '
```

A function is then defined to retrieve order information. Key code implementation:

```
def delivery_{info}():
    delivery_{{{info}}_{{dict}}} = dict()
    locator = (By.XPATH, '//*[@id="deliveryinfo"]/div[2]/div[1]/h1/a')
    WebDriverWait(driver, 5, 0.5).until(EC.presence_{{{of}}_{{element}}}_{located}(locator))
    delivery_{{{info}}_{{dict}}}['Order Number'] = driver.find_{element}(By.XPATH,
        '//*[@id="deliveryinfo"]/div[2]/div[1]/h1/a').text
    delivery_{{{info}}_{{dict}}}['Order Title'] = driver.find_{element}(By.XPATH,
        '//*[@id="deliveryinfo"]/div[2]/div[1]/p/span').text
    delivery_{{{info}}_{{dict}}}['Literature Type'] = \
        driver.find_{element}(By.XPATH,
        '//*[@id="deliveryinfo"]/div[2]/div[2]/p[2]').text.split(": ")[-1]
    delivery_{{{info}}_{{dict}}}['Author'] = driver.find_{element}(By.XPATH,
        '//*[@id="deliveryinfo"]/div[2]/div[2]/p[4]/span').text
    delivery_{{{info}}_{{dict}}}['Source'] = driver.find_{element}(By.XPATH,
        '//*[@id="deliveryinfo"]/div[2]/div[2]/p[7]/a').text
except Exception as e:
    return False
else:
    return delivery_{{{info}}_{{dict}}}
```

(5) Literature Type Judgment

If an order is entirely in English or the literature type includes scientific achievements, policies and regulations, product samples, scientific reports, or laws and regulations, the order is directly abandoned and the current cycle ends; remaining orders proceed to the search process.

The method for identifying English literature checks whether the title contains only English characters. If the title is entirely English, it is treated as English literature and directly abandoned. Key code implementation:

```
def is_{{{contains}}_{{chinese}}}(strs):
    for _{char} in strs:
        if '\u4e00' <= _{char} <= '\u9fa5':
            return True
    return False
```

To accelerate order processing efficiency, orders with literature types including scientific achievements, policies and regulations, product samples, scientific reports, and laws and regulations are directly abandoned to avoid consuming search time. Key code implementation:

```
if (is_{{{contains}}_{{chinese}}}(title) is False) \
    or (delivery['Literature Type'] == 'Scientific Achievements') \
    or (delivery['Literature Type'] == 'Policies and Regulations') \
    or (delivery['Literature Type'] == 'Product Samples') \
    or (delivery['Literature Type'] == 'Scientific Reports') \
    or (delivery['Literature Type'] == 'Laws and Regulations'):
    delivery['Status'] = 'Abandoned'
```

```
    abandon()
    dict_{{{to}}_{{csv}}}(delivery)
    continue
```

(6) Searching and Downloading

After literature type filtering, orders are sequentially searched across four databases. If results are found and successfully downloaded, the process jumps to the upload workflow. If no results are found or results exist but cannot be downloaded due to permissions, the order is abandoned and the current cycle ends.

Using CNKI as an example, the CNKI advanced search page serves as the entry point. The process first selects "Title" from the dropdown menu, inputs the order title, uses "exact" matching by default, and finally clicks the "Search" button. The system checks for matching literature and downloads it if found; otherwise, it returns False. Key code implementation:

```
def get_{cnki}(delivery_{title}):
    locator = (By.XPATH, '/html/body/div[2]/div[3]/div[3]/div[2]/div[1]/div[9]')
    WebDriverWait(driver, 10, 0.3).until(EC.presence_{{{of}}_{{element}}}_{located}(locator)
    search = Select(
        driver.find_{element}(By.XPATH,
        '/html/body/div[2]/div[3]/div[3]/div[2]/div[1]/div[2]/span[2]/div/select'))
    search.select_{{{by}}_{{value}}}("TI")
    driver.find_{element}(By.ID, 'txt_1_{value1}').send_{keys}(delivery_{title})
    driver.find_{element}(By.XPATH,
        '/html/body/div[2]/div[3]/div[3]/div[2]/div[1]/div[9]').click()
except Exception as e:
    return False
else:
    locator = (By.XPATH, '//*[@id="gridTable"]/div/div[2]/table/tbody/tr/td[9]/a[1]')
    WebDriverWait(driver, 10, 0.3).until(EC.presence_{{{of}}_{{element}}}_{located}(locator)
    driver.find_{element}(By.XPATH,
        '//*[@id="gridTable"]/div/div[2]/table/tbody/tr/td[9]/a[1]').click()
except Exception as e:
    return False
else:
    sleep(5)
```

(7) Uploading

After successful document download, the program automatically switches to the platform's order upload page, clicks the "Upload Now" button, and completes the order upload. Since the platform's upload dialog calls the Windows system window, the upload implementation requires Python's Win32 library—a toolkit for developing Python applications in Windows that includes Win32API modules, extended types, and auxiliary tools, enabling developers to access low-level operating system APIs or write Windows GUI applications. Key code implementation:

```
def upload(file_{title}):
    locator = (By.XPATH, '//*[@id="deliveryinfo"]/div[2]/div[4]/a[1]')
    WebDriverWait(driver, 10, 0.5).until(EC.presence_{{{of}}_{{element}}}_{located}(locator)
    driver.find_{element}(By.XPATH, '//*[@id="deliveryinfo"]/div[2]/div[4]/a[1]').click()
except Exception as e:
else:
    sleep(2)
    locator = (By.XPATH, '/html/body/div[9]/div[1]/div[2]/div/div[1]/div/form')
    WebDriverWait(driver, 10, 0.5).until(EC.presence_{{{of}}_{{element}}}_{located}(locator)
    driver.find_{element}(By.XPATH, '/html/body/div[9]/div[1]/div[2]/div/div[1]/div/form').c
except Exception as e:
else:
    sleep(3)
    window = win32gui.FindWindow("#32770", "Open")
    ComboBoxEx32 = win32gui.FindWindowEx(window, 0, "ComboBoxEx32", None)
    ComboBox = win32gui.FindWindowEx(ComboBoxEx32, 0, "ComboBox", None)
    Edit = win32gui.FindWindowEx(ComboBox, 0, 'Edit', None)
    Button = win32gui.FindWindowEx(window, 0, 'Button', "Open(&O)")
    win32gui.SendMessage(Edit, win32con.WM_{SETTEXT}, None, file_{title})
    sleep(2)
    win32gui.SendMessage(window, win32con.WM_{COMMAND}, 1, Button)
except Exception as e:
else:
    sleep(30)
    return True
```

3.4 Source Database Selection

The platform's orders encompass ten primary literature types: journal articles, dissertations, patents, conference papers, scientific achievements, policies and regulations, product samples, scientific reports, laws and regulations, and monographs. Chinese literature is relatively concentrated, primarily sourced from CNKI and Wanfang Data (hereinafter referred to as Wanfang). Foreign literature is more dispersed, mostly from SpringerLink, ScienceDirect, Wiley, and smaller quantities from EBSCO, ACS, ProQuest, IEEE, and OXFORD JOURNALS.

Due to the diverse and dispersed sources of English literature, limited institutional subscriptions to foreign databases, and significant latency and error rates when accessing foreign databases from China, foreign database retrieval was temporarily abandoned.

Our institution's subscribed Chinese database sub-collections include journal articles, dissertations, patents, and conference papers. We lack download permissions for scientific achievements, policies and regulations, product samples, scientific reports, and laws and regulations. Therefore, Chinese literature processing includes only four categories: journal articles, dissertations, patents, and

conference papers. Orders for other literature types are directly abandoned.

In summary, considering our institution's database subscriptions, four source databases were selected: CNKI, Wanfang, Superstar Journals (hereinafter referred to as Superstar), and Chinese Medical Journals. CNKI and Wanfang have high overlap rates, covering journal papers, dissertations, conference papers, and patents. Chinese Medical Journals primarily include medicine-related literature, while Superstar serves as a supplement to the first three journal sources. The search sequence is therefore: CNKI, Wanfang, Superstar, and Chinese Medical Journals.

4. Operational Results and Analysis

4.1 Overall Operational Performance

The system has been in testing since November 2022 and has operated stably for one year. This paper analyzes order data from November 2023, with overall performance summarized in Table 1 .

In November, our account obtained 8,261 orders on the platform, completing 3,392 orders for a completion rate of 41.06%. The remaining 4,869 uncompleted orders fall into three categories. First, English literature: Due to unstable network speeds for foreign databases, retrieval and download frequently fail. Therefore, the system first identifies English literature upon order acquisition and immediately abandons such orders to proceed to the next order in queue. Actively abandoned English orders totaled 1,735, representing 21% of total orders. Second, unretrievable literature: When orders cannot be found in any of the four databases (CNKI, Wanfang, Superstar, and Chinese Medical Journals) after sequential searching, the delivery is abandoned and the system proceeds to the next order. Third, literature that can be retrieved but not downloaded: Some orders can be located but cannot be downloaded due to insufficient copyright subscriptions, resulting in abandoned deliveries.

Figure 4 [Figure 4: see original paper] illustrates the source database distribution for November's 8,261 orders: 23% were completed in CNKI, 11% in Wanfang, 1% in Superstar, 6% in Chinese Medical Journals, and 59% were abandoned after being unavailable across all four databases.

Daily order acquisition and completion rates for November are shown in Figure 5 [Figure 5: see original paper], with an average of 275 orders acquired and 113 orders completed daily, maintaining a stable completion rate of approximately 41%.

Order time distribution statistics for November appear in Figure 6 [Figure 6: see original paper]. Order demand concentrates between 8:00 and 22:00, with two peak periods at 10:00-11:00 and 15:00-17:00, aligning with researchers' work schedules. Additionally, Figure 6 [Figure 6: see original paper] demonstrates that order demand occurs 24 hours a day, and the document delivery robot can respond immediately during non-working hours, ensuring delivery timeliness.

Among the 1,735 English orders, 123 were abnormal orders that were directly abandoned without entering the processing workflow, thus consuming zero processing time. The processing duration for the remaining 6,403 normal orders concentrated between 41-81 seconds, with five peaks corresponding to the five search and download stages across four databases in the Figure 3 workflow: CNKI, Wanfang Journals, Wanfang Patents, Superstar, and Chinese Medical Journals. The platform requires order processing within 5 minutes; the longest order took 2 minutes and 30 seconds, meeting platform requirements while accelerating order processing and freeing up on-duty librarians' time.

5. Conclusion

Currently, the document delivery robot cannot completely replace manual delivery for two primary reasons: (1) User-submitted order information is often non-standard, with titles that do not perfectly match database retrieval results; and (2) Literature complexity requires manual verification—for instance, simple titles may retrieve multiple results. These situations prevent simple title matching from fully satisfying order requirements, necessitating human judgment and error correction to ensure successful completion of each order.

Nevertheless, document delivery robots play a crucial role in document delivery services by improving efficiency, reducing human resource investment, and extending service hours. We recommend that engineering literature centers improve order dispatch strategies by first assigning ordinary orders to institutional document delivery robots for automatic processing. Only after all institutional robots fail to retrieve the documents should orders enter the difficult-order pool for manual searching by librarians. This approach saves human resources, improves delivery efficiency, and allows delivery librarians to focus on difficult-to-find literature such as English documents rather than merely pursuing delivery volume. This example demonstrates that document delivery robots can be promoted to broader document delivery fields and related scenarios.

References

[1] Xu Chun, Yin Ming. Research on the joint reference consultation service model based on a buzzer mechanism—A case study of the "Jiangsu Engineering Technology Literature Information Center Platform"[J]. Library Science Research, 2014(3): 80-83.
[2] Xu Chun, Yin Ming. Empirical study on the service model of regional joint reference consultation platform—A case study of the "Jiangsu Engineering Technology Literature Information Center Platform"[J]. Library Theory and Practice, 2015(2): 102-105.
[3] Qin Xia. Research on user evaluation and behavior across multiple document delivery platforms[J]. Library and Information Service, 2014, 58(18): 41-49.
[4] Lu Yao, Liu Chunhong, Yang Daiqing. Research on user interest association in interlibrary document delivery services in Beijing universities[J]. Information

Studies: Theory & Application, 2020, 43(4): 101-107.
[5] Xu Feng, Zhou Xiuxia. Analysis of the causes of pseudo-demand in document delivery and countermeasures—A case study of Northeast Normal University Library[J]. Library Work and Study, 2019(8): 66-70.
[6] Zhu Yuqiang. Development and application of document delivery robot in WeChat ecosystem[J]. Library Tribune, 2019(9): 135-139.
[7] Fan Chuang, Yin Ming, Wang Fei, et al. Application of order dispatch system in Jiangsu Engineering Technology Literature Information Center Platform[J]. Jiangsu Science and Technology Information, 2021(9): 56-58.

Corresponding Author: Shen Jinian, E-mail: shen@cpu.edu.cn

Author Contributions Statement:
Shen Jinian: Conceived the research idea, designed the development plan, and wrote the paper;
Li Jing: Conducted testing and provided improvement suggestions;
Liu Xinxia: Processed data and created charts.

Note: Figure translations are in progress. See original paper for figures.