EasySpider: A No-Code Visual System for Crawling the Web

Naibo Wang
National University of Singapore
Singapore
naibowang@u.nus.edu

Wenjie Feng
National University of Singapore
Singapore
wenchiehfeng.us@gmail.com

Jianwei Yin
Zhejiang University, Hangzhou
China
zjuyjw@cs.zju.edu.cn

See-Kiong Ng
National University of Singapore
Singapore
seekiong@nus.edu.sg

ABSTRACT

The web is a treasure trove for data that is increasingly used by computer scientists for building large machine learning models as well as non-computer scientists for social studies or marketing analyses. As such, web-crawling is an essential tool for both computational and non-computational scientists to conduct research. However, most of the existing web crawler frameworks and software products either require professional coding skills without an easy-to-use graphic user interface or are expensive and limited in features. They are thus not friendly to newbies and inconvenient for complicated web-crawling tasks.

In this paper, we present an easy-to-use visual web crawler system, `EasySpider`, for designing and executing web crawling tasks without coding. The workflow of a new web crawling task can be visually programmed by following `EasySpider`'s visual wizard on the target webpages using an intuitive point-and-click interface. The generated crawler task can then be easily invoked locally or as a web service. Our `EasySpider` is cross-platform and flexible to adapt to different web-resources. It also supports advanced configuration for complicated tasks and extension. The whole system is open-sourced and transparent for free-access at GitHub [1], which avoids possible privacy leakage.

CCS CONCEPTS

• **Information systems** → **World Wide Web**; • **Human-centered computing** → **Visualization systems and tools**.

KEYWORDS

Web Crawler, Visualization, Data Collection, GUI, Web Service

ACM Reference Format:
Naibo Wang, Wenjie Feng, Jianwei Yin, and See-Kiong Ng. 2023. EasySpider: A No-Code Visual System for Crawling the Web. In *Companion Proceedings of the ACM Web Conference 2023 (WWW '23 Companion), April 30–May 04, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3543873.3587345

[1]https://github.com/NaiboWang/EasySpider

1 INTRODUCTION

The web has become the go-to source for data in a wide range of real-world applications. For example, internet companies routinely collect data about their products or reputation online for product improvement or promotion, financial professionals collect timely data from the web for market analysis and forecasting, and social scientists increasingly look to the web for large-scale social behavioral data that can be difficult or costly to observe otherwise. The recent exceptionally successful big machine learning models such as ChatGPT [4] has further illustrated the usefulness and power of massive amounts of internet-sourced data. This raises the natural question of: *how to obtain these public online data from different domains in a simple and convenient way?* and *how can we design an easy-to-operate, safe, flexible, automatic, and lightweight tool for general purpose web-crawling?*

Web crawlers [3] has thus become an essential and widely-used tool for researchers and practioners from many disciplines in collecting data from the World Wide Web. While there are many open frameworks for web crawling such as Scrapy, Beautiful Soup, and PySpider [1], most if not all of them require users to have a deep knowledge of the mechanism of the web crawler and programming skills, such as Java or Python, which is a big challenge for the non-programmers from other disciplines. They are also often not lightweight—to build even a simple task for collecting data usually require writing many lines of code.

There are commercial software products that provide user-friendly graphical user interface (GUI) [5] for performing the crawler tasks without heavy programming such as WebHarvy, Visual Scraper, Web Scraper, Octoparse, and so on. However, they may not be affordable to all (they claim as "free" softwares but actually not), and most of such non-coding crawler products are not open-source. While there are some open source visual crawlers, they are often not fully functional (e.g. Portia does not support filling forms), not well-maintained, lack complex workflow support which makes it hard or impossible for advanced users to customize and extend the functionality based on their needs, such as to support dynamic expansion of tasks and multi-layer loop nesting.

It can be challenging to design and develop an easy-to-use and well-designed crawler system. Other than technical difficulties such as robust system design, good compatibility, and convenient interaction, etc., the issue of user-data privacy protection [2] should also be considered. As there is often the need to log in to access the data (e.g., for the online social media or forum platform), passing the user's credentials such as username & password or local cookies, to black-box crawler products can result in the risk of leakage of

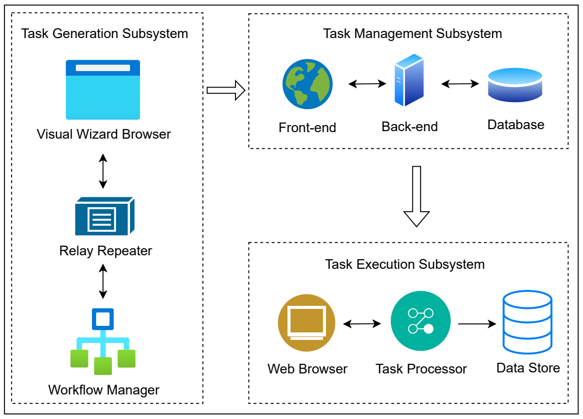


Figure 1: High level system design of EasySpider.

personal privacy information, especially if the crawler task runs on the cloud servers. For safer web scraping, a transparent white-box system that keeps private data locally would be thus preferable.

In this paper, we design and develop a new web crawler software, EasySpider, which is:

- **Interactive and user-friendly:** it provides a user-friendly point-to-click GUI for quick and easy task design for non-programmers [2].
- **Flexible:** it is able to adapt to various web resources and platforms, dynamic content support [3].
- **Configurable:** it supports easy design of complicated crawling tasks with unlimited loops, if-conditions, and fields.
- **Cross-platform:** it supports Windows, Linux, and MacOS.
- **Open-Source & Trustworthy:** the source code is publicly available, making it a transparent and white-box software, with no privacy leakage risk.
- **Free:** all features listed in the manual are free of charge.

Our system is suitable for both general users and experts, and it can also serve as microservice to incorporate with other systems (e.g., DBMS, HDFS, and others).

2 SYSTEM ARCHITECTURE

Figure 1 illustrates the overview of our EasySpider system. It consists of three subsystems for generation, management, and execution of a web crawling task, respectively. Each of them is introduced in the following sections.

2.1 Task Generation Subsystem

The task generation subsystem is responsible for new crawling task creation. Users design their tasks via a visual web client consisting of three main components: *Visual Wizard Browser*, *Relay Repeater*, and *Workflow Manager*. The detailed process is as follows. The user starts by entering the URL of a target web page, and define the whole task workflow by point-and-click in the *Visual Wizard Browser*, i.e., she can select any element(s) on the web page(s) she wants by mouse click, and follows the wizard in the browser to define some operations, such as collecting data, inputting in the textbox, or loop-clicking links to collect data from the opened new pages, etc. Each operation defined by the user will then be listed in the *Workflow Manager*, which is a visual user interface that displays the whole workflow of this task; it also shows the key properties of the operations such as the waiting time before starting the operation, the XPath of the selected element, the default input for the textbox, the number of screen scrolls, etc. All of these properties can be modified freely on the GUI to make the whole process smoother and user-customizable.

Our system can deal easily with complex structures in a webpage. There are often situations where we need to perform iterations, such as clicking all links on product names to check their detail pages on shopping websites, or collecting the titles in the list of items returned by a Google search, etc. In many cases, we will need the nested loops for more complex operations, e.g., keeping clicking the next page button and capturing the news headlines of each page. Therefore, EasySpider supports the *loop* operation for users to freely define the advanced task workflow; and the *loops can be nested infinitely* to perform more complicated crawling tasks. At the same time, users can also define the *conditional judgment* operation, i.e., the "IF" condition, to expand the scenario, for example, setting the "Click" operation to be executed only when the page contains a "Next" Button.

EasySpider also supports the "clipboard" function for all the above operations, including loop, conditional judgment, input the text, click an element, collect data, etc. That is, users can **CREATE, COPY, PASTE, CUT, UPDATE, and DELETE** any operation in the workflow freely through the *workflow manager*.

The *Relay Repeater* is the bridge between the *Visual Wizard Browser* and *Workflow Manager* to transmit the operations' information, including the operation type, the URL of web pages, XPaths of target elements, etc.

2.2 Task Management Subsystem

Once the task is created by the previous steps, the entire processing workflow and metadata will be saved in the Task Management Subsystem, and can be invoked in the form of a Web service. For instance, every input operation in the workflow will be mapped as an input parameter of the service API, such as the URL(s) of the 'Open Page' operation or the text in a searching box of the 'Input Text' operation. All task information is stored in the *Database (MongoDB)*, users can read, update, remove, and invoke their tasks via the provided API or GUI of the *Front-end*.

Before executing the task, we need to preset the input parameters for it, which is called "invoke" the task. For instance, to get all news about a celebrity in a certain time period, we need to specify the value of the related input parameters, i.e., the celebrity's name, the start and end time, etc. Users can set those values through the *Front-end* GUI or call the API provided by the *Back-end*. After getting the parameters, the *Back-end* will return a task-execution number *EID* to the users (or the client) used for the next step.

2.3 Task Execution Subsystem

Finally, the Task Execution Subsystem runs a given task by reading the execution number *EID* produced in the previous subsystem, which contains key information about the task.

The *Task Processor* is the core component here, and it reads the workflow information and input parameters by the *EID* from the

[2]Only a little basic knowledge, such as XPath, is needed to modify the task flow.
[3]Can collect web content generated by JavaScript (AJAX).

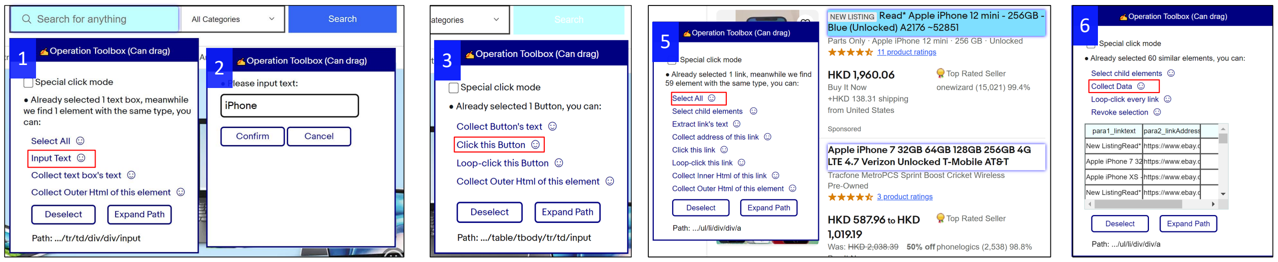

Search for anything
All Categories
Search
1
Operation Toolbox (Can drag)
Special click mode
• Already selected 1 text box, meanwhile we find 1 element with the same type, you can:
Select All
Input Text
Collect text box's text
Collect Outer Html of this element
Deselect
Expand Path
Path: .../tr/td/div/div/input
2
Operation Toolbox (Can drag)
Please input text:
iPhone
Confirm
Cancel
3
Operation Toolbox (Can drag)
Special click mode
• Already selected 1 Button, you can:
Collect Button's text
Click this Button
Loop-click this Button
Collect Outer Html of this element
Deselect
Expand Path
Path: .../table/tbody/tr/td/input
5
Operation Toolbox (Can drag)
Special click mode
• Already selected 1 link, meanwhile we find 59 element with the same type, you can:
Select All
Select child elements
Extract link's text
Collect address of this link
Click this link
Loop-click this link
Collect Inner Html of this link
Collect Outer Html of this element
Deselect
Expand Path
Path: .../ul/li/div/div/a
NEW LISTING Read* Apple iPhone 12 mini - 256GB - Blue (Unlocked) A2176 ~52851
HKD 1,960.06
Apple iPhone 7 32GB 64GB 128GB 256GB 4G LTE 4.7 Verizon Unlocked T-Mobile AT&T
HKD 587.96 to HKD 1,019.19
6
Operation Toolbox (Can drag)
Special click mode
• Already selected 60 similar elements, you can:
Select child elements
Collect Data
Loop-click every link
Revoke selection
para1_linktext
para2_linkAddress
Deselect
Expand Path
Path: .../ul/li/div/div/a

Figure 2: UI for the Task Generation Process of `EasySpider` to collect the names and detail page links of specified products from eBay. Step 4 is skipped since it is similar to Step 3.

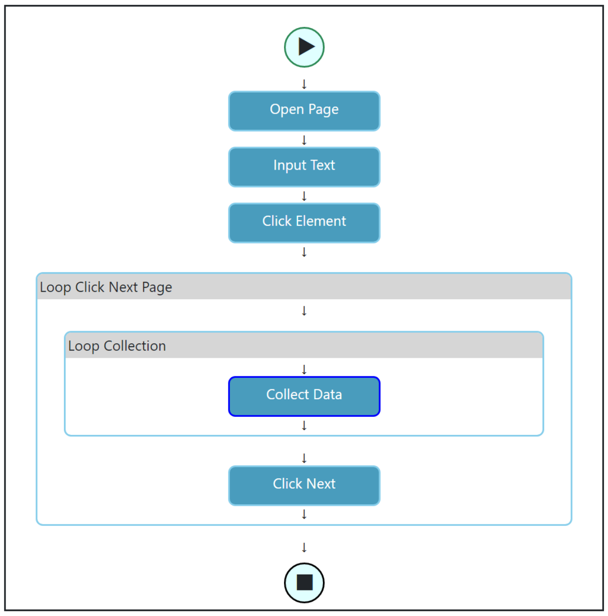


Figure 3: UI of `EasySpider`'s workflow for the task of collecting product information on eBay.

Task Management Subsystem, then controls the *Web Browser* to automatically execute the operations defined in the workflow, such as collect data or click button from the specific web page(s), and exports the results into the *Data Store* in various formats, such as .csv, .json, and databases like *MongoDB*.

The workflow will be executed sequentially from top to bottom. If loops and conditional judgments are encountered, they will be executed according to the corresponding logic, i.e., no matter how many loops and nested loops inside the task workflow, the *Task Processor* will ensure the correct execution of the task through techniques/algorithms, e.g., the program trees and recursion. Here, the end-users can treat the whole process as a black box if they do not care about specific implementation details.

3 DEMONSTRATION

We give a simple running example of `EasySpider` to demonstrate how to use our system. Let's say we are interested to collect the names and corresponding link of the detail pages of the top n (specified later) pages of products searched for a given keyword on an e-commerce website such as *eBay* (https://www.ebay.com).

Given a keyword and n, say "iPhone" and 20, we would like the product information collection task (refers as "task" later) workflow to be executed as follows:

(1) Enter the product keyword in the *searching box* and click the "Search" button.
(2) Identify and collect the name and link of the products in the main product list shown on the web page.
(3) Click the "Next" button to go to the next page.
(4) Repeat step 2 and 3 for n times.

Each time we run this information collection task, we would like to be able to modify the keyword (step 1) and n (step 4) at will, and all of the above steps should be automatically executed accordingly.

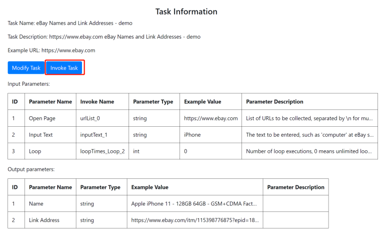
3.1 Task Design & Creation

Figure 2 shows the whole process to create the above crawling task, and Figure 3 shows the overview workflow.

As we can see, the total task creation process only contains the following 6 steps after entering the URL of the target website, eBay.

(1) Select the searching textbox with mouse right-click to identify it on the page, which will make the textbox highlighted in bright blue, then appear a toolbox, which contains a lot of options based on the selected element type, such as selecting all similar elements, inputting text, and collecting text from the textbox, etc. Here, we select the "Input Text" option by mouse click[4].
(2) Input a keyword, e.g., "iPhone" in Fig. 2, and click the *Confirm* button at the toolbox. The keyword will be automatically filled into the searching box, and an operation "Input Text" will then be added to the workflow, as shown in Fig. 3.
(3) Select the "Search" button on the webpage and click the "Click this Button" option in the toolbox. Then the browser will automatically click the search button and receive the returned product list page, and the "Click Element" operation will also be added to the workflow at this moment.
(4) Select the "Next" button at the bottom of the webpage and click the "Loop-click this link" option. This step is similar to step 3 and ignored in Fig. 2.
(5) Select the 'title/name' of the 1st item, we can see in addition to the bright-blue highlighted first product title, all other products' titles are also indicated by the blue border at the same time in Fig. 2, which are automatically detected by our system. Then click the "Select All" option in the toolbox to

[4]Left-click unless otherwise specified.



Task Information

Task Name: eBay Names and Link Addresses - demo

Task Description: https://www.ebay.com eBay Names and Link Addresses - demo

Example URL: https://www.ebay.com

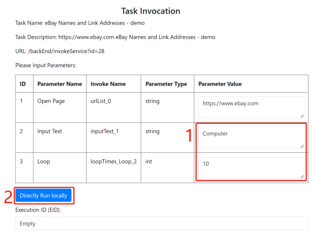
Input Parameters:

ID	Parameter Name	Invoke Name	Parameter Type	Example Value	Parameter Description
1	Open Page	urlList_0	string	https://www.ebay.com	List of URLs to be collected, separated by \n for mu...
2	Input Text	inputText_1	string	iPhone	The text to be entered, such as 'computer' at eBay s...
3	Loop	loopTimes_Loop_2	int	0	Number of loop executions, 0 means unlimited loo...

Output parameters:

ID	Parameter Name	Parameter Type	Example Value	Parameter Description
1	Name	string	Apple iPhone 11 - 128GB 64GB - GSM+CDMA Fact...	
2	Link Address	string	https://www.ebay.com/itm/115398776875?epid=18...	

(a) UI for the task information.



Task Invocation

Task Name: eBay Names and Link Addresses - demo

Task Description: https://www.ebay.com eBay Names and Link Addresses - demo

URL: /backEnd/invokeService?id=28

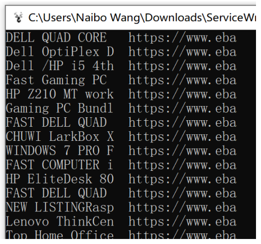
Please Input Parameters:

ID	Parameter Name	Invoke Name	Parameter Type	Parameter Value
1	Open Page	urlList_0	string	https://www.ebay.com
2	Input Text	inputText_1	string	Computer
3	Loop	loopTimes_Loop_2	int	10

Execution ID (EID):

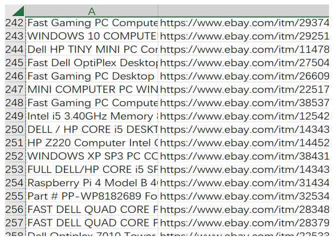
Empty

(b) UI for the task invocation.



```
C:\Users\Naibo Wang\Downloads\ServiceWr
DELL QUAD CORE   https://www.eba
Dell OptiPlex D  https://www.eba
Dell /HP i5 4th  https://www.eba
Fast Gaming PC   https://www.eba
HP Z210 MT work  https://www.eba
Gaming PC Bundl  https://www.eba
FAST DELL QUAD   https://www.eba
CHUWI LarkBox X  https://www.eba
WINDOWS 7 PRO F  https://www.eba
FAST COMPUTER i  https://www.eba
HP EliteDesk 80  https://www.eba
FAST DELL QUAD   https://www.eba
NEW LISTINGRasp  https://www.eba
Lenovo ThinkCen  https://www.eba
Top Home Office  https://www.eba
```

(c) Sample log of *Task Processor.*



	A	
242	Fast Gaming PC Compute	https://www.ebay.com/itm/29374
243	WINDOWS 10 COMPUTE	https://www.ebay.com/itm/29251
244	Dell HP TINY MINI PC Cor	https://www.ebay.com/itm/11478
245	Fast Dell OptiPlex Deskto	https://www.ebay.com/itm/27504
246	Fast Gaming PC Desktop	https://www.ebay.com/itm/26609
247	MINI COMPUTER PC WIN	https://www.ebay.com/itm/22517
248	Fast Gaming PC Compute	https://www.ebay.com/itm/38537
249	Intel i5 3.40GHz Memory	https://www.ebay.com/itm/12542
250	DELL / HP CORE i5 DESKT	https://www.ebay.com/itm/14343
251	HP Z220 Computer Intel (	https://www.ebay.com/itm/14452
252	WINDOWS XP SP3 PC CC	https://www.ebay.com/itm/38431
253	FULL DELL/HP CORE i5 SF	https://www.ebay.com/itm/14343
254	Raspberry Pi 4 Model B 4	https://www.ebay.com/itm/31434
255	Part # PP-WP8182689 Fo	https://www.ebay.com/itm/32534
256	FAST DELL QUAD CORE F	https://www.ebay.com/itm/28348
257	FAST DELL QUAD CORE F	https://www.ebay.com/itm/28379

(d) Sample of the collected data.

Figure 4: An example of the Task Execution Process of EasySpider.

let the system select all recommended items, which makes all products' title/name be marked with bright blue.

(6) After step 5, the toolbox will show a field filled with example data, i.e., the product names and links here. Click the "Collect Data" option in the toolbox to create a loop in the workflow to collect all product names and their links.

As shown, the task creation process can be completed quickly in minutes, and intuitively with just a few mouse clicks. For more complex task such as looping to open detail pages, and collecting the name, price, # of stars, etc. of products, please refer to our tutorials and video-clips.

3.2 Task view and execution

We can inspect the task information via the Task Management Subsystem. Figure 4(a) shows the meta information of the task, including task name, description, input parameters (e.g., search keyword), and output parameters (e.g., product name). Users can click the "Invoke Task" button to execute the task.

As Fig. 4(b) shows, we can execute the task by using the GUI or via the provided API. Only 2 simple steps are required for execution,

(1) Specify the search keyword and the number of loops (pages) *n*. Here, we entered "Computer" and 10.
(2) Click the "Directly Run Locally" button to run the task.

After that, EasySpider will run the task automatically.

As mentioned, the workflow in Figure 3 will be executed from top to down, where the loop will follow the "do-while" rule. In other words, our system will perform the web crawling task by automatically entering the keyword "Computer" into the searching box, clicking the search button, and repeating the following steps 10 times in a "do-while" loop,

(1) Collect all product names and their links at current page by iteratively locating the elements in the product list, which is shown as the "Loop Collection" operation inside the "Loop Click Next Page" operation in Figure 3.
(2) Click the "Next" button to jump to the next page, which is shown as the "Loop Click Next Page" operation in Figure 3.

Figure 4(c)-4(d) shows the example output logs and collected data after executing this task, allowing the users to monitor the progress of the task and the results.

3.3 Performance Evaluation

We tested this eBay collection task on a PC with Intel Core i7-8700K CPU, 64GB memory and Windows 10 x64. When executing the task, EasySpider utilized an average of 7% of CPU and 380 MB of memory. It takes around 6 seconds to collect data on a single page, and the overall time for collecting data from all 10 pages is 90 seconds, with all operations included like clicking "Next" button.

4 DISCUSSION

Limitations: While EasySpider can collect a majority of web pages on the Internet, certain resource types such as encrypted video streams cannot be gathered directly. We also need to configure tasks with caution to prevent getting blocked by web servers. Other cases such as how to collect on heterogeneous websites or deal with very large amount of data, please refer to our documentation.

Ethics: Inevitably, there will be some risk of malicious use or data infringement issue, e.g., automatic order swiping and ticket grabbing, but this is contrary to our expectations. As a tool developer, we only hope that it can be used for legitimate purposes. We advocate the reasonable and legal utilization of our system, respecting and protecting the data security and privacy.

5 CONCLUSION

We present and develop EasySpider, a point-and-click visual web crawler system to enable quick and easy customizable design of crawler tasks for both non-programmers and experts. The generated crawler tasks can be invoked locally or in the form of web services for easy integration with other systems. Our system is flexible, configurable, platform-agnostic, open-sourced, and free.

ACKNOWLEDGMENTS

This project is supported by the National Research Foundation, Singapore under its Industry Alignment Fund – Pre-positioning (IAF-PP) Funding Initiative. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore.

REFERENCES

[1] Rabiyatou Diouf, Edouard Ngor Sarr, Ousmane Sall, Babiga Birregah, Mamadou Bousso, and Sény Ndiaye Mbaye. 2019. Web scraping: state-of-the-art and areas of application. In *2019 IEEE Big Data*. IEEE, 6040–6042.
[2] Carlos Jensen, Chandan Sarkar, Christian Jensen, and Colin Potts. 2007. Tracking website data-collection and privacy practices with the iWatch web crawler. In *Proceedings of the 3rd symposium on Usable privacy and security*. 29–40.
[3] Moaiad Ahmad Khder. 2021. Web Scraping or Web Crawling: State of Art, Techniques, Approaches and Application. *IJASCA* 13, 3 (2021).
[4] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training. (2018).
[5] Andrea Stocco, Rahulkrishna Yandrapally, and Ali Mesbah. 2018. Visual web test repair. In *Proceedings of the 2018 26th ACM ESEC/FSE*. 503–514.