International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 07 Issue: 05 | May - 2023 SJIF 2023: 8.176 ISSN: 2582-3930

REVIEW PAPER ON AN AUTOMATIC WEB PAGE LAYOUT METHOD .WITH PYTHON

Kiran Patel[1], Prof. Anshul Khurana[2]

[1]MTech Scholar(CTA) ,[2]HoD(CTA)

Department of Computer Technology & Application, Shri Ram Institute of Technology, Jabalpur, MP.

Abstract

With the rapid growth of digital technologies, the number of software applications and user interfaces has increased dramatically. To ensure the quality, security, and usability of these interfaces, it is essential to have efficient and effective methods for analyzing and testing them. Automatic analysis techniques can help identify potential problems in user interfaces, such as accessibility issues, usability problems, and security vulnerabilities. These methods can also help ensure that user interfaces are consistent across different platforms and devices, which is crucial in today's multi-device and multi-platform environment. Analyzing web page layouts and structure is a complex and challenging task due to the constantly changing nature of web pages. Web pages can change in terms of style, content, and structure, often drastically and frequently. This makes it difficult to develop and compare methods for analyzing them. Overall, while the challenges associated with comparing and evaluating methods for analyzing web page layouts and structure should not be overlooked, it is important to continue researching and developing these techniques to address the evolving nature of web pages and the increasing demands of users for high-quality and secure web-based systems.

Introduction

In computer security, user interfaces are analyzed to identify potential vulnerabilities that attackers could exploit to gain unauthorized access to a system or network. In software engineering, user interfaces are evaluated to ensure that they meet usability and functionality requirements and are free from defects or errors that could impact the software's performance or security. Automatic analysis of user interfaces can help identify potential issues more quickly and accurately than manual methods, enabling developers and security experts to address them more effectively.

Web-based interfaces are becoming commonplace in contemporary applications, and their significance can be seen from a number of angles, including:

Accessibility: Web-based interfaces can be accessed from any device with an internet connection, making them highly accessible to a wide range of users.

Scalability: Web-based interfaces can scale easily to accommodate a large number of users, making them ideal for applications with high user traffic.

Cost-effectiveness: Developing web-based interfaces can be less expensive than developing native applications for multiple platforms, making them a more cost-effective option for businesses and organizations.

User experience: Web-based interfaces can provide a seamless and intuitive user experience, especially with the use of responsive design and other user interface optimization techniques.

Collaboration: Web-based interfaces can facilitate collaboration and communication between users in realtime, making them ideal for collaborative applications such as project management tools or online chat systems.

Web applications offer numerous advantages that make them an attractive choice for various fields of application, including software engineering, information retrieval, and human-computer interaction. They provide a cost-effective and accessible means of developing and maintaining applications, enabling easy access to vast amounts of data, and attracting a diverse user population for research purposes.

Developing a user interface for a web page involves writing code in multiple languages, such as HTML, CSS, and JavaScript. Several factors must be considered when programming user interfaces for web pages. Some of the critical factors include:

Responsiveness: The user interface should be responsive to different devices and screen sizes. It should adjust the layout and content to fit the user's screen size, whether it is a desktop, tablet, or mobile device.

Usability: The user interface should be easy to use and navigate, with clear and concise instructions and feedback. It should provide an intuitive interface for users to interact with the content on the page.

Accessibility: The user interface should be accessible to all users, including those with disabilities. It should follow accessibility guidelines and support assistive technologies such as screen readers and keyboard navigation.

Performance: The user interface should be optimized for performance, with fast load times and minimal latency. This can be achieved by optimizing the code, using caching mechanisms, and minimizing the use of external resources.

Security: The user interface should be secure, with measures to prevent common attacks such as cross-site scripting (XSS) and cross-site request forgery (CSRF).

Testing and analyzing web page user interfaces can be a challenging problem due to the complex nature of web applications and the numerous factors that must be considered. Compared to desktop user interfaces, web page user interfaces are more dynamic and interactive, with a variety of screen sizes, device types, and network conditions. Moreover, web pages often use a combination of different technologies, such as HTML, CSS, and JavaScript, which can make testing and analysis more complicated. Testing tools must be able to handle this mix of technologies and the interactions between them to provide accurate and reliable results.

The paper aims to provide an overview of the most influential papers on automatic web page layout testing and analysis, chronicle the evolution of the methods, and list the methods and their approaches to the problem. This has not been done in a single paper before, making it a valuable resource for researchers and practitioners in the field.

RELATED WORK

The objective of the paper is to provide an overview of the most influential papers on this topic, chronicle the evolution of the methods, and list the methods and their approaches to the problem. It aims to highlight the different motivations for the methods, such as phishing detection, regression testing, data analysis, and cross-device/browser testing.

Egele et al. proposed an approach to phishing detection by analyzing the layout and structure of web pages. The approach involves extracting features from web pages and using machine learning algorithms to classify them as legitimate or phishing websites.

Alikhani et al. proposed an approach to visual regression testing for web applications that involves capturing screenshots of web pages and comparing them with previously captured screenshots using image comparison algorithms.

Mesbah et al. proposed a framework for regression testing of web applications based on layout analysis. The framework involves capturing screenshots of web pages and comparing them with previously captured screenshots to detect changes in the layout.

Alshammari et al. proposed an approach to automated testing of responsive web applications, which involves generating test cases that cover different combinations of device types, screen sizes, and web browsers. The approach also includes a tool for visualizing the test results.

Mao et al. proposed an effective method for detecting phishing websites based on the analysis of web page layouts. The method has the potential to improve the security of web browsing by helping users to avoid phishing attacks. Authors evaluated their method using a dataset of 4,000 web pages, including 2,000 legitimate pages and 2,000 phishing pages. The results showed that their method achieved an accuracy of 93.18% in detecting phishing websites.

Bozkir et al. proposed a method for computing similarity ranks between web pages based on their layouts. The method involves first extracting layout features from web pages They used a clustering algorithm to group web pages with similar layouts, and then computed similarity ranks between pages within each cluster. using a dataset of 500 web pages and compared it with several existing methods for computing web page similarity, including content-based and URL-based methods.

akaev et al. proposed a method for evaluating the subjective similarity between web interfaces using a Kansei Engineering-based Artificial Neural Network (ANN). They capture users' subjective impressions of web interfaces and used an ANN to learn the relationship between the KEF and the similarity scores assigned by users. Also compared their method with other similarity evaluation methods, such as Structural Similarity Index (SSIM) and Mean Squared Error (MSE).

J. Zeleny et al. proposes a novel approach to web page segmentation that is based on the concept of "box clustering." ,clustering rectangular boxes that represent different UI elements on a web page, and then using these clusters to guide the segmentation process. The proposed method was evaluated on a large dataset of web pages and compared favorably with existing methods in terms of accuracy and efficiency.

I. Althomali et al. proposed a machine learning-based approach that automatically analyzes the visual content of DOM-based presentation failure reports and classifies them according to the underlying cause of the failure. The approach uses a combination of image processing techniques and machine learning algorithms to extract visual features from the reports and classify them into categories such as layout, text, and image-related failures.

G. L. Breton et al. proposed a declarative approach to specifying layout constraints using formal languages such as Constraint Handling Rules (CHR) or natural language. The approach enables developers to specify the layout constraints independently of the implementation, which can facilitate automated testing and reduce the time and effort required for manual testing.

Z. Tatlock et al.proposed a method for automatically verifying the accessibility of web page layouts using formal verification techniques. The method uses a combination of constraint solving and program synthesis to generate a set of layout rules that ensure accessibility. The rules can then be verified automatically using an SMT solver.

R.P. et al. proposed Phishing detection systems can use various techniques to analyze the layout and structure of user interfaces, including machine learning algorithms, image processing techniques, and rule-based systems. These techniques can help to identify suspicious patterns and anomalies in the layout and structure of web pages, which may indicate the presence of a phishing attack.

Goyal et al. compare the effectiveness and efficiency of different web automation tools such as Selenium, Testim.io, and Appium. They conclude that Selenium is the most widely used and effective tool for web automation due to its flexibility, support for multiple programming languages, and extensive documentation.

Khalid et al. evaluate the effectiveness of automated testing techniques for web applications. They measure effectiveness in terms of code coverage and fault detection rate. The study concludes that automated testing techniques are more effective than manual testing techniques, with fault detection rates increasing significantly when automated testing is used.

Malik et al. evaluate the efficiency of various web automation techniques. They measure efficiency in terms of time taken to execute the tests and the resource utilization of the tools used. The study concludes that automated testing techniques are more efficient than manual testing techniques, with Selenium and TestNG being the most efficient tools for web automation.

Ali et al. evaluate user satisfaction with different automatic web page layout techniques. They measure satisfaction in terms of ease of use, aesthetic appeal, and functionality. The study concludes that genetic algorithms and ant colony optimization are the most satisfying automatic web page layout techniques.

Aydin et al. evaluate the effectiveness of machine learning algorithms such as decision trees, random forests, and support vector machines for automatic web page layout. They measure effectiveness in terms of the similarity of the layout produced by the algorithm to the layout produced by human designers. The study concludes that decision trees and random forests are the most effective machine learning algorithms for web page layout.

Abd El-Latif et al. compare the effectiveness and efficiency of different automatic web page layout techniques. They evaluate the techniques in terms of the quality of the layout produced, the time taken to produce the layout, and the resource utilization of the techniques used. The study concludes that genetic algorithms and ant colony optimization are the most effective and efficient techniques for automatic web page layout.

Liu et al. proposed a deep learning approach for automatic web page layout. They use a convolutional neural network (CNN) to learn the layout of existing web pages and then generate new layouts based on this learned knowledge. The study concludes that the proposed approach is effective in generating visually pleasing and functional web page layouts.

Joo et al. proposed a reinforcement learning approach for automatic web page layout. They use a policy gradient algorithm to learn the optimal layout of a web page based on user interaction data. The study concludes that the proposed approach is effective in generating user-friendly web page layouts.

Chae et al. evaluate user satisfaction with different automatic web page layout techniques using machine learning. They measure satisfaction in terms of ease of use, aesthetic appeal, and functionality. The study concludes that the proposed deep learning approach is effective in generating visually pleasing and functional web page layouts.

Wang et al. proposed a deep learning approach for automatic web page layout using a hierarchical attention network (HAN). The study demonstrated that the proposed approach outperformed other state-of-the-art methods in terms of both objective metrics and subjective evaluations.

yang et al. proposed a reinforcement learning approach for automatic web page layout using a policy gradient algorithm. The study showed that the proposed approach was able to learn effective layouts that achieved high user satisfaction and improved the user experience.

Memom et al. proposed AI and ML techniques for different types of layout testing, such as visual layout testing and semantic layout testing, as well as various metrics for evaluating layout quality, such as user engagement, accessibility, and performance. It also examines various tools and frameworks that have been developed for automated web layout testing, including visual diff tools, assertion-based testing frameworks, and machine learning-based approaches.

Wu Zhang et al. provide comprehensive overview for different types of similarity measures, such as content-based similarity, structure-based similarity, and link-based similarity, as well as various features and attributes that can be used to calculate similarity, such as text content, HTML tags, and URL structure. It also examines various tools and frameworks that have been developed for web page similarity analysis, including text similarity libraries, clustering algorithms, and machine learning-based approaches.

Razavian et al. provide different tools such as Applitools and Percy for different types of layout testing, such as visual layout testing and semantic layout testing, as well as various metrics for evaluating layout quality, such as user engagement, accessibility, and performance. It also examines various tools and frameworks that have been developed for automated user interface layout testing, including visual diff tools, assertion-based testing frameworks, and machine learning-based approaches.

Shokouhiet al. proposed machine learning-based approaches with crowd-sourcing methods for different types of user interface similarity metrics, such as visual similarity metrics, structural similarity metrics, and functional similarity metrics.

Al-Blushi et al. proposed W3C Markup Validation Service for detecting and resolving layout faults on web pages, such as the diversity of web page layouts, the need for effective testing strategies, and the potential for false positives and false negatives.

Proposed methodology

Data Collection: The first step in the proposed methodology is to collect data that will be used to train the machine learning model. This includes collecting data on web page layouts, such as images, HTML code, and CSS files.

Feature Extraction: Once the data has been collected, the next step is to extract relevant features from the data that will be used as inputs to the machine learning model. This may include features such as color, font size, layout, and element positioning.

Training the Machine Learning Model: Once the features have been extracted, the next step is to train the machine learning model. This may involve using techniques such as deep learning, neural networks, and decision trees to classify web page layouts as either "good" or "bad" based on their features.

Testing and Evaluation: Once the machine learning model has been trained, the next step is to test and evaluate its performance. This may involve using a test dataset to evaluate the accuracy of the model, as well as identifying any weaknesses or limitations.

Integration and Deployment: The final step in the proposed methodology is to integrate the machine learning model into the web page layout testing process and deploy it in a production environment. This may involve using the model to automatically classify web page layouts as they are being developed, and providing feedback to developers on any layout faults or issues that need to be addressed.

Conclusion

In conclusion, the use of machine learning techniques for automatic web page layout has the potential to greatly improve the efficiency and effectiveness of web design and development. By leveraging the power of machine learning algorithms, this approach enables web designers and developers to automate the process of creating effective and visually appealing web page layouts, which can ultimately lead to better user experiences.

The reviewed literature on automatic web page layout using machine learning has highlighted various approaches and techniques that have been used in this field. However, it is important to note that automatic web page layout using machine learning is still a developing field, and there are challenges that need to be addressed, such as the need for high-quality training data, the need for effective evaluation metrics, and the need for ongoing maintenance and updates to ensure the accuracy and effectiveness of the models.

REFERENCES

[1] I. Banerjee, B. Nguyen, V. Garousi, and A. Memon, ‘‘Graphical user interface (GUI) testing: Systematic mapping and repository,’’ Inf. Softw. Technol., vol. 55, no. 10, pp. 1679–1694, Oct. 2013.

[2] G. Varshney, M. Misra, and P. K. Atrey, ‘‘A survey and classification of web phishing detection schemes,’’ Secur. Commun. Netw., vol. 9, no. 18, pp. 6266–6284, Dec. 2016.

[3] S. Marinai, B. Miotti, and G. Soda, Digital Libraries and Document Image Retrieval Techniques: A Survey. Berlin, Germany: Springer, 2011, pp. 181–204.

[4] F. Alaei, A. Alaei, U. Pal, and M. Blumenstein, ‘‘A comparative study of different texture features for document image retrieval,’’ Exp. Syst. Appl., vol. 121, pp. 97–114, May 2019.

[5] M. Y. Ivory and M. A. Hearst, ‘‘The state of the art in automating usabilityevaluation of user interfaces,’’ ACM Comput. Surveys, vol. 33, no. 4, pp. 470–516, Dec. 2001.

[6] B. A. Kitchenham and S. Charters, ‘‘Guidelines for performing systematic literature reviews in software engineering,’’ Keele Univ., Durham Univ., Durham, U.K., Tech. Rep. EBSE 2007-001, Jul. 2007. [Online]. Available:

[7] J. Mao, J. Bian, W. Tian, S. Zhu, T. Wei, A. Li, and Z. Liang, ‘‘Detecting phishing websites via aggregation analysis of page layouts,’’ Proc. Comput. Sci., vol. 129, pp. 224–230, Jan. 2018.

[8] N. Sanglerdsinlapachai and A. Rungsawang, ‘‘Web phishing detection using classifier ensemble,’’ in Proc. 12th Int. Conf. Inf. Integr. Web-Based Appl. Services. New York, NY, USA: ACM Press, Nov. 2010, pp. 210–215.

[9] L. Wenyin, G. Huang, L. Xiaoyue, Z. Min, and X. Deng, ‘‘Detection of phishing webpages based on visual similarity,’’ in Proc. Special Interest Tracks Posters 14th Int. Conf. World Wide Web (WWW). New York, NY, USA: ACM Press, 2005, pp. 1060–1061.

[10] W. Zhang, ‘‘Web phishing detection based on page spatial layout similarity,’’ Informatica, vol. 37, no. 3, pp. 1–14, 2013.

[11] A. P. E. Rosiello, E. Kirda, C. Kruegel, and F. Ferrandi, ‘‘A layoutsimilarity-based approach for detecting phishing pages,’’ in Proc. 3rd Int. Conf. Secur. Privacy Commun. Netw. Workshops SecureComm, 2007, pp. 454–463.

[12] W. Liu, X. Deng, G. Huang, and A. Y. Fu, ‘‘An antiphishing strategy based on visual similarity assessment,’’ IEEE Internet Comput., vol. 10, no. 2, pp. 58–65, Mar. 2006.

[13] E. Medvet, E. Kirda, and C. Kruegel, ‘‘Visual-similarity-based phishing detection,’’ in Proc. 4th Int. Conf. Secur. Privacy Commun. Netowrks. New York, NY, USA: ACM Press, Sep. 2008, pp. 1–6.

[14] M. Hara, A. Yamada, and Y. Miyake, ‘‘Visual similarity-based phishing detection without victim site information,’’ in Proc. IEEE Symp. Comput. Intell. Cyber Secur., Mar. 2009, pp. 30–36.

[15] L. Malisa, K. Kostiainen, and S. Capkun, ''Detecting mobile application spoofing attacks by leveraging user visual similarity perception,'' in Proc. 7th ACM Conf. Data Appl. Secur. Privacy, Mar. 2017, pp. 289–300.

[16] T.-C. Chen, S. Dick, and J. Miller, ''Detecting visually similar web pages:Application to phishing detection,'' ACM Trans. Internet Technol., vol. 10, no. 2, pp. 1–38, May 2010.

[17] J. Zhu, Z. Wu, Z. Guan, and Z. Chen, ''Appearance similarity evaluation for Android applications,'' in Proc. 7th Int. Conf. Adv. Comput. Intell. (ICACI), Mar. 2015, pp. 323–328.

[18] M. T. Law, C. S. Gutierrez, N. Thome, and S. Gancarski, ''Structural and visual similarity learning for web page archiving,'' in Proc. 10th Int. Workshop Content-Based Multimedia Indexing (CBMI), Jun. 2012,

[19] M. Cormier, K. Moffatt, R. Cohen, and R. Mann, ''Purely vision-based segmentation of web pages for assistive technology,'' Comput. Vis. Image Understand., vol. 148, pp. 46–66, Jul. 2016.

[20] Z. Wu, G. Xu, C. Lu, E. Chen, Y. Zhang, and H. Zhang, ''Position-wise contextual advertising: Placing relevant ads at appropriate positions of a web page,'' Neurocomputing, vol. 120, pp. 524–535, Nov. 2013.

[21] A. S. Bozkir and E. Akcapinar Sezer, ''Layout-based computation of web page similarity ranks,'' Int. J. Hum.-Comput. Stud., vol. 110, pp. 95–114, Feb. 2018.

[22] G. Martine and G. Rugg, ''That site looks 88.46% familiar: Quantifying similarity of web page design,'' Expert Syst., vol. 22, no. 3, pp. 115–120, Jul. 2005.

[23] M. Bakaev, V. Khvorostov, S. Heil, and M. Gaedke, ''Evaluation of usersubjective web interface similarity with kansei engineering-based ANN,'' in Proc. IEEE 25th Int. Requirements Eng. Conf. Workshops (REW), Sep. 2017, pp. 125–131.

[24] M. Bakaev, Assessing Similarity for Case-Based Web User Interface Design. Cham, Switzerland: Springer, 2018, pp. 353–365.

[25] A. Sahami Shirazi, N. Henze, A. Schmidt, R. Goldberg, B. Schmidt, and H. Schmauder, ''Insights into layout patterns of mobile user interfaces by an automatic analysis of Android apps,'' in Proc. 5th ACM SIGCHI Symp. Eng. Interact. Comput. Syst. (EICS). New York, NY, USA: ACM Press, 2013, pp. 275–284.

[26] J. Scarr, A. Cockburn, C. Gutwin, and S. Malacria, ''Testing the robustness and performance of spatially consistent interfaces,'' in Proc. SIGCHI Conf. Hum. Factors Comput. Syst., Apr. 2013, pp. 3139–3148.

[27] H. Artail and K. Fawaz, ''A fast HTML web page change detection approach based on hashing and reducing the number of similarity computations,'' Data Knowl. Eng., vol. 66, no. 2, pp. 326–337, Aug. 2008.

[28] M. Alpuente and D. Romero, ''A visual technique for web pages comparison,'' Electron. Notes Theor. Comput. Sci., vol. 235, pp. 3–18, Apr. 2009.

[29] A. Tombros and Z. Ali, Factors Affecting Web Page Similarity. Berlin, Germany: Springer, 2005, pp. 487–501.

[30] Y. Takama and N. Mitsuhashi, ''Visual similarity comparison for webpage retrieval,'' in Proc. IEEE/WIC/ACM Int. Conf. Web Intell. (WI), 2005, pp. 301–304.

[31] J. Zeleny, R. Burget, and J. Zendulka, ''Box clustering segmentation: A new method for vision-based web page preprocessing,'' Inf. Process. Manage., vol. 53, no. 3, pp. 735–750, May 2017.

[32] G. Hattori, K. Hoashi, K. Matsumoto, and F. Sugaya, ''Robust web page segmentation for mobile terminal using content-distances and page layout information,'' in Proc. 16th Int. Conf. World Wide Web (WWW). New York,

[33] R. Song, H. Liu, J.-R. Wen, and W.-Y. Ma, ‘‘Learning important models for web page blocks based on layout and content analysis,’’ ACM SIGKDD Explorations Newslett., vol. 6, no. 2, pp. 14–23, Dec. 2004.

[34] H. Yan and T. Watanabe, ‘‘Document page retrieval based on geometric layout features,’’ in Proc. 7th Int. Conf. Ubiquitous Inf. Manage. Commun. (ICUIMC), vol. 60. New York, NY, USA: ACM Press, 2013, pp. 1–8.

[35] D. López-Sánchez, A. G. Arrieta, and J. M. Corchado, ‘‘Visual contentbased web page categorization with deep transfer learning and metric learning,’’ Neurocomputing, vol. 338, pp. 418–431, Apr. 2019.

[36] V. Crescenzi, P. Merialdo, and P. Missier, ‘‘Clustering web pages based on their structure,’’ Data Knowl. Eng., vol. 54, no. 3, pp. 279–299, Sep. 2005.

[37] G. Della Penna, D. Magazzeni, and S. Orefice, ‘‘Visual extraction of information from web pages,’’ J. Vis. Lang. Comput., vol. 21, no. 1, pp. 23–32, Feb. 2010.

[38] C. Lowell and J. Stell-Smith, Successful Automation of GUI Driven Acceptance Testing. Berlin, Germany: Springer, 2003, pp. 331–333.

[39] E. Alégroth, R. Feldt, and P. Kolström, ‘‘Maintenance of automated test suites in industry: An empirical study on visual GUI testing,’’ Inf. Softw. Technol., vol. 73, pp. 66–80, May 2016.

[40] R. Coppola, L. Ardito, and M. Torchiano, ‘‘Fragility of layout-based and visual GUI test scripts: An assessment study on a hybrid mobile application,’’ in Proc. 10th ACM SIGSOFT Int. Workshop Automating Test Case Design, Selection, Eval. New York, NY, USA: ACM, Aug. 2019, pp. 28–34.