Abstract

Artificial Intelligence is the trend in software development. Unfortunately, Artificial Intelligence algorithms and technologies are still not utilized enough in software testing. Designing Test automation has become the main job for quality engineers and software testers. Mainly, Test Automation is beneficial in reducing manual testing efforts. Utilizing AI in test automation can form a huge benefit in code optimization and test oracle problem. The primary objective of the research was to approve the usability of the Fuzzy Inference System in providing a test oracle for web application functional testing. The secondary objective was to utilize Artificial Intelligence techniques like self-healing for the test Automation using web scraping. Also, to compare the web scraping approach and the Image processing approach in locating the web elements on the websites dynamically. I have addressed the problem by developing Test Automation that verifies the search functionality for a given website. The hypothesis is mainly to check if the Fuzzy Inference System can predict if the search functionality for a given website is working or not. I tested the hypothesis on ten different websites. Then, after I analysed the results, I have found that implementing the Fuzzy Inference System in test automation can form a reasonable solution for the test oracle problem. Furthermore, using the Fuzzy Inference System is as efficient as the manually prepared test oracle that covers all the possible cases for the inputs using if-else statements. Finally, I have demonstrated how web scraping can be utilized to perform self-healing for the test Automation.

Keywords

* **Test Automation**
* **Artificial Intelligence**
* **Fuzzy Inference System**
* **Test Oracle**

[Download](https://link.springer.com/content/pdf/10.1007/978-3-031-10464-0_24.pdf?pdf=inline%20link) conference paper PDF

1 Introduction

AI is a phenomenon that nearly everyone has heard of recently. Nowadays, AI is taking over the software testing field as well. Utilizing AI in software testing to solve the testing problems is being the new trend in the software Quality assurance industry [[14](https://link.springer.com/chapter/10.1007/978-3-031-10464-0_24#ref-CR14)]. Software Testing is a very critical role in the software development cycle. Software quality assurance (QA) puts a lot of effort into accelerating the testing process like implementing Test Automation [[4](https://link.springer.com/chapter/10.1007/978-3-031-10464-0_24#ref-CR4)]. The field of utilizing Artificial Intelligence in Test Automation has grown during the last years [[14](https://link.springer.com/chapter/10.1007/978-3-031-10464-0_24" \l "ref-CR14" \o "King, T.M., Arbon, J., Santiago, D., Adamo, D., Chin, W., Shanmugam, R.: AI for testing today and tomorrow: industry perspectives. In: 2019 IEEE International Conference On Artificial Intelligence Testing (AITest), 81–88. IEEE (Apr 2019))], the main target is to minimize the QA time and efforts of writing or modifying the Test Automation. Also, to overcome problems facing the Test Automation like the Test Oracle Problem [[9](https://link.springer.com/chapter/10.1007/978-3-031-10464-0_24#ref-CR9)].

In this project, I aim to design a Test Automation for a Web Application that aims to apply functional testing on webpage elements by tracking the HTTP requests generated while interacting with web elements [[5](https://link.springer.com/chapter/10.1007/978-3-031-10464-0_24#ref-CR5)]. For verification, I will use Fuzzy Logic [[10](https://link.springer.com/chapter/10.1007/978-3-031-10464-0_24" \l "ref-CR10" \o "Zadeh, L.A.: Fuzzy logic. Computer 21(4), 83–93 (1988))] to predict the result of the Test Automation execution.

The project’s biggest goal is to speed up the QA process and to provide a solution for the Test Oracle Problem using AI. The project and the conducted research aim to provide answers to the following questions. Can the designed Test Automation be intelligent to validate the test results to overcome the test oracle problem by using Fuzzy logic? Moreover, how to utilize the Fuzzy Inference system in test automation for web functional testing, and what is the efficiency for such implementation compared to the old way of preparing a static condition for all possible cases using if else statements?

What is the best AI mechanism for executing a test scenario on a webpage? That is, the Test Automation can be intelligent and dynamic. e.g.: one test Automation script for testing Search functionality that can work on different websites. Also, what is the ability for applying self-healing for the test Automation using web scraping.

Finally, my project aims to evaluate the approach of testing the user interface (UI) for a webpage by analysing the HTTP requests generated while interacting with the webpage elements. That is, the Test Automation aims to test the functionality of the server-side for web applications through the client-side by applying Gray box testing. This approach should help in locating the failure layer, e.g., the bug location is on the client side or on the server side.

To answer the above question, the primary objective of the project is to develop a test Automation that utilizes AI. The test Automation aims to perform a search functional test on web applications and evaluate the results using the Fuzzy Inference system.

2 Related Work

Many pieces of literature used Fuzzy Logic to evaluate the website quality. In [[13](https://link.springer.com/chapter/10.1007/978-3-031-10464-0_24#ref-CR13)], authors implemented the Fuzzy Inference System to evaluate website design quality based on Fuzzy-DEMATEL. In other literature [[11](https://link.springer.com/chapter/10.1007/978-3-031-10464-0_24#ref-CR11)], authors used Neuro-Fuzzy Logic based on nine web measures like performance, number of elements on each webpage and other metrics to evaluate the web page quality. In another literature [[12](https://link.springer.com/chapter/10.1007/978-3-031-10464-0_24#ref-CR12)], this work proposed a fuzzy classifier to evaluate the Web applications vulnerability toward well-known attacks like xss and SQL injection. In other literature, [[8](https://link.springer.com/chapter/10.1007/978-3-031-10464-0_24#ref-CR8)] authors evaluated the website quality using Fuzzy logic. The authors added the correctness of the web page element “link” as one of the inputs for the Fuzzy inference system. Authors calculated the rate of correctness for webpage elements “links” to evaluate the quality of a website. In this method, the authors used the Xenu tool to check the link correctness and then Fuzzy Logic to evaluate the quality of that website based on other different factors like (‘rate of links correctness’ and website performance). Authors found that Fuzzy Logic is an excellent way to evaluate the quality of websites.

Generally, all the previous Literature found Fuzzy Logic to be feasible in predicting the quality of a website.

The authors [[6](https://link.springer.com/chapter/10.1007/978-3-031-10464-0_24" \l "ref-CR6" \o )] have presented a novel approach of providing a test oracle for software using deep learning and fuzzy inference systems. The proposed work applied only for the software that has numeric output. Authors used the Fuzzy Inference system as a first layer to map inputs into a fuzzy space. Mainly, this layer is used to train the Deep learning network layer.

The second layer is the deep learning network, which is designed to process data provided by the Fuzzy inference system and try to find a pattern. Based on that, the output of the Deep learning network is the test oracle. Authors have tested their approach on different software and approved its validity.

In my project, I aim to implement a Fuzzy Inference System for a different purpose, which is evaluating the functionality of the search feature in a given website.

I aim to process the Search HTTP request and response content using Fuzzy Inference System to determine if the content generated is valid or not.

The novelty in my approach is that I get the advantages of the fuzzy inference system output surface to form a test oracle for system inputs based on certain rules.

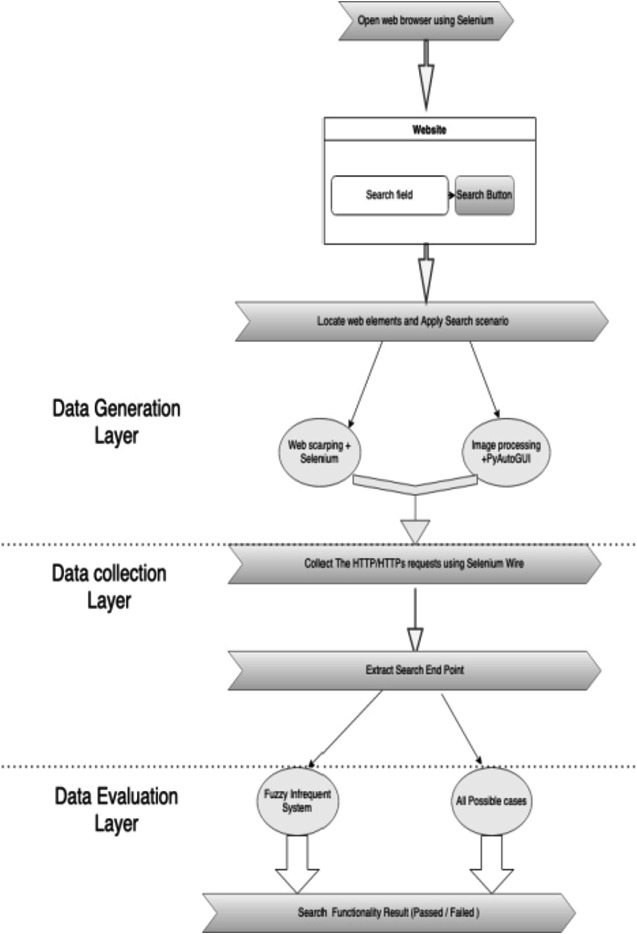
In conclusion, according to the literature, AI has been implemented in a different area of software testing for web applications. Utilizing AI to solve the test oracle problem is one main area. My project target is to develop test Automation to test the server-side of web applications through client-side and provide test oracle using Fuzzy Logic for the server-side level. This approach aims to save the software testers time and effort. Furthermore, it will provide a solution for the test oracle problem by utilizing AI in test automation using Fuzzy inference System.

For the secondary objective, I aim to utilize AI techniques in test Automation. That is, Test Automation is intelligent where it can test on different websites without being supervised.

3 Methodology

This work attempts to check if the search functionality of a website is working or not. That is, when the end-user fills the search and then clicks the search button on the web page it will function properly and retrieve the search results. In order to evaluate if search functionality is working or not, I aim to use Fuzzy Logic. Mainly, using the Fuzzy Inference system to provide a test oracle for the test Automation. Furthermore, to approve that Fuzzy Logic test oracle can be a feasible replacement for other test oracle methods. The reason behind choosing fuzzy logic is to get advantage of the Fuzzy set theory ability of classifying the inputs in order to predict the result of the test Automation.

**Fig. 1.**

[](https://link.springer.com/chapter/10.1007/978-3-031-10464-0_24/figures/1)

Test automation system architecture

[**Full size image**](https://link.springer.com/chapter/10.1007/978-3-031-10464-0_24/figures/1)

The proposed test automation system architecture as showed in Fig. [1](https://link.springer.com/chapter/10.1007/978-3-031-10464-0_24#Fig1) is consisting from two main layers: First layer is the Data Generation and Collection. In the First layer mainly, I execute the test scenario script on the web browser. The test scenario script is to locate the search field and the search button. Then, fill the search field with a search Keyword and click on the search button. There are two methods for Data Generation, first one using Image processing and the other one is to use web scraper agent. The data collected is the HTTP request and response for the search endpoint, which is generated by clicking on the search button. As the test case validation is done on the server-side level by processing this HTTP request and response of the search endpoint generated by interacting with the webpage element on the client-side (web browser). Therefore, in the first layer I aim to develop a function that extracts the search Endpoint request from a group of HTTP requests.

The second layer is the Data Evaluation Layer, where I have two approaches. First one is the Fuzzy Inference system approach that is targeting to present a test oracle for the test automation. Fuzzy Inference system layer aims to predict if the search functionality of the website is passed or failed based on the Fuzzy inference rules. The other evaluation approach is to prepare all different possible cases and if the constraints meet for one case, then the test case is passed otherwise it is failed. I call this approach “All possible cases using if-else”. The overall structure of the Test Automation should be flexible in a way that works on different websites. In my Test Automation I am targeting different 10 websites.

Test Automation should verify the search functionality per each one of them.

I aim to design the test automation in a way that is not supervised or personalized. In other words, Test Automation should be flexible and able to test each one of the below websites listed in Table [1](https://link.springer.com/chapter/10.1007/978-3-031-10464-0_24#Tab1).

| **URL** |
| --- |
| [https://www.google.com](https://www.google.com/) |
| [https://www.amazon.com](https://www.amazon.com/) |
| <https://old.bau.edu.jo/> |
| <https://www.reddit.com/> |
| <https://www.bbc.co.uk/search> |
| <https://uk.yahoo.com/?p=us> |
| <https://www.nottingham.ac.uk/search.aspx> |
| <https://www.ox.ac.uk/> |
| <https://www.leeds.ac.uk/> |
| <https://stackoverflow.com/> |

**Table 1. List websites that will be tested using the test automation**

[**Full size table**](https://link.springer.com/chapter/10.1007/978-3-031-10464-0_24/tables/1)

3.1 Data Generation and Collection

The first layer aims to execute the Test Automation for Search functionality on different websites. The test case scenario for the Search functionality is mainly filling the search field with a search keyword and then clicking on the search button. The test case expected result is to retrieve the search result for the search keyword.

First of all, the test script explores the given URL to extract the needed data for the test scenario. The test Automation should locate the search field and search button web elements. I follow two different AI mechanisms for collecting the location of the search field and the search button for different websites. (1) Using image processing to detect the field and the search button, this is explained in more detail below. (2) Using web scraper to dynamically identify the Xpath for the search field and search button.

The second step is the Data collection, which is to collect all the HTTP/HTTPs requests and responses that were generated by performing the search functionality on the Automation web browser.

In order to collect the HTTP requests, System uses the Selenium-Wire library which extends Selenium. System Later Processes and filters all The HTTP requests generated by the browser to extract the Search Endpoint.

3.2 Data Evaluation Using Fuzzy Inference System

In this layer of the system, the test Automation evaluates the collected data using the Fuzzy Inference System. The Fuzzy Inference System task is to predict if the Search Endpoint web-service generated by the search automated test on a given website is valid. I add different rules for the fuzzy inference system to classify the output based on the inputs.

The following are the two inputs in the Fuzzy Inference system: The first input is the status code for the Search web-service HTTP request where each status code is divided by 100. The reason for that is to standardise the input variables range so both inputs are at the same range, instead of having the first input range in hundreds and the other input from (1–5). The second Input is the rate of the existence of the search keyword in the Search Endpoint response. System has a function for calculating the rate of the existence of the search keyword. The Function takes the body of the response for search end-point as an Input. Then by using Natural Language Toolkit - NLTK library to record the number of times each word has occurred in a document using the method: “Frequency distributions”. Formally, a frequency distribution can be defined as a function mapping from each sample to the number of times that sample occurred as an outcome [[1](https://link.springer.com/chapter/10.1007/978-3-031-10464-0_24#ref-CR1)]. In this test Automation case, the targeted word is the search Keyword and the document is the HTML body response. “Frequency distributions” are generally constructed by running a number of experiments, and incrementing the count for a sample every time it is an outcome of an experiment (Bird, 2006). The function will produce a frequency distribution that encodes how often each word occurs in a text. By looking only for the search keyword in the text, in the system:

1. 1.

If the word is repeated more than 10 times in the HTTP body response of the search endpoint that means the search functionality rate is high. I assign the value 5 as an input for the Fuzzy Inference System which indicates the high rate for the search keyword in the search endpoint.

1. 2.

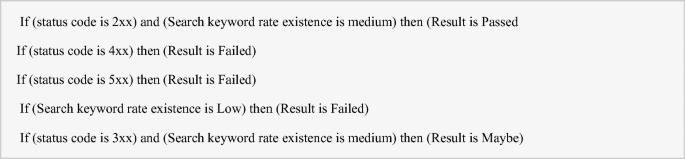
If the word repeated for around 5 times in the HTTP body response of the search endpoint, that means the search functionality rate is medium, I assign the value 3 as an input for the Fuzzy Inference System which indicates the medium rate for the search keyword in the search endpoint.

1. 3.

Finally, if the search keyword word was repeated less than 2 times, then this indicates a low frequency rate for the search keyword times in the HTTP body response of the search endpoint. Thus, I assign the value 1 as an input for the Fuzzy Inference System which indicates the low rate for the search keyword in the search endpoint.

The Fuzzy Inferences System has two inputs, for example if the Inputs are (2,5): That means that the search endpoint request status code is 200 and the search keyword rate in the body of the request is high. Such cases indicate that the search functionality is passed.

For the fuzzy Inference system rules, Rules are designed to be simple and clear.



Test Automation processes the inputs using the Fuzzy inference system. The output of the Fuzzy system will be one of the following (Failed, Passed, Unknown):

1. 1.

“Failed”: indicates that an incorrect behaviour was detected in the Test Automation, that is, the HTTP request and response generated when interacting with the webpage element was not as expected. Thus, the functionality of the webpage element is not working.

Range of the output “Failed” is from (0 to 40).

1. 2.

“Passed”: indicate that the behaviour detected in Test Automation is correct and the functionality of the webpage element is working based on the server-side level.

Range of the output “Failed” is from (75 to 100).

1. 3.

“Maybe”: System was unable to give a decision for this case.

Range of the output “maybe” is from (40 to 75).

The Fuzzy Inference Output surface viewer 3D Plot represents the test oracle for the search functionality for the combination of two inputs as demonstrated later (see Fig. [7](https://link.springer.com/chapter/10.1007/978-3-031-10464-0_24#Fig7))*.*

3.3 Alternative Data Evaluation Layer

Assertions are well known as one of the test oracles where QA usually design test automation and provide a previous expected result. In this approach, I prepared a group of cases using “If- else” statements and assertion. That is, if a case meets the expected results, then I add assert (True) else I add assert (False). An assertion in general is evaluating a constraint that applies to some rules and computation. When the assertion to the value is false then the test automation will exit the run and highlight that error was found [[3](https://link.springer.com/chapter/10.1007/978-3-031-10464-0_24#ref-CR3)].

In this alternative test oracle method, I implement a function that uses the normal Assertion Library in python. The Inputs of this layer is the status code of the response for the Search End-point and the search keyword existence rate. In this layer, I defined each expected case using if else statement. For example, all possible cases number for the inputs equals to 12 as demonstrated in Table [2](https://link.springer.com/chapter/10.1007/978-3-031-10464-0_24#Tab2). As for the first input there are four possible cases for the status code (2xx,3xx,4xx,5xx) and for the second input, there are 3 possible cases (high, medium, low).

| **Rate of Search keyword in the HTTP response** | | | |
| --- | --- | --- | --- |
| **status code** | **Low** | **Medium** | **High** |
| 2xx | Failed | Passed | Passed |
| 3xx | Failed | Passed | Passed |
| 4xx | Failed | Failed | Failed |
| 5xx | Failed | Failed | Failed |

**Table 2. All possible cases for the**

[**Full size table**](https://link.springer.com/chapter/10.1007/978-3-031-10464-0_24/tables/2)

The Output from this layer is an assertion of True or False. Where False means search test functionality does not work, and True means Test functionality does work. The main target of this layer is to compare the results of this method with the result of the fuzzy inference system to evaluate the usability of applying Fuzzy Logic in the functional automation test. Furthermore, to evaluate the quality of the prediction for the Fuzzy Inference system. The comparison is mainly, to compare the two codes (sizes and performances) and lastly their results.

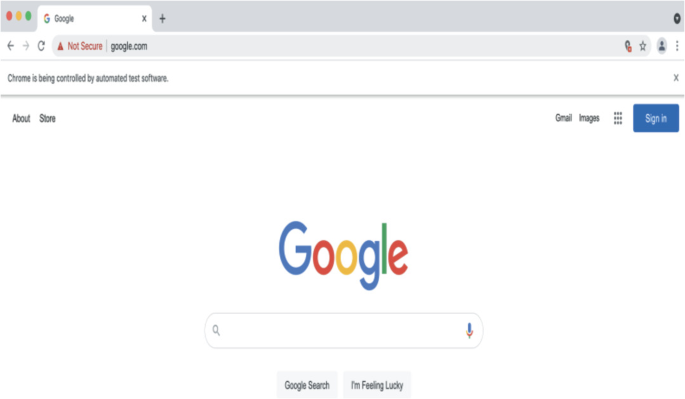
The plan is to run two experiments using the same websites. The first Run uses the Fuzzy Inference System evaluation. The second Run uses the alternative evaluation. I expect that both Data Evaluation layers will have the same results for the same inputs. If so, then Fuzzy Inferences will be approved to be a good replacement as a test oracle for the approach of listing all possible cases. The motive of this comparison is not to suggest that the Fuzzy Logic approach is better than or should replace this method, but rather to see it as an improvement step toward reducing code that leads to reducing the maintenance effort.

3.4 Implementation

Data Generation using Web Scarping

The Test Automation system starts with a command to open the test browser using Selenium web driver. A browser is opened wherein the top bar, the browser should display a message that “this browser is controlled by the automated test software” as showed in Fig. [2](https://link.springer.com/chapter/10.1007/978-3-031-10464-0_24#Fig2).

**Fig. 2.**

[](https://link.springer.com/chapter/10.1007/978-3-031-10464-0_24/figures/2)

The browser is being controlled by the test automation of the selenium web driver

[**Full size image**](https://link.springer.com/chapter/10.1007/978-3-031-10464-0_24/figures/2)

In this layer the search functionality test-case is executed by the Selenium web driver. Selenium Web driver communicates directly with the browser. Selenium Web driver provides great support for testing the properties of page components that keep changes [[7](https://link.springer.com/chapter/10.1007/978-3-031-10464-0_24#ref-CR7)]. Test case is a group of steps done on the webpage usually by the end-user for manual testing. Test cases can be scripted using Selenium Web driver as it supports a wide range of programming languages.

For web scraping approach for data generation, Browser Calls the URL for the targeted website as classified in the Test Automation. Then, test Automation locates the search field and the search button using web scraping. The web scraping mainly looks for the search form in the page HTML, which usually has the attribute “action = search” as shown in Fig. [3](https://link.springer.com/chapter/10.1007/978-3-031-10464-0_24#Fig3).

**Fig. 3.**

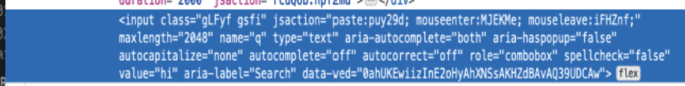
[](https://link.springer.com/chapter/10.1007/978-3-031-10464-0_24/figures/3)

Demonstrates the HTML for the search form with action = “/search”

[**Full size image**](https://link.springer.com/chapter/10.1007/978-3-031-10464-0_24/figures/3)

Inside the search form, for the field web element, “Input” always represents the search field with HTML attribute “type = text” as shown in Fig. [4](https://link.springer.com/chapter/10.1007/978-3-031-10464-0_24#Fig4).

**Fig. 4.**

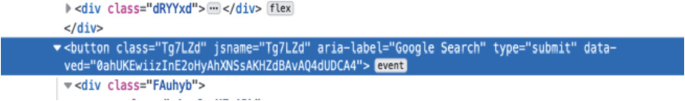
[](https://link.springer.com/chapter/10.1007/978-3-031-10464-0_24/figures/4)

“Demonstrates the field type: = “Text” for the field web element

[**Full size image**](https://link.springer.com/chapter/10.1007/978-3-031-10464-0_24/figures/4)

Also, Inside the search form, for the button web element, “Input” always represents the search button with HTML attribute “type = submit” as shown in Fig. [5](https://link.springer.com/chapter/10.1007/978-3-031-10464-0_24#Fig5).

**Fig. 5.**

[](https://link.springer.com/chapter/10.1007/978-3-031-10464-0_24/figures/5)

Demonstrates the button web element where value = “submit” for the button web element

[**Full size image**](https://link.springer.com/chapter/10.1007/978-3-031-10464-0_24/figures/5)

The system applies web scraping on the given URL. I built a class called parser that will have four functions in total. The first function: gets all the HTML forms in the given URL. The second function: gets the details for a specific form.

The third function finds the field and saves all the attributes for the element field and returns the attribute name that later will be used in determining the field location. The fourth function saves the attributes and the values for the search button and later will be used in locating the search button location.

The System saves the website HTML then Iterates through forms to find the form that has the HTML attribute “action = search” then requests this form details. After that System determines the web elements attribute name for both (search field and search button) using the third and the fourth function.

At this point, the parser class has finished the job by determining the attribute name for web elements. Then Test Automation substitutes the value of the attribute “name” in the Xpath. The following commands show how Selenium web drivers determine the location for the web element based on Xpath using the HTML attribute name.

After allocating the web page elements, the search field, and the search button, the system performs the selenium web driver commands to perform the following test case.

1. 1.

Fill the search field with the search Keyword.

1. 2.

Click on the search button.

Data Generation using Image Processing

For the second approach of Data generation using image processing and PYAUTOGUI, test Automation locates the web page elements using python library PYAUTOGUI. First of all, I provide the test Automation with images for the search field and the search button. Then System uses PYAUTOGUI to locate the (x,y) coordination for the position of where the images were identical to the provided ones on the screen using image processing and segmenting. Later, Test Automation interacts with the web page elements using PYAUTOGUI. PYAUTOGUI takes over the mouse movement on the screen and does the following:

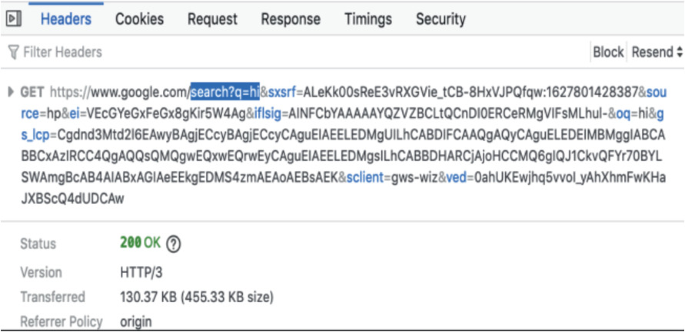
* moves the cursor to the (x,y) position of the identical image of the search field and fills it with the search keyword
* moves the cursor to the (x,y) position of the identical image of the search button and clicks on it.

The First Layer of the system aims to collect data. Data is mainly the search endpoint HTTP request and response that is generated from actions performed on the test Automation web browser. All HTTP requests and responses generated from the moment the selenium web driver has opened the browser till the test automation clicks on the search button are collected using the Selenium wire library.

Data Collection

System iterates through all the HTTP requests to extract the search endpoint. Generally, there is best practice for naming the search endpoint, where it should include the term “/search?”. However, sometimes Websites might use a different name. In this level, I add a function to extract any request that has an endpoint “/search?” at the same time ignores the endpoints “/suggest?” or “/complete?”, as for the functionality test, I aim to test the search itself. not the Auto complete functionality that provides a search keywords suggestion. Moreover, Search endpoint should have the search keyword in its payload of the request for example q =” search keyword”. In the below Fig. [6](https://link.springer.com/chapter/10.1007/978-3-031-10464-0_24#Fig6) is a sample of a search endpoint.

**Fig. 6.**

[](https://link.springer.com/chapter/10.1007/978-3-031-10464-0_24/figures/6)

Shows a search endpoint sample where the name of the service is “/search?” and search keyword “q = hi”

[**Full size image**](https://link.springer.com/chapter/10.1007/978-3-031-10464-0_24/figures/6)

Data Evaluation

After extracting the Search Endpoint in the Data collection phase. The next step is to evaluate if the search Endpoint is valid or not. To do so, Test Automation has to provide a test oracle. In this project, I aim to implement AI to provide this test oracle. Thus, in this Layer, I evaluate data using two approaches. The first one, “Fuzzy Inference System”. The second one, “All possible cases using if-else”.

The first approach of data evaluation is the Fuzzy Inference System. I built a class for the Fuzzy inference system. The Fuzzy inference system accepts two inputs. The Two inputs are extracted from the Search Endpoint that was collected from the first layer. The Fuzzy inference system has a rule that helps to classify the outputs based on the inputs. The Fuzzy inference output is a crisp value, which is a number from zero to 100%. I check the crisp output of the fuzzy inference system, whereas:

* If the crisp output for the Fuzzy inference system is greater than 85% then the system predicts that the search functionality is working. Thus, the test case is marked as passed.
* Otherwise, if the crisp output of the fuzzy inference system is less than 80, based on that the system predicts that search functionality is not working. Thus, the test case is marked as failed.

The Second approach system has an alternative function to evaluate the inputs. The function takes the status code and the rate of the search keyword as inputs. Then through different conditions, System checks which if- statement meets the constraints so it can identify the test result.

4 Result

This Chapter discussed the results of different experiments and studies that were applied on the data generation and the data evaluation levels of the test Automation.

First of all, the main purpose of the study conducted for the Data Generation level of the test Automation is to compare the different approaches of locating the web page elements on different websites. Also, perform a self-healing test-case using a web scraper. That is, the test Automation can perform when HTML selectors change across different websites. Furthermore, to check the feasibility of identifying the bug location by performing functional testing on client side then evaluate the result on the server-side level by analysing the HTTP request generated during the interacting with client-side level (website UI).

Secondly, for the Data evaluation experiments, the main purpose of the experiments conducted is to validate the Fuzzy inference system approach in providing a test oracle for the test Automation. Mainly, validate the quality of prediction for the test result for the search functionality of a given website. Lastly, to compare the test oracle results and performance for the “All possible cases using if-else conditions “approach and “Fuzzy Inference system” approach.

I applied the experiments and the study on different websites categories, like university websites, search engine websites and entertainment websites. The websites that were selected had a search bar feature. The test Automation is designed to verify if the search functionality is working or not based on the search endpoint in the server-side level. The Search endpoint is generated once the user clicks on the search button. The search endpoint is a GET HTTP request that takes the search keyword as a parameter in the payload and retrieves the search result as a response.

The test automation script simulates the user behaviour, where it fills the search field with search keyword then hits the search button. The expected result for this behaviour is to retrieve results for the search keyword.

4.1 Compare the Different Approaches for Data Generation

At the beginning of Data generation layer, I conducted a study where I proposed two methods to identify the best way to allocate the search field and the search button in a given website.

The approaches are “Image processing approach” and “web scraping approach “.

In the first method “Image processing approach for data generation” I used the PYAUTOGUI python library. There it can locate the web element that matches a given image. Also, PYAUTOGUI can control the mouse movement on the screen. In the implementation, test Automation located the web page elements using library PYAUTOGUI successfully. Then, test automation interacted with the webpage elements to perform the search test case script. My findings were as follows, first of all, the PYAUTOGUI library needs access over the mouse and the screen. That means while executing the test automation, PYGUIAUTO takes over the mouse movement. Thus, a tester cannot move the mouse or open other windows while running the Automation script as the PYGUIAYTO won’t be able to locate the element if it was hidden under another screen. Most importantly, if a tester moves the mouse while running the test script that will cause a wrong calculation of the mouse cursor position which will affect the test Automation execution. Secondly, I found that PYGUIAUTO takes so long to apply the image comparison and processing to locate the web elements. Lastly, while using this approach for different websites, different sets of images are needed as there are different icons and themes per different websites. That means, a test Automation should be provided with different Images for different websites.

On the other hand, for “The web scraping approach for data generation”, I found that I can execute the test in the background and still be able to use the running device for other purposes. Unlike the “Image processing approach”, the device’s screen that runs the test Automation should only be dedicated for the test. One more advantage for the web scraping approach is that there is no pre-training needed or data preparation to perform scraping as most of the websites use the same HTML structure and tags naming. Most importantly, web scraping is much faster than image processing in allocating the web elements on a website.

* For the Image processing approach, it took 41703 ms to locate the web page elements and execute the test case.
* For the web scraper agent, it took 11086 ms for locating web elements and executing the test script.

To summarize, this study has shown that web scraper agent is better compared to image processing in allocating the web page elements on a website, due to different reasons. The first reason, web scraping is faster compared to Image processing. Also, a web scraper agent does not require data preparation to perform the task. Finally, Test Automation that uses web scrapers can be run in the background unlike the image processing approach that needs a dedicated device for running the test automation. For all of the above reasons, I decided to use web scraping for this test Automation.

4.2 Self-hEALING TEST Automation

Self-healing automation is the ability to carry out the task of locating web page elements successfully even when the environment has changed [[2](https://link.springer.com/chapter/10.1007/978-3-031-10464-0_24#ref-CR2)]. For normal test Automation that does not include self-healing feature, Testers use the HTML attribute “ID” in locating the webpage elements as it is the best way for allocating the webpage elements [[7](https://link.springer.com/chapter/10.1007/978-3-031-10464-0_24#ref-CR7)]. Testers use the HTML attributes ID as a hardcoded value in the testing script. The problem with this approach is that the “ID” of the webpage elements changes frequently. Thus, testers should keep updating the ID values in the testing script accordingly.

In my approach, the proposed Test Automation used scraper agent to perform web scraping to allocate the web page elements. I designed the scraper agent to dynamically identify the HTML attribute “name” for the web element. The HTML attribute “name” is used to allocate the XPATH for the web elements. Later, the Selenium web driver uses the XPATH to allocate the element on the website and interacts with it. Also, I created a memory text file for the scraper agent to save HTML attribute “name” per each run. The test Automation checks the memory text file to see if existing data can help the agent in locating any web elements before performing a scraping. I was able to perform a self-healing test Automation as the scraper agent was able to allocate the search field and search button for different websites without any supervision using one selector only which is the Xpath.

To conclude, the proposed Test Automation used scraper agent to perform web scraping to allocate the web page elements, like the search field and the search button. I found that the scraper agent was able to allocate the search field and search button for different websites (environment) then perform the search test case scenario of filling the field and clicking on the search button.

Moreover, I found that scraper agent has improved the test automation by learning from each run by interacting with new webpage elements and then saving the data in the memory file.

4.3 Data Collection by Extracting Search Endpoint

In the Data generation layer, after performing the search scenario on the webpage, Test Automation was able to collect all HTTP requests generated. Most importantly, extract the search endpoint from a group of requests. In this approach, I collected every request generated by the test Automation using selenium wire library. I was able to extract the search endpoint successfully for all targeted websites. Table [3](https://link.springer.com/chapter/10.1007/978-3-031-10464-0_24#Tab3) shows the search endpoint collected per each website.

| **URL** | **Search endpoint** |
| --- | --- |
| [https://www.google.com](https://www.google.com/) | <https://www.google.com/search?q=master> |
| [https://www.amazon.com](https://www.amazon.com/) | <https://www.amazon.com/s/ref=nb_sb_noss_2?url=search-alias%3Daps&field-keywords=master> |
| <https://old.bau.edu.jo/> | <http://old.bau.edu.jo/SearchResults.aspx?q=master> |
| <https://www.reddit.com/> | <https://gateway.reddit.com/desktopapi/v1/q=master> |
| <https://www.bbc.co.uk/search> | <https://www.bbc.co.uk/search?q=master&page=1> |
| <https://uk.yahoo.com/?p=us> | <https://uk.search.yahoo.com/search?p=master&fr=yfp-t&ei=UTF-8&fp=1> |
| <https://www.nottingham.ac.uk/search.aspx> | <https://www.ox.ac.uk/search?query=master> |
| <https://www.ox.ac.uk/> | <https://www.leeds.ac.uk/search?q=master&searchOption=searchSite> |
| <https://www.leeds.ac.uk/> | <https://stackoverflow.com/search?q=master> |
| <https://stackoverflow.com/> | <https://www.ebay.com/sch/i.html?_from=R40&_trksid=p2380057.m570.l1313&_nkw=master&_sacat=0> |

**Table 3. Search endpoints generated by the test automation per each website**

[**Full size table**](https://link.springer.com/chapter/10.1007/978-3-031-10464-0_24/tables/3)

4.4 Data Evaluation

In this phase, Search HTTP endpoint is evaluated if it is passed or failed. In order to do so, a test oracle is needed. Therefore, I had proposed two alternative ways to provide a test oracle for the search functionality for a given website. First method is to use the Fuzzy Inference system model. Second method is to use normal probability logic of adding different cases using if- else statements and for each case I added assert True or assert False.

The Fuzzy Logic Data Evaluation Experiment

This experiment uses the Data evaluation Layer of the system that has the Fuzzy inference system. I run the test automation script once per each website and recorded the following:

1. 1.

Search Endpoint Request.

1. 2.

Search Endpoint status code.

1. 3.

Running Time for the fuzzy Logic system layer.

Below Table [4](https://link.springer.com/chapter/10.1007/978-3-031-10464-0_24#Tab4) have details for The Fuzzy Logic approach experiment.

| **URL** | **Fuzzy logic time (ms)** | **Fuzzy-system results** |
| --- | --- | --- |
| [https://www.google.com](https://www.google.com/) | 10 | Passed |
| [https://www.amazon.com](https://www.amazon.com/) | 8 | Passed |
| <https://old.bau.edu.jo/> | 6 | Failed |
| <https://www.reddit.com/> | 8 | Passed |
| <https://www.bbc.co.uk/search> | 7 | Passed |
| <https://uk.yahoo.com/?p=us> | 6 | Passed |
| <https://www.nottingham.ac.uk/search.aspx> | 7 | Passed |
| <https://www.ox.ac.uk/> | 8 | Passed |
| <https://www.leeds.ac.uk/> | 15 | Passed |
| <https://stackoverflow.com/> | 7 | Passed |

**Table 4. Fuzzy system experiment result for testing search on a given website**

[**Full size table**](https://link.springer.com/chapter/10.1007/978-3-031-10464-0_24/tables/4)

The All-Possible Conditions Using If- else Approach Experiment

The second experiment was to run the test automation script again, once per each website as well but this time using the alternative layer for data evaluating. The alternative Layer has a predefined testcase with all possible conditions using If-else statements. Below Table [5](https://link.springer.com/chapter/10.1007/978-3-031-10464-0_24#Tab5) that has details for All possible cases approach.

| **URL** | **All-possible cases run time (ms)** | **All-possible cases results** |
| --- | --- | --- |
| [https://www.google.com](https://www.google.com/) | 0 | Passed |
| [https://www.amazon.com](https://www.amazon.com/) | 0 | Passed |
| <https://old.bau.edu.jo/> | 0 | Failed |
| <https://www.reddit.com/> | 0 | Passed |
| <https://www.bbc.co.uk/search> | 0 | Passed |
| <https://uk.yahoo.com/?p=us> | 0 | Passed |
| <https://www.nottingham.ac.uk/search.aspx> | 0 | Passed |
| <https://www.ox.ac.uk/> | 0 | Passed |
| <https://www.leeds.ac.uk/> | 0 | Passed |
| <https://stackoverflow.com/> | 0 | Passed |

**Table 5. Result of all-possible conditions using if- else approach experiment**

[**Full size table**](https://link.springer.com/chapter/10.1007/978-3-031-10464-0_24/tables/5)

Comparing Experiments Result

Based on the result from the experiments demonstrated in Table [3](https://link.springer.com/chapter/10.1007/978-3-031-10464-0_24#Tab3) and Table [4](https://link.springer.com/chapter/10.1007/978-3-031-10464-0_24#Tab4), the Prediction proposed by fuzzy Logic shows that the inference system was 100% accurate in prediction compared to “All possible cases testing results” in verifying the search functionality for different websites.

With a deeper look into the Fuzzy inference system layer, only six rules added in the inference system. The Number of overall assertions needed was two; assert False if the value of the crisp output for the fuzzy inference system is less than 85%. Otherwise, assert True.

On the other hand, “All possible cases testing results” had twelve test-cases in total each case has assertion based on the scenario.

For the performance of both data evaluation layers:

1. 1.

The Fuzzy inference system running time did not exceed 30 ms which is a reasonable time compared to the advantages of providing a test oracle.

1. 2.

For “All possible cases data evaluation Layer” the running time was 0 ms.

Based on the results of both experiments, Nine out of ten websites search functionality appeared to work as expected. However, Test Automation has detected a failure in one website. Based on the Test Automation result the search endpoint generated correctly but the rate of the existence for the search keyword in the body of the response was low. Thus, the test Automation detected the failure. I checked the website by manual testing and confirmed that for the website the search functionality did not work.

To conclude, based on the result, I approved that the test oracle provided from the Fuzzy inference system was 100% correct for all ten websites. The running time for the Fuzzy inference system is reasonable. Also, the Fuzzy Inference System has less rules to cover the same test cases compared to the “All possible test cases “approach.

4.5 Detecting Failure’s Layer

The main advantage of the designed test automation is to locate the failure layer. To do so I used the following Rules:

* If the issue appeared in the Data generation layer where no search end point generated then the issue of the search functionality is related to the front-end layer (Web client level).
* If the issue appeared in the data evaluation that means the issue is in the backend layer (server-side level).

For example, if the test result has passed that means the server has returned the correct results for the search endpoint request. The experiment for testing the search functionality for “BAU” website’s result has failed in the Data Evaluation Level. That means the issue is on the back-end (Server-side level).

To conclude, I found that the proposed approach of test Automation of evaluating the functionality for a client side (website UI) using a server-side level is useful in detecting the failure layer for the search functionality.

4.6 Challenges and Limitation

In the Data Generation Layer, for the web scraper agent, the main challenge related to using selenium webdriver is that some websites do not follow the stranded of HTML tags and names. Because of that, the task of scraping becomes harder for the agent, for example, in some websites the form’s action is not defined as “search”. To overcome this issue, I provided the agent with a list of common names and titles used for the search field and button like (value = “submit”) for the search button which is commonly used across different websites.

Another challenge was the performance of the Test Automation. Obviously, Test Automation takes a long time while running. As a lot of sleep code and waiting for the page to load was inserted in the code. The reason for inserting wait and sleep code is to make sure all of the HTML elements are displayed on the screen.

Another challenge faced me during collecting the search Endpoint, was that Targeted website detected the AI bot when using the “request” library when calling the search Endpoint as a normal API. Thus, blocking the request. To overcome this issue, I had to use the Selenium web driver to call the GET request. However, such an approach would not work for POST requests. So, in the future other ways should be implemented to avoid bot detection.

5 Conclusions

This Chapter describes the project’s key findings and future work. First of all, I have presented a “Gray Box Testing” approach for testing websites, where the verification for the search function for a website is applied on both, client-side and server-side levels. I approved the efficiency of this approach in identifying the defect layer.

As a result of using this testing approach, I was able to detect the layer that caused the failure in the search functionality for a website.

Secondly, I conducted experiments to compare two AI mechanisms that allocate the web page elements on a website. The first mechanism is web scraping that locates the XPath of the element based on the HTML attribute “name”. The second mechanism is Image processing using PYAUTOGUI. Based on the experiments, I found that web scraper is better than Image processing in locating the web elements as it is faster and reliable.

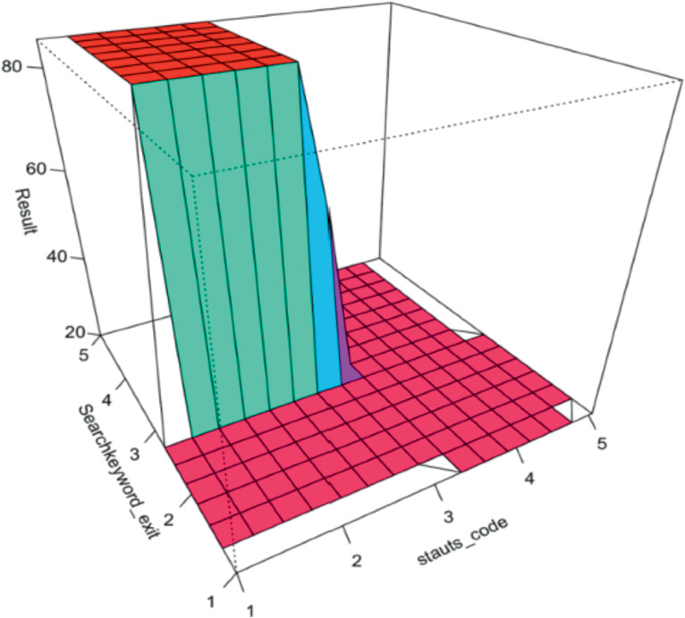
Thirdly, I have presented an approach of a self-healing test Automation. That is, a web scraper agent is used to locate web page elements dynamically using the HTML Attribute “name”. Even when the attribute name for the web element changes, the test Automation is still able to identify it again by applying web scraping. Consequently, test Automation can perform on different websites as the web scraper agent locates the web elements dynamically. Furthermore, the scraper agent learns from previous runs and saves the Attribute- name in a memory file. The Test Automation checks this memory file every run before applying the scraping.

Lastly, the oracle problem is known to be the main obstacle in test automation. This work proposed a test automation which utilizes an intelligent decision-making algorithm known as fuzzy logic by using Fuzzy set theory which classifies the inputs to predict the output. For evaluating the Search Endpoint correctness, this novel approach can predict any possible results for a combination of two inputs. Indeed, this work approved that the Fuzzy inference system is a great artificial intelligent approach that can provide a test oracle for functional testing applied on web applications.

Consequently, Experiment results have been further analysed to determine the correctness of the prediction. The results revealed, when the test case was marked by the Fuzzy inference system as “failed”, that indicated an issue on the website search functionality. On the other hand, when the Fuzzy inference system marked the test case as passed, then the search functionality was approved to be working as expected.

These results imply that the Fuzzy Inference System model is approved to be effective for predicting the functional test result for the given website. Furthermore, For the Fuzzy Inference system, testers only need to define the rules for the inputs, and then any possible case is covered by the Fuzzy inference system. The 3D plot for the output surface of the Fuzzy Inference System can help to visualise the test oracle for a combination of two inputs. In Fig. [7](https://link.springer.com/chapter/10.1007/978-3-031-10464-0_24#Fig7) below, the high peak illustrates when the result is passed for the combination of two inputs (x-axis is the first input: status code, y-axis is the second input: is the search keyword rate). The low area in the 3D plot shows when the result must fail.

**Fig. 7.**

[](https://link.springer.com/chapter/10.1007/978-3-031-10464-0_24/figures/7)

Visualisation for the test oracle determined by the fuzzy system for combination of inputs on X axis and Y axis. The test oracle result represented by the Z axis

[**Full size image**](https://link.springer.com/chapter/10.1007/978-3-031-10464-0_24/figures/7)

For future this work can be extended to implement different AI algorithms for predicting a test oracle like Bayesian Belief Network. Furthermore, evaluate the results of the Fuzzy Inference System.

References

1. Bird, S.: NLTK: the natural language toolkit. In: Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions, pp. 69–72 (Jul 2006)

[**Google Scholar**](https://scholar.google.com/scholar?&q=Bird%2C%20S.%3A%20NLTK%3A%20the%20natural%20language%20toolkit.%20In%3A%20Proceedings%20of%20the%20COLING%2FACL%202006%20Interactive%20Presentation%20Sessions%2C%20pp.%2069%E2%80%9372%20%28Jul%202006%29)

1. Hudaib, A.A., Fakhouri, H.N., Al Adwan, F.E., Fakhouri, S.N.: A survey about self-healing systems (desktop and web application). Commun. Netw. **9**(01), 71 (2017)

[**CrossRef**](https://doi.org/10.4236%2Fcn.2017.91004)[**Google Scholar**](https://scholar.google.com/scholar_lookup?&title=A%20survey%20about%20self-healing%20systems%20%28desktop%20and%20web%20application%29&journal=Commun.%20Netw.&volume=9&issue=01&publication_year=2017&author=Hudaib%2CAA&author=Fakhouri%2CHN&author=Al%20Adwan%2CFE&author=Fakhouri%2CSN)

1. Korel, B., Al-Yami, A.M.: Assertion-oriented automated test data generation. In: Proceedings of IEEE 18th International Conference on Software Engineering, pp. 71–80. IEEE (Mar 1996)

[**Google Scholar**](https://scholar.google.com/scholar?&q=Korel%2C%20B.%2C%20Al-Yami%2C%20A.M.%3A%20Assertion-oriented%20automated%20test%20data%20generation.%20In%3A%20Proceedings%20of%20IEEE%2018th%20International%20Conference%20on%20Software%20Engineering%2C%20pp.%2071%E2%80%9380.%20IEEE%20%28Mar%201996%29)

1. Di Lucca, G.A., Fasolino, A.R., Faralli, F., De Carlini, U.:Testing web applications. In: International Conference on Software Maintenance, 2002. Proceedings, pp. 310–319. IEEE (Oct 2002)

[**Google Scholar**](https://scholar.google.com/scholar?&q=Di%20Lucca%2C%20G.A.%2C%20Fasolino%2C%20A.R.%2C%20Faralli%2C%20F.%2C%20De%20Carlini%2C%20U.%3ATesting%20web%20applications.%20In%3A%20International%20Conference%20on%20Software%20Maintenance%2C%202002.%20Proceedings%2C%20pp.%20310%E2%80%93319.%20IEEE%20%28Oct%202002%29)

1. Di Lucca, G.A., Fasolino, A.R.: Testing web-based applications: the state of the art and future trends. Inf. Softw. Technol. **48**(12), 1172–1211 (2006)

[**CrossRef**](https://doi.org/10.1016%2Fj.infsof.2006.06.006)[**Google Scholar**](https://scholar.google.com/scholar_lookup?&title=Testing%20web-based%20applications%3A%20the%20state%20of%20the%20art%20and%20future%20trends&journal=Inf.%20Softw.%20Technol.&volume=48&issue=12&pages=1172-1211&publication_year=2006&author=Lucca%2CGA&author=Fasolino%2CAR)

1. Monsefi, A.K., Zakeri, B., Samsam, S., Khashehchi, M.: Performing software test oracle based on deep neural network with fuzzy inference system. In: Grandinetti, L., Mirtaheri, S.L., Shahbazian, R. (eds.) High-Performance Computing and Big Data Analysis: Second International Congress, TopHPC 2019, Tehran, Iran, April 23–25, 2019, Revised Selected Papers, pp. 406–417. Springer International Publishing, Cham (2019). <https://doi.org/10.1007/978-3-030-33495-6_31>

[**CrossRef**](https://doi.org/10.1007%2F978-3-030-33495-6_31)[**Google Scholar**](https://scholar.google.com/scholar_lookup?&title=Performing%20software%20test%20oracle%20based%20on%20deep%20neural%20network%20with%20fuzzy%20inference%20system&pages=406-417&publication_year=2019&author=Monsefi%2CAK&author=Zakeri%2CB&author=Samsam%2CS&author=Khashehchi%2CM)

1. Ramya, P., Sindhura, V., Sagar, P.V.: Testing using selenium web driver. In: 2017 Second International Conference on Electrical, Computer and Communication Technologies (ICECCT), pp. 1–7. IEEE (Feb 2017)

[**Google Scholar**](https://scholar.google.com/scholar?&q=Ramya%2C%20P.%2C%20Sindhura%2C%20V.%2C%20Sagar%2C%20P.V.%3A%20Testing%20using%20selenium%20web%20driver.%20In%3A%202017%20Second%20International%20Conference%20on%20Electrical%2C%20Computer%20and%20Communication%20Technologies%20%28ICECCT%29%2C%20pp.%201%E2%80%937.%20IEEE%20%28Feb%202017%29)

1. Rekik, R., Kallel, I.: Fuzz-Web: A methodology based on fuzzy logic for assessing web sites. Int. J. Comput. Inf. Syst. Ind. Manag. Appl. **5**, 126–136 (2013)

[**Google Scholar**](https://scholar.google.com/scholar_lookup?&title=Fuzz-Web%3A%20A%20methodology%20based%20on%20fuzzy%20logic%20for%20assessing%20web%20sites&journal=Int.%20J.%20Comput.%20Inf.%20Syst.%20Ind.%20Manag.%20Appl.&volume=5&pages=126-136&publication_year=2013&author=Rekik%2CR&author=Kallel%2CI)

1. Trudova, A., Dolezel, M., Buchalcevova, A.: Artificial intelligence in software test automation: a systematic (2020)

[**Google Scholar**](https://scholar.google.com/scholar?&q=Trudova%2C%20A.%2C%20Dolezel%2C%20M.%2C%20Buchalcevova%2C%20A.%3A%20Artificial%20intelligence%20in%20software%20test%20automation%3A%20a%20systematic%20%282020%29)

1. Zadeh, L.A.: Fuzzy logic. Computer **21**(4), 83–93 (1988)

[**CrossRef**](https://doi.org/10.1109%2F2.53)[**Google Scholar**](https://scholar.google.com/scholar_lookup?&title=Fuzzy%20logic&journal=Computer&volume=21&issue=4&pages=83-93&publication_year=1988&author=Zadeh%2CLA)

1. Malhotra, R., Sharma: Application of adaptive neuro-fuzzy inference system for predicting software change proneness. In: 2013 In International Conference on Information Technology. IEEE

[**Google Scholar**](https://scholar.google.com/scholar?&q=Malhotra%2C%20R.%2C%20Sharma%3A%20Application%20of%20adaptive%20neuro-fuzzy%20inference%20system%20for%20predicting%20software%20change%20proneness.%20In%3A%202013%20In%20International%20Conference%20on%20Information%20Technology.%20IEEE)

1. Alhassan, J.K., Misra, S., Umar, A., Maskeliūnas, R., Damaševičius, R., Adewumi, A.: A fuzzy classifier-based penetration testing for web applications. In: International Conference on Information Technology & Systems, 95–104. Springer, Cham (Jan 2018)

[**Google Scholar**](https://scholar.google.com/scholar?&q=Alhassan%2C%20J.K.%2C%20Misra%2C%20S.%2C%20Umar%2C%20A.%2C%20Maskeli%C5%ABnas%2C%20R.%2C%20Dama%C5%A1evi%C4%8Dius%2C%20R.%2C%20Adewumi%2C%20A.%3A%20A%20fuzzy%20classifier-based%20penetration%20testing%20for%20web%20applications.%20In%3A%20International%20Conference%20on%20Information%20Technology%20%26%20Systems%2C%2095%E2%80%93104.%20Springer%2C%20Cham%20%28Jan%202018%29)

1. Kaur, S., Gupta, S.K.: A fuzzy-based framework for evaluation of website design quality index. Int. J. Digit. Libr. **22**(1), 15–47 (2020). <https://doi.org/10.1007/s00799-020-00292-6>

[**CrossRef**](https://doi.org/10.1007%2Fs00799-020-00292-6)[**Google Scholar**](https://scholar.google.com/scholar_lookup?&title=A%20fuzzy-based%20framework%20for%20evaluation%20of%20website%20design%20quality%20index&journal=Int.%20J.%20Digit.%20Libr.&doi=10.1007%2Fs00799-020-00292-6&volume=22&issue=1&pages=15-47&publication_year=2020&author=Kaur%2CS&author=Gupta%2CSK)

1. King, T.M., Arbon, J., Santiago, D., Adamo, D., Chin, W., Shanmugam, R.: AI for testing today and tomorrow: industry perspectives. In: 2019 IEEE International Conference On Artificial Intelligence Testing (AITest), 81–88. IEEE (Apr 2019)

[**Google Scholar**](https://scholar.google.com/scholar?&q=King%2C%20T.M.%2C%20Arbon%2C%20J.%2C%20Santiago%2C%20D.%2C%20Adamo%2C%20D.%2C%20Chin%2C%20W.%2C%20Shanmugam%2C%20R.%3A%20AI%20for%20testing%20today%20and%20tomorrow%3A%20industry%20perspectives.%20In%3A%202019%20IEEE%20International%20Conference%20On%20Artificial%20Intelligence%20Testing%20%28AITest%29%2C%2081%E2%80%9388.%20IEEE%20%28Apr%202019%29)

[**Download references**](https://citation-needed.springer.com/v2/references/10.1007/978-3-031-10464-0_24?format=refman&flavour=references)

Author information

Authors and Affiliations

1. **The University of Nottingham, Nottingham, NG7 2RD, UK**

Dalia Alamleh

Corresponding author

Correspondence to [Dalia Alamleh](mailto:dalia.amleh@yahoo.com).

Editor information

Editors and Affiliations

1. **Saga University, Saga, Japan**

Kohei Arai