Automated usability evaluation utilizing log files and data mining techniques.

Sima Shafaei
University of Louisville

AUTOMATED USABILITY EVALUATION UTILIZING LOG FILES AND DATA MINING TECHNIQUES

By

Sima Shafaei
M.Sc., University of Isfahan, 2008
M.Sc., Isfahan University of Technology, 2005

A Dissertation
Submitted to the Faculty of the
J. B. Speed School of Engineering of the University of Louisville
in Partial Fulfillment of the Requirements
for the Degree of

Doctor of Philosophy in Industrial Engineering

Department of Industrial Engineering
University of Louisville
Louisville, Kentucky

December 2023

AUTOMATED USABILITY EVALUATION UTILIZING LOG FILES AND DATA MINING TECHNIQUES

By

Sima Shafaei
B.Sc., Isfahan University of Technology, 2005
M.Sc., University of Isfahan, 2008

A Dissertation Submitted on

Nov 9, 2023

By the following Dissertation Committee:

DEDICATION

This dissertation is dedicated to my husband

Mr. Ryan Mahdi Samani

and

to my mother

Minoo Ghafarian

who have given me invaluable support.

ACKNOWLEDGMENTS

I extend my heartfelt appreciation to my advisor, Dr. Jason Saleem, for his invaluable guidance, unwavering support, and patience. Gratitude is also due to the esteemed committee members, Dr. Monica Gentili, Dr. Gail DePuy, and Dr. Olfa Nasraoui, for their insightful feedback and assistance throughout my educational journey.

Special thanks are reserved for my husband, Ryan Mahdi Samani, who has been my pillar of strength and most enthusiastic cheerleader. I am deeply grateful to my mother, Minoo Ghafarian, for her enduring patience. During moments of uncertainty, she provided encouragement that kept me persevering. I also express my thanks to my family members—Sara, Mahsa, Arsalan, and Shahryar—for their unwavering support over the years.

ABSTRACT

AUTOMATED USABILITY EVALUATION UTILIZING LOG FILES

AND DATA MINING TECHNIQUES

Sima Shafaei

November 9, 2023

Usability evaluation is one of the essential aspects of software production. This evaluation should be done during the entire life cycle of a software application, from pre-production to production and post-production. However, the collection and evaluation of usability data can be a very challenging, time-consuming, and expensive task to be conducted manually, particularly for certain types of products and working conditions. These challenges may include the need to recruit participants fully engage and motivate them during evaluation, and factor in environmental conditions. Other challenges may include collecting data in real-world environments, especially when the users are geographically dispersed, minimizing evaluator and participant bias, and analyzing complex data sets, particularly when the volume of data is large.

This research explores an alternative approach to automate the whole process of usability evaluation by utilizing data mining and machine learning techniques on data recorded in log files. The objective of this dissertation is to extract the value of usability-related metrics from user interactions and estimate usability measures in a quantitative manner. The use of log files for usability evaluation offers several advantages over

traditional methods of evaluation, such as collecting data objectively without the need for subjective interpretation of the evaluator in some cases, creating a comprehensive view of user interactions with the possibility of identifying user behavioral patterns and trends that appear in large data sets, and collecting data from real-world scenarios instead of data from simulated scenarios. Other advantages include reducing evaluation costs, enabling remote data collection from anywhere in the world, which leads to the identification of location-dependent usability problems, and facilitating continuous monitoring over time, which leads to the identification of time-dependent usability issues.

In this dissertation, **Chapter II** provides a comprehensive categorization, comparison, and summary of the pertinent usability evaluation techniques that utilize log files as input data in both academic and industrial research. Each method is examined carefully, and its respective strengths and weaknesses are highlighted to provide a systematic understanding of the advantages and limitations of each technique. **Chapter III** assesses the originality of the research questions and proposed solutions. Given the ongoing development and refinement of log file analysis tools and techniques by proficient teams in advanced corporations, and the substantial research already conducted in web usage mining, this dissertation compares its solution with prior works in these two domains. The similarities and disparities between them are evaluated to determine the uniqueness and value of the approach advocated in this research. In **Chapter IV**, the complexities of data collection are explored, highlighting its prominence as one of the foremost challenges when dealing with log files. This chapter examines the generation of simulated data, as well as the collection of real data. In order to generate synthetic log files that closely resemble real log files and reflect the same challenges, a Bayesian networks model is proposed. This

model includes nodes at the highest level that can be numerically measured, such as the number of task actions, entries, words per item, percentage of items with missing information, percentage of legible items, and other relevant variables. By assigning values to these variables at the highest level, log files can be generated based on measurable and understandable factors. Moreover, the values can be adjusted in each iteration to produce a new simulated log file of any desired size. **Chapter V** introduces a general framework for estimating usability metrics and attributes through log file analysis. The systematic steps of the presented framework are followed to introduce the essential models for log file analysis and knowledge extraction. Within the knowledge extraction phase, a two-stage clustering approach that leverages similarity distance and Hidden Markov Models (HMM) is employed to identify page view sequences related to each task. In the subsequent knowledge analysis stage, the required data for calculating usability metrics is extracted and computed. Finally, the outcomes of the experiments are shared, solidifying the model's effectiveness. A summary of overall conclusion from this dissertation is presented at the end of Chapter V, demonstrating the unique contributions of this work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Background and Motivation

"In software engineering, usability is the degree to which a software can be used by specified consumers to achieve quantified objectives with effectiveness, efficiency, and satisfaction in a quantified context of use. (International Organization for Standardization, 1998). Usability evaluation and improvement are important not just for individuals who want to have a positive experience with a product, but also for companies and manufacturers to survive in today's competitive market and generate more profit by attracting more customers and enhancing their reputation.

There are a variety of methodologies available for evaluating software usability, including formative and summative usability testing, focus groups, card sorting, wireframe testing, first-click testing, and surveys. Irrespective of the kind of methodology, there is a common underlying procedure for all of them that can be broken down into five steps: 1) definition of metrics, 2) choosing subjects, 3) collecting data, 4) cleansing data, 5) extracting measures from data, and 6) analyzing results. Currently, the process of data collection and preparation in these methods is typically performed manually (i.e., facilitated by a usability specialist) using observation, interview, and/or survey. Consequently, the evaluation results are derived from the information collected from a limited number of users. Although there is previous work that investigated the use of log files for analyzing an application’s usability, much of that work is dated and there is a need to revisit this to leverage newer approaches for log file analysis. A log file is a machine-generated file that records all events or user actions in a certain application, system, or website.

In this dissertation, the first objective is to establish a general comprehensive framework that covers all the usability evaluation steps through the analysis of log files (from the data collection step to analyzing the results.). The second objective is to determine metrics that can be derived from the information captured in the log files and formulate them to provide a measurable assessment of usability. Then, these quantitative metrics will be extracted from log files by employing machine learning and data mining techniques to track user behavior and extract patterns and errors.

While using log files as the source of information presents certain challenges, it also provides a revolutionary opportunity to evolve usability evaluation and reach a larger population of end-users compared to the traditional methods. We can define the following steps to achieve this goal.

1) Since log files are unstructured text files, we first need to parse these files and convert them into a structured database.

2) As with all research in the field of data mining, after the creation of the dataset, we need to perform a data cleansing process and extract the appropriate features.

3) Next, we need to determine which usability metrics can be measured using the information contained in the log files and propose a methodology for extracting the value of these metrics.

4) Then, we can apply various machine learning techniques such as pattern mining, sequence mining, clustering, and many more to determine the metrics values identified in Step 3.

5) Finally, the method will be evaluated and compared with state-of-the-art models to determine its effectiveness in achieving the study objectives.

There is a lot of research in the field of web usage mining to apply web server log files to gain insights into user behavior and usage pattern in order to enhance marketing, advertising, or website structure. However, there is limited recent research on exploring the utilization of these data for usability evaluation. Much of the previous research in this area is dated and does not leverage newer approaches for log file analysis. Further, there does not seem to be a comprehensive framework that includes all the usability evaluation steps using the log files (from the data collection step to analyzing the results). Therefore, the chosen problem area is promising and warrants further investigation as a potential research avenue.

1.1.1 What is Usability?

There are many different definitions for usability according to the context in which it is mentioned and also the perspective it comes from, such as psychology, management, development, and users' point of view. By combining them and extracting a common thread, we can assert that usability is one of the system's qualities that concerns with effective interaction of users and products and how easily a product can be operated. One of the most authoritative definitions for usability is provided by ISO 9241-11 as " the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use". In this context, effectiveness specifies achieving the intended goal(s); efficiency is about the resources, such as time or effort needed by users to achieve their goals; and satisfaction deals with the comfort and positive attitudes towards the use of the product (Organizacion Internacional de Normatizacion - ISO, 2018).

1.1.1.1 Usability Evaluation Methods

During different phases of the product lifecycle, including design, development, and testing, many crucial research questions arise. For instance, how should I categorize information on a website to make it more understandable for users and to make it easier and faster for users to find what they want? How easy is it for users to work with the application in the initial use? What modifications can be made to speed up accomplishing tasks in an application? Can international users work with and learn this application easily? The quality of the final product significantly depends on finding correct answers to these questions.

There are plenty of usability research methods available, each capable of answering a range of questions at various stages of the product development process. Some of these methods are for generating data or insights and others support ongoing activities to get things done. So, how one should know which usability research method to do when? Determining the appropriate research method requires careful consideration of several factors such as product development stage, context, budget, and specific needs. To achieve this, certain criteria can be considered.

First think about watching or asking. This leads us to select between behavioral and attitudinal research methods respectively. Behavioral techniques are based on observing users' actions whereas attitudinal techniques rely on users' self-reported perceptions. When the research question has to do with what people actually do, or whether something is discoverable, findable, understandable, or usable, it's most informative to observe them. So, a method like a usability test is recommended. On the other hand, when the research objective is to learn insights into people's opinions, such as their preferences or aversions towards something, a self-reported approach involving interviews, surveys, and focus groups can be utilized.

Another criterion to help us decide which research method to use, is whether the research questions require quantitative or qualitative answers. If the objective is to compare several products or designs, obtain benchmarks, or calculate cost savings from design modifications, then a quantitative research method such as card sorting or tree testing should be employed. The flip side of that is qualitative studies that are ideal for obtaining quick and inexpensive insights so the design can be iterated and improved as much as possible before releasing it.

Additionally, the required number of individuals to involve in a study can sometimes be a critical factor. In some methods, evaluation can be done by a limited number of users or experts while others necessitate the participation of a large group of individuals in the study to produce reliable results. Although this number may vary depending on various factors, such as the scope of the test, the complexity of the product, the diversity of the user population, and the goals of the study, the minimum number required to conduct the evaluation can be a deciding factor. Furthermore, to choose a method, it is essential to determine the primary purpose of the study and the time of performing it. One method may work better for comparing two different versions of a product design while the other better suits for time analysis and performance of a single design. Certain methods are highly effective in the early stages of production, when only sketches or wireframes are available, while others are most appropriate for post-development stages. Meanwhile, certain methods can be applied in every phase of production (Nielsen Norman Group, 2023; Rohrer, 2014).

There exist many usability evaluation methods. Nielsen & Molich (1990) classified all of them into four categories: 1) Formal methods which are structured and systematic approaches and work based on a set of predefined rules and procedures; 2) Automated methods, which computerize the procedure and apply tools to automatically collect data and evaluate the usability of a system; 3) empirical methods, a group of research-based methods that involve collecting and analyzing data from experiments with test users; and 4) Informal or heuristic methods, which are subjective and flexible approaches that rely on rules of thumb and evaluators' personal experience and general skill. These methods simply work by looking at the interface and passing judgment according to their own expert assessment as a usability analyst.

Providing an in-depth explanation of all usability evaluation methods is beyond the scope of this dissertation. However, to have an overview of their capabilities, Table 1 lists several prominent user interface evaluation methods and shows their primary focus and measuring instrument. A measuring instrument refers to a specialized tool or technique used to gather and generate quantitative or qualitative data that can aid in the assessment of usability. The selection of an appropriate measuring instrument depends on the type of data needed, the research questions, and the target user group. They can be either subjective or objective. Objective measuring instruments are employed to obtain quantifiable data on user behavior or performance. Common examples of objective measuring instruments include benchmarks, log files, eye-tracking, and clickstream and navigation paths. On the other hand, subjective measuring instruments are utilized to gather user opinions, feedback, and perceptions. Examples of subjective measuring instruments include questionnaires, interviews, and the verbalized thoughts of users in a think-aloud protocol (Hartson & Pyla, 2012).

Table 1: A number of prominent user interface evaluation methods, their primary focus, and type of measuring instrument and/or technique they utilize

Method	**Primary Focus**	**Measuring instrument and/or technique**
Think aloud	Identifying the significant usability issues associated with the set of critical tasks	Transcribed verbalization of participants, including what they do, and notes of their difficulty
GOMS	Focus on time analysis and performance for skilled users	Systematic documentation of small-scale steps of experienced user's action and the time of each step
User testing/ usability testing	Evaluating the product or service with real users	Observation of users working with the product
Heuristic evaluation/ expert evaluation	Identifying the significant usability issues based on predefined rules of thumbs and expert experience	Guidelines, set of rules of thumb, And general skill and experience of evaluators + Interface design
Cognitive walkthroughs	focus on first-time use	Interface prototype and walk through scenario
Pluralistic walkthroughs	focus on first-time use with multiple stakeholders concurrently	Interface prototype and walk through scenario
Card Sorting	Find the best way to organize and group items	Grouping and labeling content and features by actual users
Log file analysis	Identify usability problems by analyzing information recorded in the log files and monitoring performance	Information recorded in server, proxy, or client-side log files
Web testing software (WebArch, Web Trends	Identify usability problems by analyzing transactions between users and the website and monitoring performance	Server-side log files
Eye tracking	Determining cognitive and visual impacts of user interface (UI) elements.	Recorded information by eye tracker, including person's eye movements and gaze patterns
Clickstream analysis	Focus on identifying problems by analyzing users' navigation paths	Recorded information by a software tool that can track and record the user's interactions with a website or digital product
A/B Testing	Comparing two design options	Recorded information by the A/B testing software tool such as Optimizely, VWO, and Google Optimize
Interview	Learning about users' perceptions of the design	Interview guide used to structure the interview and elicit feedback from

		users on their experience using a website or digital product.
Focus Groups	Discovering what users want from the system.	The moderator guide used to structure the discussion and gather feedback from a group of users on their experience using a website or digital product
Questionnaires	Gathering information about users, their attitudes, and behaviors	Questionnaire itself, which is used to gather feedback from users about their experience using a website or digital product.
Tree Testing	Evaluating information-architecture hierarchies	The software tool used to conduct the tree test and collect data on user behavior and performance such as Treejack, OptimalSort, and TreeTesting.com
Clustering Qualitative Comments	Analyzing qualitative data and extracting important themes in them	Comments provided by users and typically a qualitative analysis software tool that can identify patterns and themes in those comments and group them into clusters based on similarities in their content.

Table 2: A number of prominent user interface evaluation methods and their features

Method	**Phase of development**			**People involved**	**Approach**		**Behavioral/ Attitudinal**		**Context**			**Category**
	Early stage /Design	Development	Post Development		Qualitative	Quantitative	Behavioral	Attitudinal	Not Task Base	Specific tasks	Normal Usage	
Think aloud	🟦	🟦	🟦	5-8 users	🟩	▨	**B**			🟨		Formal
GOMS	🟦	🟦	🟦	5-15 Skilled users	▨	🟩		**A**		🟨		Formal
User testing/ usability testing	🟦	🟦	🟦	5-8 users	🟩	🟩	**B**			🟨		Empirical
Heuristic evaluation/ expert evaluation	🟦	🟦	🟦	3-5 experts	🟩	🟩		**A**				Informal
Cognitive walkthroughs	🟦			3-5 experts	🟩			**A**		🟨		Informal
Pluralistic walkthroughs	🟦			2-3 experts, 3-5 users	🟩			**A**		🟨		Informal
Card Sorting	🟦			20 users	🟩	🟩		**A**	🟨			Empirical
Log file analysis		▨	🟦	$\geq 10^2$ users		🟩	**B**				🟨	Automated
Web testing software		▨	🟦	$\geq 10^2$ users		🟩	**B**				🟨	Automated
Eye tracking		🟦	🟦	5-10 users	🟩	🟩	**B**			🟨	🟨	Empirical
Clickstream analysis		🟦	🟦	$\geq 10^2$ users		🟩	**B**				🟨	Empirical
A/B Testing			🟦	20-30 users		🟩	**B**				🟨	Empirical
Interview	🟦	🟦	🟦	5-8 users	🟩			**A**		🟨	🟨	Empirical
Focus Groups	🟦	🟦	🟦	6-10 users	🟩			**A**		🟨		Empirical

Questionnaires	🟦	🟦	🟦	≥ 30 users	🟩			A		🟨	🟨	Informal
Tree Testing	🟦			15-20 users		🟩	B		🟨			Formal
Clustering Qualitative Comments	🟦	🟦	🟦	≥ 5-10 users		🟩		A		🟨	🟨	Informal

Table 2 summarizes the features of the methods listed in Table 1, including in which phase of development they are better to use, the sample size range involved in the test, whether the result is qualitative or quantitative or both, their category as Behavioral/ Attitudinal method and whether they are based on normal usage, pre-specified tasks, or neither. A portion of the information presented in these tables is derived from Nielsen Norman Group research (Moran, 2018; Rohrer, 2014) and (Lewis, 1994; Nielsen, 1994a), while others are based on the opinions and experiences of the authors.

1.1.1.2 Usability Measures

According to Hartson & Pyla (2012), a usability measure is a general and abstract concept that refers to a user experience characteristic that needs to be measured with respect to the evaluation of an interaction design in order to achieve a desired UX goal. The selection of an appropriate usability measure informs the choice of suitable measuring instruments and usability metrics.

Prior literature has introduced various sets of usability measures (Galitz, 1994; Lin, Choong, & Salvendy, 1997; Nielsen, 1994b; Oppermann & Reiterer, 1997; Scapin & Bastien, 1997) Nielsen's model (Nielsen, 1994b), Shackel’s model (Shackel & Richardson, 1991), and the ISO standard (International Organization for Standardization, 1998) are widely recognized as prominent models for usability testing and evaluation. Nielsen's model, for instance, proposes five fundamental usability measures, including Efficiency, Learnability, Memorability, Satisfaction, and Error. Similarly, the ISO standard provides three usability measures, namely, Effectiveness, Efficiency, and Satisfaction. Shackel's model, on the other hand, outlines four distinct usability measures, including Effectiveness, Learnability, Flexibility, and Attitude. (Hussain, Mkpojiogu, & Jasin, 2017)

The following are some common usability measures that can be paired with quantitative metrics and their definitions (Nielsen, 1996) (Hartson & Pyla, 2012):

- **Efficiency**: Efficiency is the ratio of outcome to the resources expended and refers to the performance, speed, and ease with which user can complete their goals when they have become familiar with the system and are no longer constantly in a learning state.
- **Learnability**: Refers to how quickly and easily users can learn to use a system
- **Memorability or Retainability**: refers to the ability of casual users to remember how to use a product or service after a period of non-use.
- **Effectiveness**: Refers to the accuracy and completeness with which users achieve certain goals
- **Error**: This states that the system should have a low error rate, so users will encounter few errors. Those they do encounter should be easy to recover from. Further, catastrophic errors must not occur.
- **Advanced feature usage**: helps determine user experience of more complicated functions of a system.
- **Initial performance***:* a user's performance during the very first use (somewhere between the first few minutes and the first few hours, depending on the complexity of the system)

- **Satisfaction/ Long-term (longitudinal) user satisfaction**: States that the system should be pleasant to use. This refers to the user's opinion after using the system for some greater period of time, after some allowance for learning.
- **First impression** (initial opinion, initial satisfaction): refers to the first impression and opinion of users toward the product.

It's important to note that these are just a few examples of the many existing usability measures. The choice of specific measures depends on the context of use, the goals of the evaluation, and the characteristics of the user population.

1.1.1.3 Usability Metrics

A usability metric is a quantifiable data value that is used to obtain the value of a usability measure (Hartson & Pyla, 2012). Examples of these metrics include click-through rate, bounce rate, time on task, or task completion rate. It should be noted that the relationship between metrics and measures is a many-to-many relationship. This means that sometimes, we need to use multiple metrics to obtain a more accurate estimation of a measure. On the other hand, a single metric may contribute to the calculation of several different measures. For example, Efficiency can be determined by task completion time, clicks/taps per task, page views, and error rate. While the error rate can be one of the estimation factors of effectiveness as well.

After conducting an extensive literature review, a list of commonly used usability metrics has been compiled and listed in Table 3. This table also presents a clear definition for each metric and outlines several sample measures that can be calculated by them (Hashim & Isse, 2019; Kopanitsa, Tsvetkova, & Veseli, 2012; Oztekin, Kong, & Uysal, 2010; Scholtz, 2006; Wronikowska et al., 2021).

Table 3: List of common metrics in reviewed literature, their definition, and associated measures

#	Metric	Description	Measures
1	Task Completion Rate	The percentage of users who are able to successfully complete a task	Efficiency Learnability
2	Time on Task	The amount of time a user spends on a specific task or set of tasks. It can also be broken down into different stages of the task, such as time spent on navigation, time spent on data entry, or time spent on decision making	Effectiveness Efficiency Advanced Feature Usage
3	Task Completion Time	The time it takes for users to complete a task from start to finish	Learnability Efficiency Advanced Feature Usage
4	Throughput	The number of tasks that users can successfully complete in a given time.	Efficiency
5	Error Rate	The total number of errors made by users while completing a task.	Effectiveness Learnability Memorability Efficiency Satisfaction Initial performance Advanced Feature Usage
6	Error Count per Action	The total number of errors made by users per action or step taken	Efficiency Learnability Memorability
7	Task Completion Time on the First Attempt	The amount of time it takes for a user to complete a specific task or set of tasks on their first attempt	Learnability Initial performance
8	Error Rate on the First Attempt	number of errors made by the user during a specific task or set of tasks on their first attempt	Learnability Initial performance
9	First Time Task Success or "FTTS"	The percentage of users who are able to complete a specific task or set of tasks on their first attempt, within a defined time frame	Learnability Initial performance
10	Error and Recovery Time	The amount of time users spend in errors and recovering from them	Efficiency
11	Number of Repetitions of Failed Commands	The number of times that users repeat a failed command	Efficiency Learnability
12	Number of Actions	The total number of commands, actions, or steps taken by users to complete a task.	Efficiency
13	Success/Failure Ratio	The ratio of successful task completions to unsuccessful ones	Efficiency Learnability Memorability
14	Help/Documentation Frequency	This is a measure of how often users access help or documentation while using a product.	Efficiency Learnability Memorability

15	Time on task for Skilled Users	The time it takes users to complete a task after a certain amount of use and compares it to their initial performance.	Learnability Memorability Advanced Feature Usage
16	Error Rate for Skilled Users	The error rate after a certain amount of use and compare to the initial error rate.	Learnability Memorability Advanced Feature Usage
17	Reacquisition Time	The time it takes for users to relearn how to complete a task after a period of non-use (e.g., a week).	Memorability
18	Reacquisition Error Rate	The number of errors made by users when trying to complete a task after a period of non-use (e.g., a week).	Memorability
19	Average Cursor Movement Time	The average time users spend moving the cursor during a task	Efficiency
20	Total Mouse Movement Distance	The total distance the mouse cursor travels during a task	Efficiency
21	Clicks/Taps	The number of mouse-clicks or taps required for users to complete a task	Efficiency
22	Keystrokes	The number of keystrokes required to complete a task.	Efficiency
23	Page Views	The number of times a specific page is viewed by users.	Efficiency
24	User Satisfaction Rating by NPS (Net Promoter Score)	Measuring customers' willingness to recommend the product to a friend or colleague by subtracting Detractors (who have a low intention to recommend) from Promoters (who are loyal and have a very high intention to recommend)	Satisfaction
25	Likert Scale Rating	Measures the user's attitudes, opinions, or perceptions about a specific aspect of the product by asking the users to rate an agree/disagree question on a scale of 1-5 or 1-7	Satisfaction
26	Task Success Rate	The percentage of users who can complete a task successfully.	Effectiveness
27	Bounce Rate	The number of users who leave a website after visiting just a single page divided by the total number of users who visit the website.	Satisfaction Effectiveness
28	Time on Page	The amount of time that users spend on a certain page.	Efficiency Satisfaction
29	Conversion Rate	The proportion of users who perform a desired action, such as submitting a request or completing an order	Effectiveness
30	Retention Rate	The percentage of users who are able to remember how to complete a task after a period of time.	Learnability Memorability

31	Goal completion rate	The percentage of users who are able to achieve their overall goals using the system	Effectiveness

1.1.2 Log Files

It is hard to pinpoint the exact time of establishing the first definition of log file. However, one of the earliest definitions was proposed by Rice and Borgman in 1983. They defined transaction logs as "a data collection method that automatically captures the type, content, or time of transactions made by a person from a terminal with the system" (Rice & Borgman, 1983). Since then, log files have become an integrated part of various computer programs and systems, including web servers, security systems, database servers, and software tools that use log files to record events or transactions that occur within them.

Currently, logging is an important and essential feature of software development and its primary goal is to use this valuable data for performance monitoring, troubleshooting and analyzing bugs and other unexpected system events. Due to the fact that these “treasure troves of valuable information” maintain a record of all user transactions with the server or the user operations in an application, they are useful in many research areas such as web usage assessment, usability evaluation, or user behavior detection. Log files can be categorized into three main categories according to where they are located. 1) web servers, 2) proxy servers, and 3) clients. In the following, the characteristics of each of these log files will be elaborated and the limitations and advantages they impose will be discussed.

1.1.2.1 Client Side

A client log file, also referred to as a browser log file, is generated on the client side, usually by JavaScript or Java applets, and tracks the browsing history. For applying these types of log files for log analysis purposes, participants should remotely test a website by either downloading special software or using a modified browser that can record their web usage. HTTP cookies may also be utilized for this purpose. HTTP cookies are pieces of information generated by a web server and stored in the users' computers for future access.

In comparison to server-side and proxy-side log files, client-side logging can record more details about user interactions, such as mouse and keyboard events, the number of active windows, selected page items, and more. Therefore, actual user behavior can supplement the understanding of web users with more concrete data. However, utilizing these files imposes several drawbacks. Firstly, data collection requires more effort from both the user and evaluator because software installation may be required or specific system configuration on the users' end. Second, it is often platform-dependent, meaning that the software may only work on a specific operating system or browser. Furthermore, storing and transferring files can be problematic as there should be a mechanism for sending the logged data back to the evaluator. Additionally, sometimes the client system has a limitation on the amount of data that can be stored. Because of these problems, it is hard to effectively test a website with a wide variety of test participants, resulting in the loss of model scalability.

1.1.2.2 Proxy Side

A proxy server is responsible for receiving users' HTTP requests and forwarding them to a web server, then transmitting the results back to the users. There are two primary types of proxy servers:

- HTTP proxies: This approach requires either configuring the user’s browser or installing special software on the user's device to pass all requests through the proxy. In return, they offer high flexibility in capturing all user interactions and providing a detailed log of the interactions. An example of this type of proxy server commonly used is the corporate firewall.
- URL-based proxies: This type of proxy server requires no configuration or software installation on the user's end and is compatible with any standard web browser. Hence, it is generally easier to use for remote usability testing. In this approach, all links are redirected to the proxy server's URL, which captures only the traffic that is directed towards the URL of the proxy. The advantage of this easy and inexpensive setup is counterbalanced by its limitations in capturing some types of interactions, such as those involving secure connections (HTTPS). An example of using this approach is web anonymizers (such as www.anonymizer.com).

In addition to its numerous benefits, the utilization of log files collected from the proxy server is not without limitations. Table 4 shows the list of advantages and limitations of proxy-side logging (Hong, Heer, Waterson, & Landay, 2001).

Table 4: Advantages and limitations of proxy side logging

Advantages	Limitations
• The proxy represents a separation of concerns. Any necessary modifications needed for tracking purposes can be made on the proxy, allowing the server to focus only on serving content. Therefore, the server and its content do not have to be modified in any way. • The proxy allows anyone to run usability tests on any website, regardless of ownership. One can simply set up a proxy and ask testers to send their requests through it. • In the URL based proxies, end users are not required to make any changes to their settings to get started. This simplifies the process of logging and running usability tests on a competitor's website. • Having testers go through a proxy allows web designers to "tag" and uniquely identify each test participant. This way, designers can know who the tester was and what they were trying to do. • It does not require any special software on the client beyond a web browser, which results in faster execution. • It is much simpler to deploy. The proxy makes it easier to test a website with a wide variety of test participants, including novice users who may be unable or reluctant to download special software. • It is compatible with a wider range of operating systems and web browsers and works by modifying the HTML in a platform-independent way. This permits testing with a more realistic sample of participants, devices, and browsers.	• Users may experience some additional delays with websites when using the proxy due to overhead in retrieving and processing web pages. • The traditional proxy approach would require users to configure their browsers to use the proxy and then undo this setting after performing usability tests. This would seriously hamper the ease with which remote usability tests could be performed. • Any users who currently sit behind a firewall would be unable to participate, as changes to their proxy settings could render them unable to connect to the Internet. • In URL-based proxy, there are limitations on capturing links or redirects created dynamically by JavaScript and other browser scripting languages. As a consequence, the JavaScript-generated pop-up windows and DHTML menus popular on many websites are not captured by the proxy. • For URL-based proxy, there is a limitation on capturing embedded page components such as Java applets and Flash animations.

1.1.2.3 Server Side

Each time a user visits a website, information about their visit is collected and stored in the web server log file. There are three main standard formats for displaying web server log files: W3C Extended log file format, NCSA common log file format, and IIS log file format. These standard formats contain information like a) the IP address of the client (remote host), b) the user Id of the person requesting the document, c) the date and time that the server finished processing the request, d) the request line from the client, e) status code that the server sends back to the client, and f) the size of the object returned to the client. These data can be bound together as a single text file or divided into different logs, like access logs, referrer logs, or error logs. There are several limitations to using server logs in usability evaluation. First, these logs contain sensitive, personal information. Therefore, the server owners usually keep them closed. Second, the logs do not record cached pages visited. The cached pages are summoned from the local storage of browsers or proxy servers, not from web servers. Third, server-side logs are insufficient when trying to obtain logging data for user interaction (Suneetha & Krishnamoorthi, 2009).

1.1.3 Log File Analysis and its Applications

Log analysis refers to the process of analyzing machine-generated log files created by computer systems, sensors, or any other devices in order to derive insights and identify patterns. Log files typically contain unstructured records that include detailed information on the activities performed within the system such as the event or action description, time and date of occurrence, user identification, device, system responses, and other relevant data.

Jansen (2006) considered three major stages for analyzing a log file including collection, preparation, and analysis. Based on many other studies, reporting also can be added as a fourth stage due to its importance in communicating the analysis results.

The collection stage determines what information should be recorded in a transaction log for answering future research questions, the most appropriate method to store them, and the storage location. Accurate answers to these questions is essential for ensuring that the log is comprehensive, well-organized, and useful for further analysis.

The preparation stage includes all required preprocessing to make the data ready for the next step. This includes parsing the log file to break down all entries into their components and converting them into a structured or formatted file, normalization to ensure a consistent unit or scale for all recorded information and standardizing data formats, and data cleansing to filter out all irrelevant data, removing duplicate entries, correcting data errors, and handling missing values. At the end of this step, the log data is transformed into a more usable and accurate format, that allows for efficient and more reliable analysis.

The analysis stage is the core step that contains applying various kinds of analyses from simple filtering or query-based search on cleaned data to more complicated analyses including different statistical, rule-based, or machine learning-based techniques to extract features and uncover patterns and insights in the log file.

Finally, the reporting includes a clear and accurate representation of all results obtained in the previous stage by any available visualization tools such as tables, charts, graphs, or creating a dashboard. This step can summarize and highlight the critical findings and trends in the log file to the audience in an accessible and understandable manner. These reports may be used to identify issues that need to be addressed, track performance over time, or provide insights to stakeholders.

Log analysis is a versatile technique that has a wide range of applications. For instance, in the field of IT operations, Liu et al. (2021) have developed LogNADS, a scheme that leverages machine-learning techniques to identify anomalous network behaviors by analyzing log data. In a similar vein, Le & Zhang (2022) have demonstrated how the log data can be used to automatically detect system anomalies by applying a deep learning model.

Delias, Doumpos, Grigoroudis, Manolitzas, & Matsatsinis (2015) present a sample of log file analysis applications in the healthcare industry. This study proposes a clustering method that can handle large and complex event logs to identify patterns of activities in healthcare processes and use the result to support healthcare management decisions. Log analysis is also widely used in the finance industry. Anderka, Klerx, Priesterjahn, & Büning, (2014) used the log file provided by the ATM's D&S module as input data to learn patterns of normal behavior by utilizing automatic model generation techniques. They indicate a potential fraud attempt whenever there is a significant deviation from the learned behavior.

There are many more applications of log file analysis beyond the above examples. They can be used in a wide range of industries, including but not limited to cybersecurity, e-commerce, telecommunications, and transportation that is not feasible to cover all of them in this dissertation.

1.2 Advantages and Limitations and Challenges

In order to approach a perfect user experience, an iterative design process that involves testing interactions with real users is essential. The design of a user interface involves numerous variables, such as environmental and human cognitive factors, as well as image and audio variables, which result in an overwhelming number of possible combinations. Thus, the only reliable method for achieving effective user experience (UX) design is through rigorous testing.

There are many different ways to perform a usability test, each suitable for a different situation and purpose. These include methods such as card sorting, observation, interview, inspection, lab usability testing, guerilla testing, etc. The method of data collection and the type of data used in these methods are different. Therefore, they offer various types of analysis and results. However, what they all have in common is that data collection is done by one or more facilitators.

The method presented in this dissertation is to perform a usability evaluation based on log file data. These files are created automatically by tracking user transactions with the system. Although in some respects this method cannot provide all the detailed information obtained by other usability testing techniques, its benefits as a cheap and ready source should not be underestimated. The advantages of using log files instead of observation, interview, and inspection methods include the following:

- **Data is ready**: The collection of data is easy and fully automated. Server-side logging is auto-enabled and inexpensive to use and a proxy-side log file is also faster and easier to deploy than traditional usability testing techniques.

- **No Hawthorne effect**: This method can reduce the Hawthorne effect. The whole process of logging is hidden from the user's perspective. Therefore, it can diminish the alteration of behavior by the subjects due to their awareness of being observed.
- **Reflect actual usage**: When tracking user operations with log files, users perform tasks to meet their own goals in their natural environments, with real contexts and constraints affecting them (Hong et al., 2001). This is a benefit over usability testing, which often asks users to interact in an isolated lab environment with artificially created tasks and contexts (V. F. de F. de Santana & Baranauskas, 2008; Guzdial, 1993; Rubin & Chisnell, 2008).
- **Holistic Data:** Interaction data in server-side log files are collected from all users of a website (Hong et al., 2001), therefore this data represents the website usage of the whole population, unlike usability testing, which uses a smaller and potentially less representative sample.
- **Cost-effective**: When using log files, data collection is cheap and cost-effective. It does not require hiring usability specialists to interview users or trace their actions and behaviors in the system. This is a benefit over usability testing which is costly in terms of recording user interactions or retrieving the data (Baker, Au, Dobbie, & Warren, 2008; Hong & Landay, 2001; López, Fajardo, & Abascal, 2007).
- **Remote data collection:** Recruitment of participants to test locations could be a barrier. Using log files makes it possible to perform remote analysis.

Thus, no planning is required to record and use interaction data. This is a benefit over usability testing, which requires planning of both (V. F. de F. de Santana & Baranauskas, 2008; Guzdial, 1993; C. Menezes & Nonnecke, 2014).

- **A large number of users:** Some problems could just be highlighted in quantitative analysis and also their impact could just be evaluated by analyzing a large number of users (Vargas, Weffers, & Da Rocha, 2010). Unlike traditional methods that use a small number of subjects, log files can provide the possibility of interpretation based on usage patterns created by a large and geographically separated group of real users (Hong et al., 2001; Jorritsma, Cnossen, Dierckx, Oudkerk, & van Ooijen, 2016). Therefore, they give designers more confidence to determine the features of the future system.
- **Faithful representation of usage**: In log files, interactions are recorded exactly as they happened and do not rely on the accuracy of human visual and auditory systems, memory, and interpretation.
- **Providing additional information:** The existence of special information such as *IP address*, *referrer,* and *time stamp* in server-side log files to cursor movement distance, mouse click time interval, and *the number of left and right mouse clicks* allows us to create additional useful and interesting analyses on usage data (C. Menezes & Nonnecke, 2014).
- **No need to define benchmark tasks**: Selecting Benchmark tasks is challenging work in usability testing. They should closely represent tasks

real users will perform in a real work context. One of the advantages of using a log file is that one does not have to choose the benchmark task. The test is based on regular daily use and does not require pre-defined tasks.

Despite the above advantages of using a log file, there are some potential limitations, challenges, and concerns that cannot be overlooked:

- Log files offer less detailed testing results. They are incapable of capturing all interaction information that is important for understanding website/software usability, such as a user's thoughts, intentions, gaze, voice, a visual of the user, time spent interacting with another website or program, time spent away from the computer, or network latency (C. Menezes & Nonnecke, 2014),
- Qualitative usability testing is best for discovering problems in the user experience**.** This type of testing focuses on collecting insights, findings, and anecdotes about how people use the product or service. Log files are unable to capture qualitative information.
- Server-side log files are incapable of capturing cursor movements, clicks on non-clickable items, cursor-selected text, page scrolling, interactions with the Internet browser, and whether a link was opened in the same window or a new one (C. Menezes & Nonnecke, 2014).
- Proxy-side and server-side log files also do not capture interactions with dynamic content such as AJAX or Flash, which do not send requests to the webserver and thus do not prompt log file entries to be generated (C. Menezes & Nonnecke, 2014).

- While server-side logging is an easy way to collect user data, there are several limitations to its effectiveness that can undermine the validity of the data collected. For example, proxy servers mask users' IP addresses, making it impossible to distinguish distinct users (Burton & Walther, 2006). Additionally, Client-side caching stores previously visited webpages on the user's machine, which can prevent entries from being made in the log file when the user revisits the site (Hong et al., 2001; C. Menezes & Nonnecke, 2014). Finally, robots (bots), spiders, and web crawlers are other problem makers for server-side logging, which can cause non-human-generated interactions to be recorded in the log files (Atterer, Wnuk, & Schmidt, 2006; C. Menezes & Nonnecke, 2014).
- Although Log files come in standard formats, it is not very easy to extract meaningful information from these voluminous and low-level log data. Sometimes it takes a lot of preprocessing to be able to effectively analyze them.
- Sometimes (especially for the client-side log file), it is necessary to reproduce the software/website or change their code to create a suitable log file.

Because of these limitations and barriers, while log files are very useful and informative for usability evaluation, it is not always the only best method. However, considering the limited resources available for usability evaluation, a log data analysis could be useful as a preliminary analysis that guides the collection of other sources of usability data. This could focus other usability evaluation methods on aspects of the system with which a large number of users exhibit sub-optimal interaction patterns (Jorritsma et al., 2016). Using the results obtained from log data along with other traditional methods can considerably increase the accuracy and reliability of the evaluation. According to Balbo, Goschnick, Tong, & Paris (2005) “Even if an automated technique finds only a subset of existing problems, if it is efficient and easy to use, and could be used on areas that would otherwise receive no attention, then it makes sense to use it."

1.3 Scope and the Purpose of This Study

The objective of this research is to establish a practical, reliable, and effective approach for utilizing log file information in usability evaluation. Specifically, this research is designed to discover the barriers and facilitators to calculating numerical value of usability metrics as well as usability measures based on log file data. It is further designed to investigate to compare the accuracy and efficiency of traditional usability evaluation method with the proposed automatic method.

While the approach outlined in this study can be extended for any type of log file data, including client-, proxy-, and server-side log files of a web site, application-generated log files, or even log files generated by sensors or other devices, to reduce the complexity and keep it within the scope of a PhD dissertation, I will focus on only server-side log files, which are one of the most straightforward ways of collecting log data.

The aim of this research is to develop models and algorithms to quantify usability metrics required for estimating five main usability attributes proposed by Nielson, which include efficiency, learnability, memorability, error and user satisfaction. To validate the proposed approach, I will conduct testing on both real and synthetic log file data.

To the best of my knowledge, there are no current methods for generating synthetic log files that align with the objectives of this dissertation. Therefore, providing a method for generating a log file that accurately replicates the complexities and limitations of real log files will be a key component of this study. To achieve this goal, graphical Bayesian models that offer the unique capability of integrating expert knowledge with data-driven learning will be leveraged. Moreover, graphical Bayesian models are one of the specialized tools for data generation that is highly suited to this dissertation's needs.

The model will be fed with different features of the website such as number of tasks, tasks length distribution, number of users, roles of the system, and tasks complexity distribution to create an unlimited number of log files needed to use in future experiments. In this scenario, a good range of log files can be used with known design specifications and user configuration which will open broad opportunities for iterating experiments, and presents significant possibilities for improving and validating the usability evaluation model.

Furthermore, to ensure that the model is effective in real-world scenarios, it is crucial to test the model with a real log file. However, obtaining such data can be a challenge. While there are plenty data sets available for the log file in data center servers, in most cases, the original website that produced it or the specifications of the website at the time of generating the log file are unknown. Moreover, many website owners are unwilling to share their log files due to security or privacy concerns. Therefore, accessing a real dataset is problematic. To pass all these challenges, I have created an e-learning website and will keep it up and running for eight months to collect an appropriate size log file. This will provide the opportunity to not only compare the model with traditional methods on a real dataset, but also to validate the model by modifying the website's design and reevaluate and compare the usability of different design specifications.

1.4 Research Questions

The research questions used to advance the main goals mentioned in section 1.3 are as follows:

1. What steps should be followed when employing a log file for usability assessment? Is it feasible to establish a universal roadmap suitable for all types of log files and usability metrics?
2. What metrics can be used to measure five main usability attributes (learnability, memorability, efficiency, effectiveness, and user satisfaction) using the information stored in a server-side log file?
3. What are the effective and accurate data mining and machine learning techniques to calculate each metric identified in research question 2?
4. How can synthetic data be generated that effectively mirrors the complexities and attributes of an authentic log file, facilitating accurate modeling and assessment alongside real data?

The first question aims to outline a comprehensive framework for discerning the main steps involved in utilizing log file data, starting from the initial data collection and preprocessing and concluding with the attainment of numerical values of metrics and usability attributes. Additionally, it seeks to identify various methodologies that can be employed at each step.

The second question is designed to assemble a set of standard usability metrics that can be computed, or at least estimated using the information available in the server-side log files. These files will be studied precisely to examine the possibility as well as the methodology for obtaining the value of each metric. This examination will shed light on the viability and potential efficacy of the suggested approach.

Following the evaluation of theoretical calculations of usability metrics in the second question, the third question places greater emphasis on practical solutions. This research question focuses on identifying feasible, effective, and dependable methods for obtaining the metrics outlined in question one. In this section, each metric will be discussed in detail, specifying the algorithms and parameters that can be employed to obtain their value.

The fourth question revolves around the generation of data required for the research problem. Given the inherent difficulties in obtaining authentic log files, the use of simulated data holds significant importance. Specifically, simulating log files for websites with specific usability attributes serves two primary objectives. Firstly, it acts as a means to validate the outcomes derived from the proposed automated usability evaluation technique. Secondly, it empowers us to replicate tests using different log files, thereby enhancing the model's precision. Through the simulation of diverse scenarios, it becomes feasible to generate synthetic data mirroring the intricacies of real-world usage data. This, in turn, enables a more confident evaluation and refinement of the suggested approach. Consequently, recognizing the pivotal role of simulated data in this study, we introduce a log file simulation model encompassing various usability characteristics within this section.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

The primary objective of this chapter is to provide a systematic categorization, comparison, and summary of relevant usability evaluation techniques available in current academic and industrial research that use log files as the source of their input data. In this chapter, Section 2 introduces 13 related articles on the subject of usability evaluation using log files and briefly describes their methods and application domains. There are numerous candidate factors that can be utilized to categorize these 13 methods. However, a few significant dimensions are applicable to every project, and they will be evaluated in subsequent sections, Section 3 introduces and categorizes the types of analyses performed on log files for the aim of usability evaluation and specifies which types of analyses and data mining techniques were performed in each method. Section 4 identifies which metrics and attributes can be evaluated and estimated in each reviewed article. Finally, Section 5 summarizes the limitations and advantages of each method.

2.2 Previous Works

This section presents a concise overview of the methodologies employed in 13 scholarly articles relevant to usability evaluation through log file analysis. The models applied in these articles and their respective approaches and techniques will be critically reviewed.

Inversini, Cantoni, & Bolchini (2011) integrates usability and usage analysis to investigate the most erroneous and risky parts of a website. To determine what parts of the application require immediate attention for redesign or improvement, the developer should consider risk assessment. The authors believe that user experience risks are comprised of three main elements: (i) threats as usability problems inherent to the design; (ii) vulnerability as the exposure to usability problems and (iii) resilience as the user's ability to overcome usability problems. For usability analysis, they used a heuristic-driven evaluation method. A usability inspector with a set of usability guidelines runs a usability test with the think-aloud method. For usage analysis, they applied Google Analytics software to extract the ratio of page visits, bounce rate, average time per visit, etc. The authors believe that the pages with higher traffic have a higher vulnerability that can increase the threat. These pages can be determined by usage analysis. Moreover, a usability test can be used to determine which pages have lower resilience and higher error occurrence.

Menezes & Nonnecke (2014) developed a software tool called UX-Log to facilitate the analysis of user sessions by usability experts. This is accomplished by recreating user sessions from log files. These recreated sessions can then be examined by usability experts to gain insight into users' goals, strategies, successes or failures, and proficiencies. UX-Log provides an automated interpretation of the user's interactions; e.g., the time spent per webpage or textual description of interactions. Additionally, it offers several controls that allow experts to easily traverse through the interactions and pages visited.

Jorritsma, Cnossen, Dierckx, Oudkerk, & van Ooijen (2016) used a data mining technique called closed sequential pattern mining to discover frequently occurring interaction patterns in the log data. Given a sequence database, this technique generates all closed sequential patterns above a predefined support threshold. The authors used the CM-ClaSP[1] algorithm (Fournier-Viger, Gomariz, Campos, & Thomas, 2014) on the open-source Sequential Pattern Mining Framework (SPMF) library (Fournier-Viger, Gomariz, Gueniche, et al., 2014) to implement their method. The advantages of using this method are: first, it only mines closed patterns, which leads to clean output. Second, the user actions within the patterns it generates are sequential but not necessarily consecutive. This allows for the detection of patterns in which an action eventually leads to another action. It does not matter how many irrelevant actions occur in between.

Babaian, Lucas, & Topi (2007) present an infrastructure that facilitates automatic usability assessment for application enhancements and tests their approach in the domain of enterprise systems. This research uses a keystroke level and mouse data logging that records all users' data input in various UI components such as text fields, menus, list structures, and check boxes and stores collected data in a relational database. The task model is also represented explicitly within the system's data model. The created database links users, tasks, interface components, and the application's domain data, to enable sophisticated reasoning and analysis of the history of system usage. The results from this analysis can be employed for evaluating the usability of the system and improving the design.

[1] ClaSP is an efficient algorithm for discovering closed sequential patterns in sequence databases, proposed(Gomariz, Campos, Marin, & Goethals, 2013). CM-ClaSP is a modification of the original ClaSP algorithm using a technique co-occurrence pruning to prune the search space (Fournier-Viger, Gomariz, Campos, et al., 2014)

The research methods of Bader & Pagano (2013) consisted of three phases: a usage data collection phase, a heuristic construction phase, and an evaluation phase. In the usage data collection phase, they developed a generic tracking library that collects information about user interactions at runtime and that can be embedded into arbitrary mobile applications running on Apple's iOS1 platform. Then they used a test application and collected user interaction with it. In the heuristic construction phase, authors examined the obtained user interaction traces for regularities both manually and by applying a sequential pattern mining algorithm. Then they used the result of pattern mining to derive a heuristic classifier for low discoverability issues. This model is able to accurately identify and classify different types of usability issues, such as navigation problems and slow loading times. For example, the authors extract the problems of a mobile application by considering the length and existence of loops in interaction patterns. Finally, in the evaluation phase, they assessed the feasibility of automatically detecting low discoverability issues at runtime.

Siochi & Hix (1991) used the Most Repetitive Pattern (MRP) method to evaluate the learnability of a simple game application. The authors found that the interesting repetitions did not occur at the low level of actions. Therefore, they convert action to a more abstract form and then run MRP to find useful repetitive patterns. Then, they count the number of errors and the number of special meaningful sequences to show how users learn to work with software. In this method, although extracting patterns is automatic, analyzing and extracting heuristics from those patterns is done manually by experts.

Vargas, Weffers, & Da Rocha (2010) propose WebHint. In this tool, evaluators start by defining the tasks that they want to analyze. Then, the user interactions with the application interface are monitored. All actions performed by all users in the web application’s interface are captured including mouse movements, keystrokes, links accessed, pages loaded, etc. Then in the data preprocessing phase, the data of the different users are separated. For analyzing the prepared data there are several steps: 1) Determining sequence intervals – the algorithm finds in each log file the intervals in which there is a sequence of actions that represents the execution of a certain task. This is made by looking for representative actions in the task as the “begin” and “end” points. The algorithm also deals with incomplete tasks. The intervals found are extracted from the log. 2) Extracting executed tasks – For each selected interval (determined in step 1), a Longest Common Subsequence (LCS) algorithm is applied to compare it with the expected sequence of the referred task. The similarity rate is then calculated; if it is above a certain threshold, the sequence is extracted from the log. 3) Clustering the extracted sequences and searching for the most common patterns for the task; i.e., the most common way users performed the referred task. 4) Finally, the evaluator compares the expected sequence for the referred task with the most common patterns of execution for the task.

WebQuilt (Hong & Landay, 2001) is a web logging and visualization system that helps web design teams run usability tests (both local and remote) and analyze the collected data. Its strength lies in its data collection approach, which uses Java Servlet technology to collect data on the user's navigational path, including the sequence of links, parent pages, parent frames, and back/forward button clicks. The log file is created separately for each test participant session which is subsequently fed into an Action Inferencer component. The action inferencer converts the log file into a list of three possible actions including requesting a page, going back, or going forward. The next component is a GraphMerger that combines the list of actions performed by several users into a single graph and passes this graph to a Graph Layout component. Graph Layout assigns a location for each node of the graph to prepare it for final visualization. Finally, the visualization component shows the graph of the web pages traversed that can be used by usability experts for further interpretation and decision-making.

The creators of the tool GUITESTER (Okada & Asahi, 1999) selected ten usability attributes and showed that four of these attributes can be evaluated using log file analysis (clarity, safety, simplicity, continuity). To evaluate these four attributes, three metrics were identified for analysis and visualization: MRP for extracting the most repetitive patterns, average mouse movement, and percentage of users who performed the specified task according to the method specified by the designers.

Web Automatic Usability Testing EnviRonment (WAUTER) (Balbo et al., 2005) is another evaluation tool for a website. This method requires task models that are defined by Diane+ notation (Tarby & Barthet, 1996). This notation can express task hierarchy, task types, and temporal and logical relationships among tasks. Then, Web Interface Monitoring and Management (WIMM) is used to capture proxy-based XML-coded log files of end-user actions. Finally, the evaluators employ WEMA (Web EMA, where EMA is the French abbreviation for Automatic Mechanism for usability Evaluation), which produces both an annotated Task Model and an annotated log file. The annotations are based on a set of heuristic rules that model patterns of user behavior and highlight potential usability flaws. Five heuristics are selected to detect potential problems in the website:1) direction shift; a direction shift is detected when the user stops progressing along a set path in the tree; 2) immediate canceling of action; 3) re-occurring of actions; 4) irrelevant actions; and 5) timing.

Web Event Logger and Flow Identification Tool (WELFIT) (V. F. de Santana & Baranauskas, 2015) implements pattern analysis by extracting the usage graph and computing the Sequence Alignment Method(SAM) distance for each event stream. In this method, the data is collected at the client module. As soon as the log data reaches the size limit (2kb) it is sent to the server. In the server, a usage graph is built based on the collected data. The usage graph can also be seen as the combination of walks (non-empty alternating sequence of nodes and edges) representing what, where, and when users performed actions. In the usage graph, a node is identified by its label, which is the concatenation of the event name and an identifier of the UI element where the event occurred. Moreover, each node counts on information regarding the total of sessions that occurred, the mean distance from the root node, and the mean timestamp. This method considers SAM for measuring the distance between each node in the graph and the root node. A SAM-based heuristic (based on the SAM value of each node and its neighbors) is applied to point out usage incidents.

Automated Website Usability Analyzer (AWUSA) (Tiedtke, Märtin, & Gerth, 2002) uses both static and dynamic analysis techniques. The static website and the webpages are analyzed at design/definition time. For each web page, several metrics are generated, including the *Document Object Model (DOM),* and the visual representation of web pages are analyzed. Web content mining (using Webpage Analyzer) and web structure mining (using Website Crawler) are applied in this phase. Later in dynamic analysis, first, the log data is preprocessed by generating sessions and session paths from the access log files. Then, the usage data is compared with the defined task structure, and data mining techniques are applied to the gathered and generated data. During this dynamic analysis, web usage mining is applied for path mining and cluster generation, and detection of association rules. Finally, the defined structures and the results of the analysis are visualized. The goals of this tool include finding and visualizing users´ paths on the website, deviations between intended tasks and actual usage, locations/events where tasks are canceled prematurely, areas and situations with poor usability, and classification of different user groups and their mapping to the various tasks/goals.

Web Remote Usability Evaluator (WebRemUSINE) (Paganelli & Paterno, 2002) is composed of three phases: *Preparation*, which consists of creating the task model of the website, collecting the logged data, and defining the association between logged actions and basic tasks; *Automatic analysis*, where WebRemUSINE examines the logged data with the support of the task model and provides a number of results concerning the performed tasks, errors, and loading time; and *Evaluation*, the information generated is analyzed by the evaluators to identify usability problems and possible improvements in the interface design. Table 5 shows the domain and domain description for the software being evaluated in each paper.

Table 5: The domain for the software being evaluated

Evaluation tool	Domain title	Domain Description	Web/Non-Web
Inversini	The tourism online domain	The authors applied their model to the analysis of the BravoFly.com website (a Swiss Online Travel Agent) as a representative case.	Web
UX-LOG	Online research tool	Orlando website (Orlando is an online tool for researching women's literature in the British Isles) http://orlando.cambridge.org/	Web
Jorritsma	Medical imaging tool	The Picture Archiving and Communication System (PACS), which is the main software component of the radiology workstation	Non-Web
Babaian	Enterprise information system	A prototype Enterprise Information system deployed by authors. This prototype uses this model in the implementation of a Purchase Requisition Process from an ERP system	Non-Web
Bader	Mobile application	MoID, which is an electronic replacement for business cards available in the iOS App Store	Non-Web
Siochi	Game application	A simple computer game application	Non-Web
WebHint	On-line course environment	TelEduc is an environment for online courses where users have tools to interact with, such as a mailbox, file repository, wall, etc. A simulated	Web

		course was prepared in TelEduc for the experiment and the users were invited to perform some tasks as participants in the course	
WebQuilt	U.C. Berkeley Website	The task was to find a specific piece of information on the U.C. Berkeley website	Web
GUITESTER	GUI applications	Authors used GUITESTER to test the usability of several GUI applications including a printer driver that has dialog boxes for setting printing options and an emailer	Non Web
WAUTER	Website	CSIRO internal website They also used online booking of the cinema ticket for the task model	Web
WELFIT	Website	Generally for any website	Web
AWUSA	Website	Generally for any website	Web
WebRemUSINE	Website	Generally for any website	Web

2.3 Analysis and Data Mining Techniques in Previous Works

Based on the reviewed articles, usability analysis techniques that use log files as the source of their input data are divided into five categories: filtering, frequency analysis, time analysis, distance analysis, and pattern analysis.

- Filtering for extracting unique users and sessions from log files
- Frequency analysis is one of the simplest kinds of analysis that looks at the frequency of user interactions to infer their preferences. Page views, page exit rate (the rate at which users leave the website from a particular page), bounce rate, or frequency of performing specific tasks or actions are examples of metrics obtained in frequency analysis. This is a straightforward analysis that relies on counting the number of times a particular interaction occurs. Tools such as Google Analytics and AWStats offer various options for conducting this type of analysis.
- Time analysis looks at the amount of time spent on each page or the time of a single visit. It can indicate the degree to which a particular webpage requires cognitive effort from the user or the level of interestingness (sic) of its content. (C. Menezes & Nonnecke, 2014)

- Distance analysis looks at the amount of mouse movement on the page per visit or task. It measures the total distance between successively pointed coordinates. Distance analysis can indicate the proper design of the page, its consistency, and its simplicity.
- Pattern analysis aims to discover patterns within analytics data. It includes task analysis (for automatic task detection), user and group analysis (for clustering users based on their roles or groups), and error analysis. A variety of machine learning and data mining techniques can be used for this analysis. Association rules, for instance, are "if-then" statements, that help to show the probability of relationships between the contents of two pages, regardless of their time order access. Pairing analysis, on the other hand, determines the frequency of actions performed in a specific sequence and can help designers identify opportunities to streamline tasks by combining or reordering actions. For example, if a high percentage of users always perform two specific actions in sequence, it might be possible to combine them into a single step for greater efficiency. Clustering is a powerful technique that allows grouping users based on their behavior or preferences and can help designers create personalized user experiences and optimize the user interface. Finally, path analysis, or clickstream analysis, tracks the sequence of pages or screens visited by users, as well as the time spent on each page and the actions. By analyzing this data, designers can identify common paths that users take through the website or application, as well as

any areas where users tend to drop off or encounter issues (C. Menezes & Nonnecke, 2014).

Table 6 shows the types of analysis applied in the reviewed methods.

Table 6: Type of analysis used in previous works. The 'Visualization' column specifies whether the method provides an analytical visualization for the evaluator or not. The last column specifies whether the analysis performed is based on the information of a single user or the information collected from a group of users.

	Filtering	Frequency analysis	Distance analysis	Time analysis	Pattern analysis	Visualization	Single/ Group
Inversini	Yes	Yes	No	Yes	No	Yes	Group
UX-LOG	Yes	No	No	No	No	Yes	Single
Jorritsma	Yes	No	No	No	Yes	No	Group
Babaian	Yes	Yes	No	Yes	No	No	Both
Bader	Yes	No	No	Yes	Yes	Yes	Group
Siochi	No	No	No	No	Yes	No	Single
WebHint	Yes	No	No	No	Yes	No	Group
WebQuilt	No	No	No	Yes	Yes	Yes	Group
GUITESTER	Yes	No	Yes	Yes	Yes	Yes	Group
WAUTER	No	No	No	Yes	Yes	Yes	Single
WELFIT	No	Yes	No	No	Yes	Yes	Group
AWUSA	Yes	No	No	No	Yes	Yes	Group
WebRemUSINE	Yes	Yes	No	Yes	Yes	Yes	Both

The methods used in these articles are as follows:

- Inversini et al. (2007) utilized Google Analytics to obtain key metrics such as the page visit ratio, bounce rate, and average time per visit, in order to conduct their usage analysis.
- UXLog extract unique users and their sessions by parsing and filtering log file (Menezes & Nonnecke, 2014). They apply relational SQL databases for efficient data storage and retrieval.
- Jorritsma et al. (2007) first used filtering to extract the interaction sequence related to a specific usage of the application, along with the type of use (which includes modality and body part for their application). Then they employed the CM-ClaSP algorithm for a closed sequential pattern

mining approach with parameter support = 200 in the first run and support = 50 in the second run.

- In Babaian's work (Babaian et al., 2007), time analysis indicates spending time on each page and each action during a task. Moreover, their relational database model allows filtering and frequency analysis (counting) by performing various queries.
- Bader's pattern analysis includes a sequential pattern mining algorithm and heuristic classifier. They also employed the retention time (time spend on each page) and average session time in their time analysis process. (Bader & Pagano, 2013)
- Siochi & Hix (1991) use the Most Repetitive Pattern (MRP).
- WebHint (Vargas et al., 2010) applies LCS (Longest Common Subsequence) for selecting the potential task. It also uses the process-mining tool ProM (Van Dongen, de Medeiros, Verbeek, Weijters, & van Der Aalst, 2005) for clustering.
- WebQuilt (Hong & Landay, 2001) pattern analysis is simply limited to extracting the path taken by the user. They also use color code on the final graph to represent the amount of time spent on each page.
- GUITESTER (Okada & Asahi, 1999) uses the MRP (Most Repetitive Pattern) algorithm for detecting common interaction patterns. Then for the distance and time analysis, GUITESTER calculates the mean values for each operation in any of the detected common interaction patterns. The mean value of the distance between successively pointed coordinates and

the mean value of the interval between time stamps of successive operations are calculated.

- WELFIT (V. F. de Santana & Baranauskas, 2015) implements pattern analysis by extracting the usage graph and computing the SAM distance for each event stream.
- AWUSA (Tiedtke et al., 2002) has a usage pattern miner that is implemented by cumulating similar session paths. A similarity ratio for each pair of paths is computed, corresponding to the sessions' resource sequence. Sessions with a ratio greater than a given threshold value are cumulated to usage patterns. It also applied association rules to find associations between requested resources.
- In WebRemUSINE (Paganelli & Paterno, 2002), the visited pages and the number of accesses are reported in frequency analysis. For time analysis, they represent the time of downloading and visiting each page. Finally, in pattern analysis, they compared tasks with its model, the visit patterns during navigation, and their frequency.

2.4 Usability Metrics Extracted in Previous Works

Usability is not a single, one-dimensional property of a user interface; rather, the construct of usability includes multiple attributes. Various sets of attributes can be found (Galitz, 1994; Lin et al., 1997; Nielsen, 1994b; Oppermann & Reiterer, 1997; Scapin & Bastien, 1997). Among them, Nielsen's attributes are the most well-known and widely used in the literature. These attributes are (1) learnability, (2) efficiency, (3) memorability, (4) errors of effectiveness, and (5) satisfaction (Nielsen, 1994b). These attributes are closer to usability metrics because they can be measured in terms of the task completion time, the number of errors, or the subjective rating for questionnaires. Nielsen also showed ten usability heuristics that achieve the widest coverage with respect to explaining usability problems in his experiment (Okada & Asahi, 1999).

Although much research has been carried out on log-based usability testing tools, to the best of our knowledge, none of them measure the value of the above attributes automatically. Most research, after analyzing and visualizing the log file, allows the evaluator to find a potential problem or analyze some of these attributes.

Table 7 shows which of these five main attributes can be estimated and evaluated in each method based on the reports they provide to the evaluator. For example, to estimate effectiveness, the method should create results that help the evaluator detect the portion of completed tasks. For efficiency, the method should support time analysis as well as user error detection. To estimate learnability and memorability, we need to track temporal information regarding when errors occur and the time of performing tasks. The number of errors users make over time and how they improve their speed to perform a task can give us a good insight into the value of these attributes. For instance, if errors are concentrated in the initial phase of the test, this can mean that the user interface is easy to learn to use. Moreover, in consecutive sessions with longer intervals, if the number of errors increases again, this may mean memorability is low. Satisfaction is likely the most challenging attribute to measure directly from log file analysis due to its inherent dependence on quality assessment.

Table 7: Possibility of analyzing usability attributes in reviewed articles. The label NA means 'non-automatic' and implies that the method does not measure the value of this attribute automatically but provides some heuristics for the usability inspector to estimate them.

Evaluation tool	Effectiveness	Efficiency	Learnability	Memorability	Satisfaction	Other
Inversini	No	No	No	No	No	No
UX-LOG	Yes(NA)	Yes(NA)	No	No	No	No
Jorritsma	Yes(NA)	No	No	No	No	No
Babaian	Yes(NA)	Yes(NA)	Yes(NA)	Yes(NA)	No	No
Bader	No	Yes(NA)	Yes(NA)	Yes(NA)	No	No
Siochi	Yes(NA)	Yes(NA)	Yes(NA)	No	No	No
WebHint	Yes(NA)	Yes(NA)	No	No	No	No
WebQuilt	Yes(NA)	Yes(NA)	Yes(NA)	Yes(NA)	No	No
GUITESTER	Yes(NA)	Yes(NA)	No	No	No	Yes
WAUTER	Yes(NA)	Yes(NA)	No	No	No	No
WELFIT	Yes(NA)	Yes(NA)	No	No	No	No
AWUSA	Yes(NA)	Yes(NA)	No	No	No	No
WebRemUSINE	Yes(NA)	Yes(NA)	Yes(NA)	No	No	No

For GUITESTER, the authors discussed the possibility of evaluating four other usability attributes as well including clarity, safety, simplicity, and continuity (Okada & Asahi, 1999).

2.5 Limitations and Advantages of Previous Methods

To evaluate the limitations and advantages of each method, the evaluation steps in each method are examined for which ones are performed automatically. The whole process of analyzing the usability of a software application or website, regardless of whether the evaluation is automatic or not, can be divided into the following steps:

- **Collecting data**: The first step is to collect data. In traditional usability tests, data collection can be done through observation, questionnaires, interviews, checklists, etc. However, in automatic usability evaluation, we can use log files, videos, images, or recorded audio as the source of data. Regardless of the data analysis method, the data collection step can be automatic, non-automatic, or semi-automatic.

- **Preprocessing**: The second phase involves data preprocessing, which entails converting the information into a usable format, eliminating all useless details from the data sources, correcting or substituting incomplete or distorted data, and potentially applying data reduction or feature selection on the data. In the case of using a log file, this stage may comprise parsing the log file, normalization, cleaning the data, identifying users and sessions, and path completion. However, these processes may differ depending on the type and origin of the log file.
- **Analyzing**: In traditional usability testing such as think-aloud or interview, this step is performed by the usability facilitator or inspector. But when using the log files, as described in section 0, this step can include pattern analysis, time analysis, frequency analysis, task mining, exit analysis, user analysis, and distance analysis. Various methods of data mining, machine learning, statistical analysis, and pattern recognition can be applied in this step.
- **Quantifying usability attributes**: There are two approaches for determining or estimating the value of usability attributes. In the first way, which is a non-automatic approach, the evaluator can use the results and heuristics obtained from the analysis stage to derive the values. On the other hand, in an automatic method, according to the available heuristics and analysis reports, an

appropriate formula or model can be extracted for estimating each usability attribute. However, the objective of existing methods mainly is to identify potential usability problems instead of determining the values of these attributes. For example, possible design problems can be identified based on the errors that occurred during a task or the time of performing it.

- **Critique**: The last step is not necessary but can be highly beneficial. In this stage, according to the problems identified in the previous step, recommendations are provided to enhance the system design. For instance, a shortcut can be devised to reduce access time and expedite repetitive operations.

Considering that the main purpose of this dissertation is automatic evaluation, in

Table 8, all the reviewed works have been examined from this perspective. This table specifies which step of each method is automatic and semi-automatic or is completely performed manually by an evaluator. As can be seen, none of the proposed methods automatically estimate the usability attributes, detect potential problems or make improvement suggestions, and the performance of these steps depends on the analysis, knowledge, and opinion of the usability inspector.

Table 8: Automatic evaluation parts in reviewed methods

Evaluation tool	Data Collection	Preprocessing	Analyzing	Usability Attributes	Potential problems	Improvement suggestion
Inversini	Semi-Automatic	Semi-Automatic	Semi-automatic	Non-Automatic	Non-Automatic	Non-Automatic
UX-LOG	Automatic	Automatic	Semi-Automatic	Non-Automatic	Non-Automatic	Non-Automatic
Jorritsma	Automatic	Automatic	Semi-Automatic	Non-Automatic	Non-Automatic	Non-Automatic
Babaian	Automatic	Automatic	Semi-Automatic	Non-Automatic	Non-Automatic	Non-Automatic
Bader	Automatic	Automatic	Automatic	Non-Automatic	Automatic	Non-Automatic
Siochi	Automatic	Automatic	Semi-Automatic	Non-Automatic	Non-Automatic	Non-Automatic
WebHint	Automatic	Automatic	Semi-Automatic	Non-Automatic	Non-Automatic	Non-Automatic
WebQuilt	Automatic	Automatic	Semi-Automatic	Non-Automatic	Non-Automatic	Non-Automatic
GUITESTER	Automatic	Automatic	Automatic	Non-Automatic	Non-Automatic	Non-Automatic
WAUTER	Automatic	Automatic	Semi-Automatic	Non-Automatic	Non-Automatic	Non-Automatic
WELFIT	Automatic	Automatic	Semi-Automatic	Non-Automatic	Non-Automatic	Non-Automatic
AWUSA	Automatic	Automatic	Automatic	Non-Automatic	Non-Automatic	Non-Automatic
WebRemUSINE	Automatic	Automatic	Semi-Automatic	Non-Automatic	Non-Automatic	Non-Automatic

Semi-automatic values refer to cases where part of the work is done automatically and part of the work is left to the usability expert. For example, in UX-LOG, recreating users' sessions is automatic but analyzing and finding the problem is the expert's responsibility. Pattern detection in Jorritsma's method is automatic but finding errors remains to the usability inspector. In WebHint, defining the task model in the first step and analyzing the detected patterns, and comparing them to the expected behavior are all performed by the evaluator.

After data collection, the web designer could use the WebQuilt tools to aggregate, visualize, and interact with the data to pinpoint usability problems. In fact, WebQuilt represents the result of recreating several users' sessions but analyzing this result is on evaluators.

WAUTER analysis is also semi-automatic because it needs an evaluator to define the task model in the first step and analyze the patterns to detect potential problems in the final step.

In addition, to gain a better insight into the studied methods in terms of their strengths and limitations, Table 9 outlines the evaluation requirements for each method. The "*Knowledge About app/web*" column determines whether the evaluator needs to have an in-depth understanding of the software and its functionality. The *"Informal use/ formal use*" column specifies if the evaluation is done based on daily usage and performing arbitrary tasks (Informal) or performing only limited and predefined tasks (Formal). The "*Task model*" column indicates whether the best way of performing a task should be provided for the method or not.

The "*Reusability*" column determines if the presented model is specialized for a particular software or website. For instance, WELFIT (V. F. de Santana & Baranauskas, 2015) has no reusability as the employed heuristics about SAM distance may not apply to all applications. Similarly, Bader & Pagano's (2013) approach relies on finding loops in usage patterns for the purpose of error detection. This assumption may not be valid for all types of applications and websites.

Finally, the "*Scalability*" column denotes the ability of the model for performing evaluation for a large number of users or tasks. All cases that require evaluation of users' sessions or interactions one by one or depend on defining all task models do not have scalability such as Siochi's method. Furthermore, all the methods that require user cooperation, software installation, or adjusting settings on the users' system to collect data may also lack scalability since it will be challenging to coordinate with a large number of users. For example, GUITESTER (Okada & Asahi, 1999) requires running a logger on users' systems, which limits its scalability.

Table 9:Requirements of each method for performing evaluation and their scalability and reusability

Evaluation tool	Knowledge About app/web	Informal use/ formal use	Task model	Reusability	Scalability
Inversini	Required	Formal	No	Yes	No
UX-LOG	Required	Formal	Yes	Yes	No
Jorritsma	Required	Informal	No	No	No
Babaian	Required	Formal	Yes	No	No
Bader	Required	Informal	No	No	Yes
Siochi	Required	Informal	No	No	No
WebHint	Required	Informal	Yes	Yes	Yes
WebQuilt	Required	Formal	No	Yes	No
GUITESTER	Required	Formal	Yes	No	No
WAUTER	Required	Formal	Yes	Yes	No
WELFIT	Required	Informal	No	No	Yes
AWUSA	Required	Formal	Yes	Yes	Yes
WebRemUSINE	Required	Informal	Yes	Yes	No

2.6 This Dissertation Objectives Compared to Previous Works

To compare my model with other methods, I have selected 13 key analyses based on the objectives of this dissertation as well as insights extracted from the reviewed articles (Table 10, Figure 1). The method of this dissertation supports 8 out of 13 selected analyses. In my method, I will use task models to exclude automatic task detection. But I will categorize user's page view sequences into pre-defined tasks. Then I will perform time analysis and task analysis to detect errors and incomplete tasks. In the next step, I will analyze errors and evaluate error history to estimate learnability and memorability.

The ultimate goal is to formulate and estimate usability metrics based on log file information, which does not need session recreation or user and group analysis. Frequency analysis and creating usage graphs can be helpful but I exclude them for simplifying the process. Both Table 10 and Figure 1 illustrate how different methods support selected various analyses. Two different display formats are used for more clarity. Among the evaluated models, Babaian and WebRemUSINE (Paganelli & Paterno, 2002) have the most overlap with my proposed method in terms of performing analyses supporting 7 and 6 cases (i.e., columns in Table 10), respectively. However, both of these methods, unlike my approach are using client-side log files. Moreover, Babaian's work focuses on designing a relational database model for logged data and does not provide any automatic analysis or reasoning.

Table 10: Comparison of the proposed method with previous methods in terms of analysis and presentation of results

	Frequent Pattern Mining	Task Mining	Task Analysis	Error Detection	Error Analysis	Error History	Frequency Analysis	Time Analysis	Distance Analysis	Exit Analysis	Usage Graph	Session Recreation	User and Group Analysis
Proposed Method		■	■	■	■	■	■	■		■			
Inversini							■	■		■			
UX-LOG								■				■	
Jorritsma	■												
Babaian				■	■	■	■	■		■			■
Bader				■	■		■	■					
Siochi	■			■	■								

WebHint	■		■	■	■					■			
WebQuilt	■			■	■		■	■					
GUITESTER	■		■	■	■			■	■				
WAUTER			■	■	■			■		■			
WELFIT	■		■	■	■						■		
AWUSA	■			■	■					■			■
WebRemUSINE			■	■	■		■	■		■			

One fundamental differentiation between my model and prior research lies in the fact that, to the best of my knowledge, my approach is the first to formulate and obtain usability metrics directly and automatically from log files. While prior methods offer some analyses to facilitate estimating these metrics.

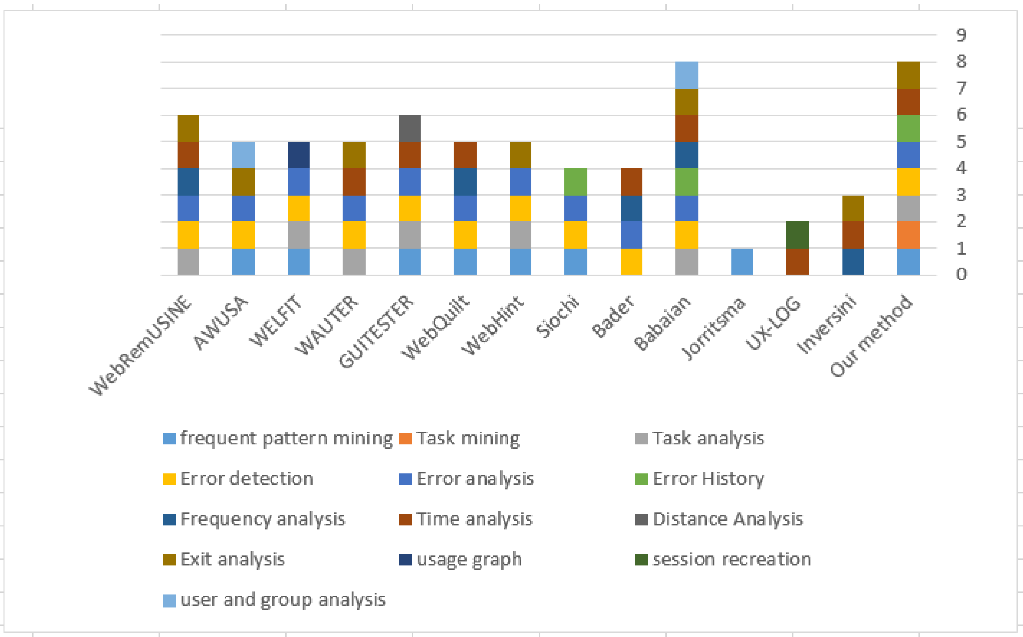


Figure 1: Key analysis coverage in each method using a stack bar chart

CHAPTER 3

COMPARISON WITH EXISTING TOOLS AND STUDIES

3.1 Introduction

This research focuses on the extraction of valuable information for usability evaluation from machine-generated files such as system-, websites-, and application-created log files, with a special emphasis on the server site website logs. As we know, there is currently a plethora of various tools and techniques available to the public for analyzing log files, which are being continually studied and enhanced by professional teams at the world's leading companies. In addition, there has been extensive research conducted in the field of web usage mining. With a simple search, one can find numerous articles and studies available on how to extract usage behavioral patterns and other insights by utilizing the data stored in log files.

Therefore, the question may be raised, does this research yield any novel accomplishments beyond what has been done so far? Can it compete with the existing strong file analysis tools like AWStats or Splunk? or does it offer additional value compared to prior research done in the realm of web usage mining?

To address these questions, the subsequent section will explore these topics in further detail. First, section 3.2 elaborates on the functionality of log analysis tools and the whole steps involved in the log analysis process. Additionally, it will provide an overview of several prominent and popular log file analysis tools, outlining their key features and capabilities. Then, section 3.3 introduces the general framework of web usage mining, the methods applied in each step, and an overview of the application domain in this field. Finally, section 3.4 compares the current study with the two above domains and clarifies their similarities and divergences in terms of three different aspects including their data source, their purpose, and their application. After reviewing this section, the reader will gain a comprehensive understanding of the position of this study on the frontier of scientific research and its association with the two above fields of study.

3.2 Log Analysis Tools

Log analysis, as a sub-branch of data analysis, is a crucial part of the IT industry where almost every product and service generates vast amounts of logs during various processes. By reviewing and analyzing log records, organizations can obtain valuable knowledge and meaningful insights that would otherwise be inaccessible. Therefore, analyzing logs became an essential need in the world of technology and the principle of supply and demand has led to the development of numerous tools designed for log analysis.

These tools utilize various techniques such as filtering, indexing, classification, clustering, and more to detect patterns, trends, and errors within log files. The output provided by them helps a wide range of professionals including security experts, system and network administrators, web developers, and reliability engineers to identify the roots of problems and guides them in making better business and security strategies.

Currently, there are numerous tools on the market for log file analysis that, along with their special and distinctive features, have remarkably similar core functionality. The main operation of these tools can be divided into the following main steps:

- **Normalization**: When the log files are coming from different sources and are produced in different formats, for further analysis, all of their elements such as IP addresses or timestamps need to be cleaned and converted into a single standard format. This is done by normalization.

- **Parsing**: The input of this section is an unstructured file that can be any type of log file such as an application, server-side, proxy side, security, etc. Its output is a data set or a structured file in various formats such as SQL or XML.
- **Filtering**: One of the simplest levels of extracting meaningful information is using filtering by applying queries to search in the log file database. A huge part of the output of many log analysis tools such as Google Analytics is produced in this step. A large portion of report information such as the average duration of the session, the number of views of each page, the number of pages viewed in each session, the bounce rate, the number of active users per day, etc. doesn’t need complicated data mining and machine learning techniques and is obtained by filtering.
- **Applying machine learning and data mining methods**: Various methods such as pattern detection and recognition, classification, correlation, etc. can be applied to extract plenty of meaningful and useful information from log files but not all common log analysis tools provide this level of analysis. By using correlation methods, it is possible to identify lines of the log file that are related to each other. These strategies are necessary for extracting events and identifying tasks. Pattern recognition methods help to identify behavioral users’ patterns in the system and spot anomalies. By applying classification techniques, we can log entries to tag and categorize users, pages, events, and tasks into classes based on different characteristics.

- **Visualization**: Log visualization, mostly in the form of the dashboard in log analysis tools, is the graphical representation of data obtained by filtering, searching, or machine learning techniques through the use of common visual elements such as charts, graphs, plots, maps, and animations. These visual representations of data-driven insights and information can make them more accessible and easier to understand. They help to see trends, outliers, and patterns in data and identify the root cause of the problems faster.

For a deeper understanding, in the rest of this section, some examples of popular log file analyzers are reviewed and for each one, it is determined which of the above parts are supported and embedded for them.

3.2.1 Splunk

Splunk (SplunkInc, 2005) is a popular commercial software platform for log analysis and log management that allows users to navigate, index, search, analyze, visualize, monitor, alert, and report on unstructured/structured machine-generated data in real-time. It is often used to derive insights from large volumes of machine data collected from virtually any source and location and provides a powerful platform for managing, monitoring, and troubleshooting complex IT environments. Splunk's search processing language, along with its machine learning toolkit (MLKT) enables simple searches as well as advanced data exploration. Users not only are able to search, filter, and analyze data using a variety of techniques, including keyword search, field extraction, and statistical analysis but also can apply MLKT to detect outliers and anomalies, as well as to perform predictive analytics and clustering algorithms. Using this toolkit, for example, user behavior analytics including threats detection and anomalous behavior recognition is possible. Real-time processing is one of the most outstanding features and maybe the biggest selling point of Splunk. Other key features are flexibility and extensibility, visualization, and security capabilities. All these features and capabilities make it a popular choice for log management, security, IT operations, and business analytics use cases.

3.2.2 LogStash

Logstash (LogStash, 2023) is a flexible and customizable open-source data processing tool that helps users to collect, process, and forward logs and other event data from various sources. Its ability for real-time processing allows users to obtain insights from data and respond to events quickly. Logstash has a rich collection of plugins that make it easy to integrate with other tools and platforms. Therefore, it can receive data from many different sources, such as files, Syslog, and beats, and can send its outputs to various ranges of destinations like Elasticsearch (ElasticSearch, 2023), Kibana (Kibana, 2023), and different data stores.

Data processing in Logstash mainly includes parsing, cleaning, filtering, and transforming data to a desired format. It does not support any data visualization or machine learning techniques by itself. But, thanks to its huge collection of plugins, it can integrate with other tools that have these capabilities and take the advantage of them for supporting more complex data analysis. For instance, Kibana provides many different machine-learning and data-mining algorithms that can be applied for pattern recognition and anomaly detection from data stored in large log files. Among all of the outstanding features of Logstash, the most salient one is that it is designed to be highly scalable which makes it possible to process a large volume of data efficiently.

3.2.3 NXLog

Another well-known log management tool that can be introduced here is NXLog (NXLog, 2023). This tool is efficient, flexible, lightweight, and easy to use. These features make it suitable for small to medium-sized businesses. NXLog is a multi-platform log collection tool. The meaning of multi-platform is that it is designed to collect log files from a wide range of sources and platforms, including Windows, Linux, and Unix systems, as well as network devices and applications. In fact, it can be said that this feature is the key feature and the biggest selling point of NXLog.

From the data processing point of view, it can be said that by supporting multithreaded processing, NXLog can process log data in parallel, which make it possible to handle a large amount of data efficiently and in real time. Data processing in this tool includes cleaning, parsing, extensive filtering, and data manipulation. NXLog does not support complex machine learning processes by itself, but by creating a wide range of output formats it is designed to support integration with many other log management tools and systems such as Elasticsearch, Graylog (GrayLog, 2023), and Splunk.

3.2.4 Amazon Cloudwatch Logs

Another powerful log management service is Amazon CloudWatch Logs (Amazon CloudWatch, 2023), which is provided by Amazon Web Services (AWS). CloudWatch Logs allows users to collect, analyze, monitor, and perform complex searches and queries. It provides data log visualization as well. CloudWatch Logs supports a wide range of log sources within the AWS environment, including Amazon EC2 instances, AWS Lambda functions, Amazon VPC flow logs, and custom applications running on Amazon Elastic Beanstalk. Moreover, by integrating with other AWS services, such as AWS CloudTrail and AWS Config, it can provide a comprehensive view of activity and resource usage within an AWS environment. With CloudWatch logs, users can obtain insights into the performance, availability, and security of their infrastructure and applications. They also can correlate related logs, troubleshoot issues, and improve operational efficiency.

Amazon CloudWatch Logs itself does not directly support machine learning or data mining analyses. However, as mentioned above, it is possible to export log data to other services for further analysis. For example, one can use Amazon Elasticsearch Service to index and search log data and use Amazon SageMaker to build and train machine learning models on log data, and then use these models to detect anomalies or predict future trends. In addition, it is possible to export log data from CloudWatch Logs to Amazon S3, which provides a highly scalable and durable object storage service. Once log data is in S3, a variety of third-party tools and services to perform data mining and machine learning analyses can be used.

3.2.5 AWStats

AWStats, which stands for Advanced Web Statistics, is another free and powerful log analyzer tool that is designed for parsing, analyzing, and visualizing data from internet services like web server logs. As input, AWStats supports most major web server log file formats including Apache, Nginx, WebStar, IIS, and many other common web log formats. It can be run through a web browser and the output, which includes descriptive statistics and comprehensive reports about website traffic, is created in an easy-to-read HTML format.

It can show you all possible information your log contains including the number of visitors, the pages they visited, the operating systems and web browsers they used, and the geographical location of visitors. These reports provide valuable insights into website performance, visitor behavior, and traffic sources. However, this log analyzer does not natively support machine learning techniques for further data analysis. Although it is possible to export data and use it as input for other data analysis tools, it yet is not very straightforward and may need additional preprocessing or cleaning of the data.

3.2.6 Microsoft Log Parser

Microsoft Log Parser is a powerful tool for cleaning, parsing, analyzing, and manipulating log files from various sources. It receives a text-based input file format, including CSV, XML, web server logs, event logs, Registry, the file system, and Active Directory. Users can perform complex search queries and analysis on input data and extract specific records based on selected conditions. The resulting dataset can then be saved in different formats, including CSV, XML, and SQL. Utilizing SQL queries is one of the key functionalities of Microsoft Log Parser, which allows users to easily extract and manipulate data from log files. Some of the most common uses of Log parsers are analyzing website traffic, monitoring server performance, and troubleshooting system issues.

Microsoft Log Parser includes neither built-in data visualization capabilities for graphically displaying results nor machine learning techniques for more complex data analysis. However, as mentioned earlier, it has various output formats including CSV, XML, and SQL that can be used by other tools and applications.

Table 11 provides an overview of several most widely used log analyzers, highlighting their distinguishing features. Furthermore, the table outlines the specific stages of the log-analyzing process that each tool covers.

Table 11: Comparing several well-known and popular log analyzer tools in terms of their abilities and their most important features.

	Normalization	Parsing	Filtering	ML methods	Visualization	Key Feature	Price
AWStats	✓	✓	✓	×	✓	Providing detailed and real-time reports on website traffic	Free and open source
Log Stash	✓	✓	✓	×	×	High-volume data processing and integration	Free and open source
Webalizer	✓	✓	✓	×	✓	Ease of use and simplicity and comprehensive website statistics reporting	Free and open source
NXLog	✓	✓	✓	×	×	Collect logs from a wide range of sources and platforms	Both a free and commercial version
Splunk	✓	✓	✓	✓	✓	Real-time (fast) processing of big data	commercial
Amazon Cloudwatch Logs	✓	✓	✓	×	✓	Seamless integration with other AWS services.	pay-as-you-go pricing
Logwatch	✓	✓	✓	×	×	Simplicity and ease of use for monitoring and analyzing system logs on Unix and Linux systems	Free and open source
GoAccess	✓	✓	✓	×	✓	Real-time web log analysis and extensive reporting capabilities.	Free and open source
Analog	✓	✓	✓	×	✓	High speed and efficiency in processing large log files	Free and open source
Microsoft Log Parser	✓	✓	✓	×	×	Powerful and flexible log file analysis using SQL-like queries	Free and open source
Graylog	✓	✓	✓	✓	✓	Centralized log management capabilities for the entire infrastructure	Both a free and commercial enterprise

3.3 Web Mining Research

Web mining involves utilizing data mining methods to automatically derive valuable insights from any web-related document or service (Singh & Singh, 2010). This extracted information can be utilized in various domains such as website design improvement, cyber-attack detection, summarization, indexing, etc. According to the type of information being processed, web mining can be divided into three main categories: web content mining, web structure mining, and web usage mining (Kandpal, Singh, & Shekhawat, 2019). Among these three categories, I will focus on web usage mining, as it usually uses information from log files and is closer to the topic studied in this dissertation.

The term web usage mining was first proposed by Cooley in 1997 and its purpose is to use data mining techniques to extract usage patterns, web structure, and user behavior by analyzing any data related to user interactions with web servers. This interaction data can be obtained from various sources or places, including server-side logs, proxy-side logs, browser logs, user profiles, registration data, cookies, user queries, bookmark data, mouse movements, and clickstreams. Analysis of this data is very essential for the development and improvement related to a website. Figure 2 illustrates the general framework of web usage mining (Malik & Rizvi, 2011).

The procedure of web usage mining is composed of four main steps: data collection, data pre-processing, knowledge discovery, and knowledge analysis. Given that the input data comes from a log file or a group of log files, data collection, and its associated challenges were discussed in Chapter I. In the next section, the next three steps and their corresponding techniques are described. Lastly, examples are provided of the primary applications of web usage mining.

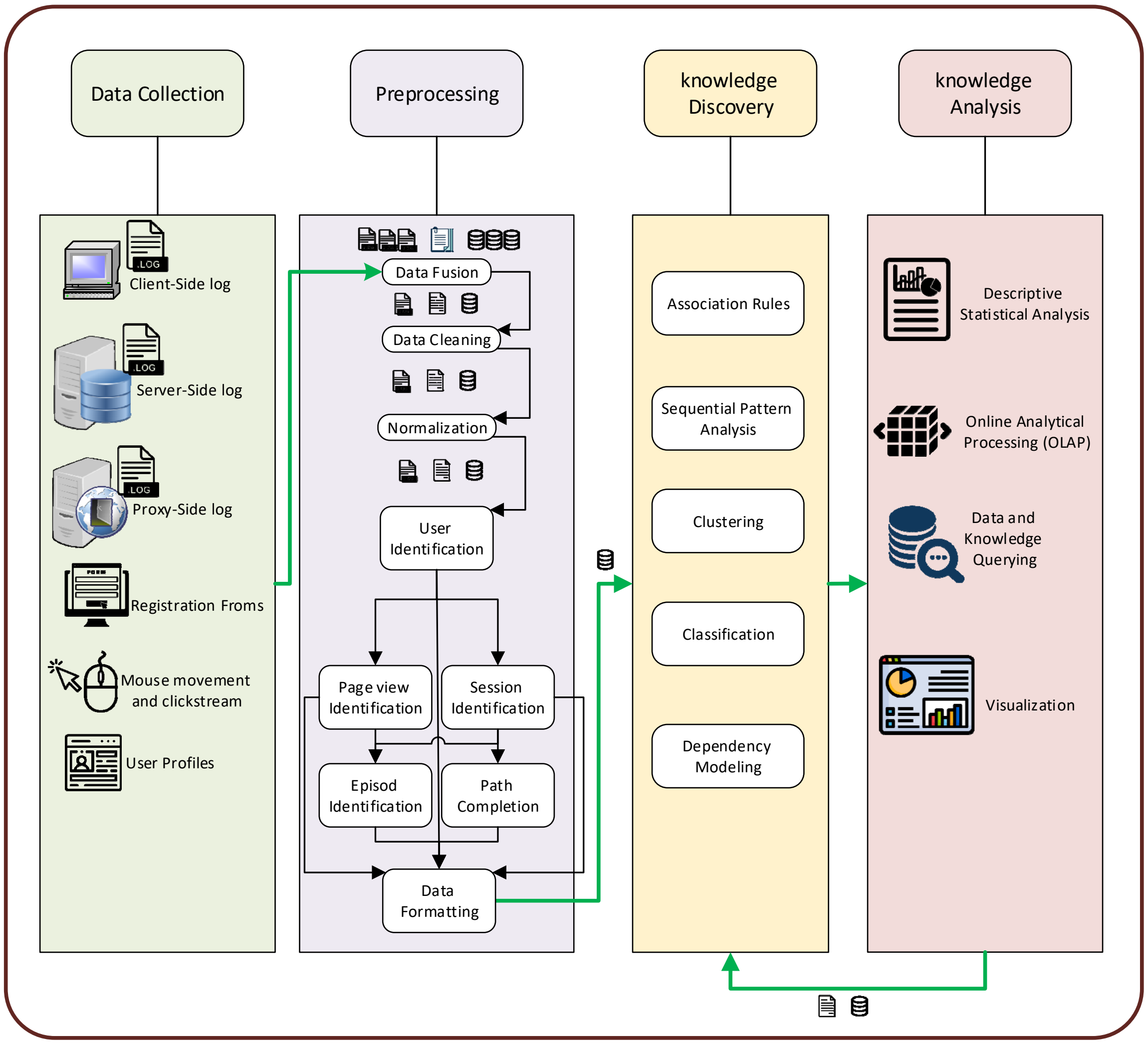


Figure 2: The general framework of web usage mining

3.3.1 Preprocessing

Real-world data can be noisy, incomplete, or inconsistent. Therefore, it is necessary to preprocess them and make them reliable and consistent before using them for further analysis. This makes the preprocessing phase an essential step in the web usage mining procedure. The main steps of preprocessing include data cleaning, normalization, identifying users, identifying sessions, and completing paths (Kandpal, Sinha, & Shekhawat, 2017). Several more steps can be added to this list based on the conditions of data collection and the purpose of the analysis.

Data Fusion: In large websites, it is common to reduce the server load by using multiple web or application servers with redundant content to serve users. However, this leads to the recording of log information in several server log files. Moreover, if client-side log files are used for web usage mining, each client will have a separate log file. In such scenarios, data fusion becomes necessary to combine these log files into a single file for further analysis. This requires more than just attaching input files. For instance, the files may come from different time zones, but they need to have the same time source in the merged file. Additionally, identifying sessions requires a heuristic method based on referrer information, along with other user or session identification methods (Mobasher, 2006). As illustrated in Figure 1, data fusion receives several log files, databases, or any other structured/unstructured files containing user interaction data as input and produces a single merged file or database as output.

Data Cleaning: Data cleaning involves the identification and removal of irrelevant or extraneous information from raw log data to improve the accuracy and speed of data analysis. The process involves several steps. First, extraneous references that are not executed or downloaded based on the user's request, such as entries for accessing style files, JPEGs, GIF files, Java Scripts, and other audio/video files, should be removed. Second, the HTTP status code should be considered for data reduction. For example, entries for resources that are not available on the web server are marked with error status codes. These records are not necessary for e-commerce applications and should be discarded. However, if the goal of web usage mining is web intrusion detection, all server error status codes are important. Third, entries resulting from crawlers or spiders need to be eliminated. It is common for a log file to contain a significant percentage of references resulting from a search engine or other crawlers or spiders, and these records do not reflect the way human visitors navigate the site. While crawlers' records can sometimes be detected easily by simple string matching, in some cases, various heuristic-based or classification methods are required to determine non-human behavior (Facca & Lanzi, 2003; Kandpal et al., 2017; V. Kumar & Thakur, 2017; Mobasher, 2006; Varnagar, Madhak, Kodinariya, & Rathod, 2013).

Normalization: Normalization is a technique used to organize data in a consistent and standardized way across all records and fields. By doing so, it can increase the coherence and consistency of entry types, leading to higher-quality data. For example, in a web server log file, normalization might involve changing the date and time format to a predefined and consistent format.

User Identification: While it's not mandatory to know the identity of users for web usage analysis purposes, it is essential to differentiate between different users who visit the website. One approach to achieve this is by utilizing client-side cookies. However, not all websites employ cookies, and privacy concerns sometimes prompt users to disable them. IP addresses also can be utilized to distinguish unique visitors, but they are not always sufficient for distinguishing between different users. For example, proxy servers can assign the same IP address to multiple clients or a single user can use various systems or devices to visit the website. To overcome this limitation, several heuristics have been proposed. Cooley et al. assume a new user if a web page is accessed directly without any hyperlink from the same IP address. This method is not foolproof either, as bookmarked page access can lead to inaccuracies in identifying unique users and misconceptions. Another assumption is using a combination of information about users' browsers, operating systems, and referrers to distinguish users, but even this method can be prone to errors because of caching at different levels and using various browsers/systems by users (Kandpal et al., 2017; V. Kumar & Thakur, 2017; Mobasher, 2006; Varnagar et al., 2013).

Page View Identification: In a static single-frame site, each HTML file corresponds to a single page view. However, multi-framed sites may require several files to form a page view. Dynamic sites also may need to combine static templates and a specific set of parameters to generate content and create a page view. Therefore, the process of page view identification depends on various factors such as the structure of the website, page contents, and domain knowledge. Page view identification can help identify specific user events through a collection of web objects or resources, or alternatively, at a higher level, a collection of pages. An example would be pages related to the same concept category such as product views on an e-commerce website (Mobasher, 2006).

Session Identification and Reconstruction: A user's session can be defined as a series of activities performed by a single user on a website from the time they enter until they leave, with the restriction that the time elapsed between two consecutive clicks should not exceed a certain period (V. Kumar & Thakur, 2017). After the user identification step, each user's click stream is divided into clusters This method of division is called Session Reconstruction or Sessionization (Kandpal et al., 2017). However, there are challenges in this process caused by proxy servers or browser caching. Proxy servers can associate a single IP address with multiple users, making it difficult to distinguish users and consequently, sessions. Browser caching can also result in information loss in log files when visitors use the back button, which makes it difficult to reconstruct the navigation path in the users' sessions (Facca & Lanzi, 2003). To identify sessions, two approaches have been proposed. The first approach involves using session IDs obtained from cookies, while the second approach involves applying various heuristics, such as the time interval between entries, the duration of time spent on the observed page, and the referrer (in the absence of a referrer, it is assumed to be the first page of a new session) (Varnagar et al., 2013).

Path Completion: Path completion is another important pre-processing step that is usually performed after sessionization and can be critical for accurate analysis. Due to browser or proxy-side caching, server access logs may not capture all access references for pages or objects that have been cached, resulting in incomplete user paths. Path completion is the process of identifying and filling in missing page sequences in the web server logs to ensure complete user path information for subsequent knowledge discovery. This technique relies on knowledge of site topology and referrer information from server logs and can be applied to achieve the goal of accurate analysis (Mobasher, 2006; Varnagar et al., 2013).

Episode identification: After performing page view identification, episode identification can be carried out to recognize relevant subsets of page views within each user session. An episode refers to a sequence of page views that are functionally or semantically related. To accomplish this, page views may need to be classified or clustered automatically or semi-automatically based on their functional types, domain ontology, or concept hierarchy. In cases where the website is highly dynamic, it may also be necessary to group page views within each session by considering parameters passed to script or database queries (Mobasher, 2006).

Data Formatting: This step represents the final stage of the preprocessing phase. Once the earlier phases have been executed, the data is correctly formatted before being processed with mining techniques. To improve support for log querying aimed at frequent pattern mining, data extracted from web server logs is stored in a relational database using a click fact schema (Facca & Lanzi, 2003).

3.3.2 Knowledge Discovery

Knowledge discovery is a critical step in web usage mining as it enables the identification of useful patterns and the extraction of valuable knowledge from web data. There is a wide range of statistical and data mining techniques available from different research fields such as data mining, machine learning, statistical methods, and pattern recognition that can be employed for knowledge discovery. Some frequently used techniques include sequential analysis, clustering, classification, association rule mining, dependency modeling, and time series analysis (V. Kumar & Thakur, 2017; Varnagar et al., 2013).

3.3.2.1 Association Rules

Association Rules represent one of the most widely used data mining techniques in web usage mining. They are used to discover relationships among web pages that are frequently referenced together during a single server session, even if these pages are not directly linked to one another via hyperlinks. This technique is helpful for generating frequent patterns and rules in website usage patterns. The outcome of association rules is often in the form of "A.html, B.html ⇒ C.html", indicating that if a user accessed both pages A.html and B.html, it is highly likely that they also accessed page C.html in the same session (Facca & Lanzi, 2003). This technique has applications in business, marketing, and website redesign. The presence or absence of association rules also can be used as a heuristic for prefetching documents to reduce user-perceived latency when loading pages from remote sites (Srivastava, Cooley, Deshpande, & Tan, 2000). There are several algorithms available for performing association rule mining, including Apriori (Agrawal, Srikant, & others, 1994), Eclat (Zaki, Parthasarathy, Ogihara, Li, & others, 1997), FP-Growth (Han, Pei, & Yin, 2000b), and Frequent Pattern tree (Han, Pei, & Yin, 2000a).

3.3.2.2 Sequential Pattern Analysis

Sequential pattern analysis is similar to association rules in detecting the co-occurrence of events but unlike association rules, they capture the temporal relationship between events as well. In web usage mining, sequential patterns are commonly used to discover maximal frequent navigation patterns among all interactions that occur during users' sessions. For instance, the order in which pages A and B are accessed or the fact that if users visit pages A and then B, accessing page C would be highly probable can be captured in the discovered patterns (Facca & Lanzi, 2003). The identification of such patterns has many applications. For example, they can aid web marketers in predicting future visit patterns and improving their ability to target specific user groups with relevant advertisements (Srivastava et al., 2000).

The algorithms used for sequential pattern extraction are either based on association rule mining, tree structures, data projection techniques, or Markov chains. The most well-known examples of algorithms include Apriori All, GSP, PSP+, FreeSpan, Prefix Span, SPADE, Markov model, and Spam (Facca & Lanzi, 2003; Kandpal et al., 2017). After identifying sequential patterns, other temporal analyses such as trend analysis, change point detection, and similarity analysis can also be applied for further analysis (Srivastava et al., 2000). As an example Rojas & Nasraoui (2007) have introduced a method for pattern discovery in stream environments by constructing a prefix tree with a dynamic attribute ranking. They applied their method to summarize evolving data streams of transactional data.

3.3.2.3 Clustering

Clustering is a technique used in unsupervised machine learning to group similar objects together based on a distance function that computes their similarity. In the web usage mining domain, there are different types of clusters that can be discovered, including user clusters, page clusters, and session clusters. User clustering is useful for identifying groups of users who exhibit similar behavior and access patterns within the system. This knowledge can be applied in e-commerce applications to infer user demographics or to provide personalized web content to users who share similar interests. Clustering can also be used to dynamically generate web page links for each user based on the traversal pattern of other users who are assigned to the same cluster (Yan, Jacobsen, Garcia-Molina, & Dayal, 1996).

Pages or items categorization can be conducted either based on usage data, such as user sessions, or based on content features such as keywords. Content-based clustering can result in groups of pages or products that are related to the same topic or category, which is useful for internet search engines and web assistance providers. Usage-based clustering automatically organizes items that are frequently accessed together into groups (Mobasher, 2006). Olfa Nasraoui, Krishnapuram, Joshi, & Kamdar (2002) introduced two robust fuzzy relational techniques designed for clustering web user sessions. These techniques offer a solution to effectively address the inherently noisy and fuzzy nature of web usage data.

Furthermore, clustering can be applied to detect anomalies in the data. After partitioning the data into clusters, there may be cases that do not fit well into any clusters, and these cases can be considered anomalies (Kandpal et al., 2017).

3.3.2.4 Classification

Classification is a technique in data mining where a data item is categorized into one of several pre-defined classes through supervised learning. In the context of web usage mining, this technique can be used to automatically assign a class label to a user based on their browsing history or other attributes. This requires selecting the most appropriate features to describe a given category. For instance, the classification of server logs can help identify a user's role or experience within a system or classify users according to their likelihood of purchasing a product, considering factors such as demographic attributes and navigation patterns. Another application of classification in the web usage mining domain is predicting the next web page a user is likely to visit based on their previous browsing behavior such as the pages they have visited, the time spent on each page, and the sequence of pages visited. Additionally, previously discovered clusters and association rules can also be used to classify new users. There are many well-known algorithms that can be used for classification, such as decision tree classifiers, naive Bayesian classifiers, k-nearest neighbor classifiers, and Support Vector Machines (Mobasher, 2006; Srivastava et al., 2000).

3.3.2.5 Dependency Modeling

Dependency modeling is a valuable task in web mining that involves creating a model that can represent significant relationships among various variables in the web domain. For instance, one may be interested in developing a model that can depict the different stages a visitor goes through while shopping online, based on their actions, from being a casual visitor to a potential buyer. To achieve this, probabilistic learning techniques such as Hidden Markov Models and Bayesian Belief Networks can be employed to model users' browsing behavior. Creating a model of web usage patterns can not only provide a framework for analyzing user behavior but can also be useful in predicting future web resource consumption. This information can be used to develop strategies that increase product sales on a website or improve the user experience (Srivastava et al., 2000).

3.3.3 Knowledge Analysis

The outcome of the knowledge discovery stage may not be presented in a way that allows for comprehension or drawing conclusions. Therefore, as demonstrated in Figure 1, knowledge analysis comes into play as the final phase in web usage mining. This step involves sifting through the results of the previous stage to identify interesting rules, patterns, or statistics while disregarding irrelevant ones. The approach used to analyze patterns is typically determined by the purpose of web mining. Descriptive statistical analysis, knowledge query mechanism, Online Analytical Processing Technique (OLAP), and Visualization Technique are the most commonly used methods for knowledge analysis (Varnagar et al., 2013).

The most widely used approach to gain insights is through statistical methods. By examining the session file, various descriptive statistical analyses can be performed on variables like page views, viewing time, and navigational path length. Several web traffic analysis tools create periodic reports that contain statistical information such as frequently accessed pages, average page view time, or average path length through the website. While these reports may have limited low-level error analysis, like detecting unauthorized entry points or finding common invalid URIs, they still have the potential to be valuable in improving system performance, enhancing system security, simplifying site modification tasks, and offering assistance in making marketing decisions (Srivastava et al., 2000).

Another commonly used method for knowledge analysis is the knowledge query mechanism such as SQL. This technique allows the analyzer to retrieve information by the analyzer in a managed manner. For instance, SQL can be applied to identify abnormal user behavior or to pinpoint the most common error experienced by international users (Kandpal et al., 2017).

OLAP, which is a robust paradigm for strategic analysis of relational databases, involves loading usage data into a data cube to perform various operations like roll-up, drill-down, or slice. Typical applications of OLAP include Customer Relationship Management (CRM), marketing and inventory analysis, business reporting, management reporting, budgeting and forecasting, sales analysis, and financial reporting (Kandpal et al., 2017).

The Visualization Technique is a method that employs various tools to transform information into knowledge and comprehend the behavior of web users. It includes using 2D and 3D pictorials, tables, charts, graphs, or any other visual presentations such as assigning colors to different values. These techniques can be beneficial in illuminating overall patterns or trends in the data (Varnagar et al., 2013).

Finally, the outcome of the knowledge analysis phase can be saved in various formats, such as HTML, CSV, or PDF. Some tools even offer the option to configure the target email address within the tool, enabling the automatic emailing of the analysis report at a predetermined scheduled time (Srivastava et al., 2000).

3.3.4 Applications of Web Usage Mining

Web usage mining has numerous applications across various domains. In the web design domain, it can be utilized for web personalization. Web log mining can learn about customer preferences and product associations based on users' profiles and their usage behavior. Some examples of applications of web personalization include individualized marketing for e-commerce and providing dynamic recommendations.

Web usage mining also finds application in the website optimization domain. It can help in developing appropriate prefetching and caching strategies that can effectively deliver web content and reduce server response time (Facca & Lanzi, 2003). Furthermore, by analyzing user behavior, it can identify areas of the site that need improvement, such as slow loading pages, high bounce rates, and ineffective navigation.

Web log mining techniques, such as clustering and classification methods, can be applied in the security domain for online crime investigation. For instance, internet fraud, hacking, virus spreading, child pornography distribution, web attacks, and cyber terrorism can be investigated using these methods. Crime patterns can be identified and network visualization can be done through techniques such as neural networks, decision trees, genetic algorithms, and support vector machines (Kandpal et al., 2019).

Web usage mining can also be applied in the e-commerce and marketing domain for mining business intelligence from web usage data and transaction analysis. It can provide an effective advantage to Customer Relationship Management (CRM) through the use of web usage mining techniques. Additionally, web log mining can improve web advertising by determining the target pages and users for each advertisement based on user groups and their usage patterns (Kandpal et al., 2017).

3.4 Comparative Analysis of Previous Works with Current Study

Log file analysis, web usage mining, and the research area of this study all involve the extraction of insights from machine-generated data. However, there are notable differences between them. This section provides a comparison of these methods from three distinct perspectives: their data sources, their purpose, and the applications they serve.

3.4.1 Data Source

Web usage mining involves analyzing various forms of data related to user interactions with web servers, including different types of weblogs (such as browser logs, proxy logs, and server logs), as well as user profiles, form registrations, click streams, keystrokes, and social media activity. Log file analysis, on the other hand, typically involves analyzing any type of web or non-web log file that records any user interaction with a system. This may include system logs, error logs, application logs, event logs, debug logs, security logs, audit logs, performance logs, and log files generated by sensor systems. While automatic usability evaluation using log files may utilize both application logs and web logs, this dissertation narrows its focus to server logs to keep the scope of the study feasible for a doctoral thesis. Figure 3 illustrates the similarities and differences between these fields of study in terms of their data sources.

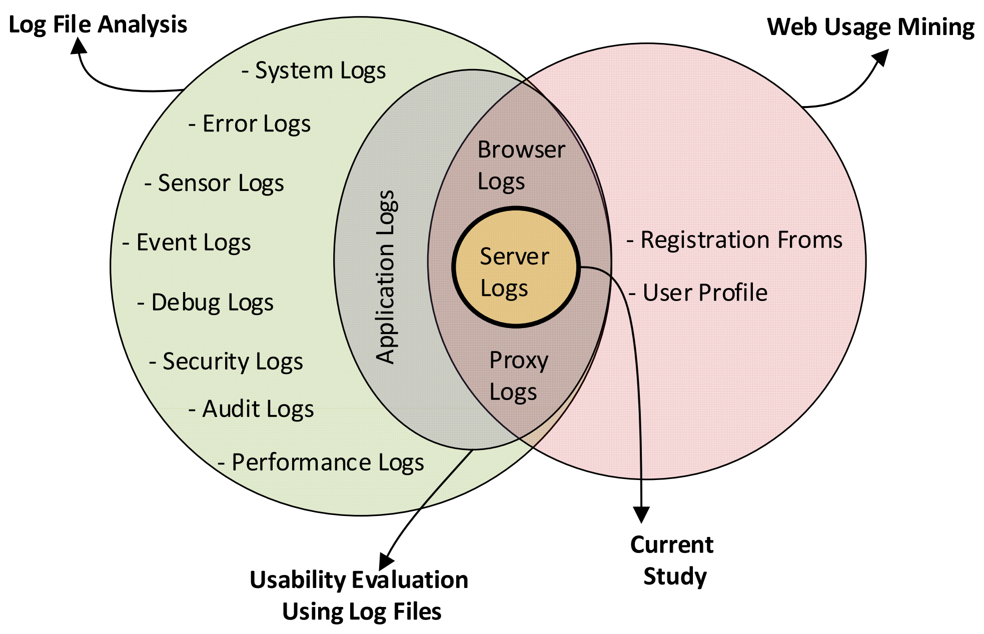


Figure 3: Comparison of Log file analysis, web usage mining, and current study in terms of their data source

Table 12, also provides a more general overview of the types of data sources associated with each method.

Table 12: Types of data sources can be used in Log file analysis, web usage mining, and current study

	Web Log Data	Other types of Log File	Non-Log Data
Log File Analysis	🟨	🟨	
Web Usage Mining	🟩		🟩
Usability Evaluation Using Log File	🟦	🟦	
Current Study	🟪		

3.4.2 Purpose

The primary purpose of log file analysis typically is tracking and examining various types of log files. Its main focus is often on preprocessing log data to make it ready for further analysis. Most log file analysis tools also involve knowledge analysis techniques such as visualization, knowledge query mechanism, OLAP, or descriptive statistical analysis. But as indicated in Table 11, only a few of them employ machine learning techniques to support knowledge discovery.

In contrast, web usage mining is mostly focused on the third step of its procedure (knowledge discovery). Although preprocessing is a necessary initial step for web log mining, most of the innovations and research topics in this field revolve around extracting knowledge and patterns from log data.

In this research, however, the main goal is to apply machine learning methods to extract knowledge from log files and use it to estimate usability metrics and attributes. While the proposed approach of this dissertation can be extended to different types of log files, it focuses on web log data, particularly server logs. Therefore, this research uses methods similar to those proposed in web mining knowledge discovery in section 3.3.2, but it employs them for a different purpose than previous applications. This study, in its knowledge discovery step, will specify which method to use and how to calculate each usability metric. In the next step, a numerical modeling approach will be presented for quantifying each usability attribute based on the value of extracted metrics in the previous step. In fact, this numerical model will be used instead of the previous knowledge analysis methods used in web mining frameworks (Figure 2), which include visualization, statistical analysis, OLAP, or SQL query methods. In Figure 4, a heat map illustrates the contributions of each domain (log analysis, web usage mining, and usability evaluation using log files) in the three primary stages of the process: preprocessing, knowledge discovery, and knowledge analysis.

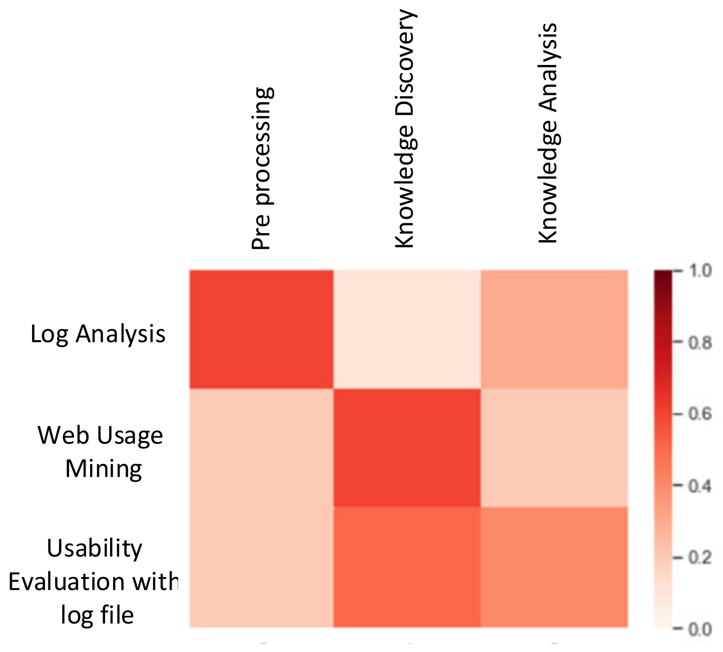


Figure 4: The heat map of the contribution of each domain (log analysis, web usage mining and usability evaluation using log file) in the three primary stages of the process including preprocessing, knowledge discovery, and knowledge analysis

3.4.3 Applications

The application of log analysis is relevant to the source that generates the log. For instance, analyzing the log file of an application can identify and improve its error-prone areas. Similarly, analyzing the log data from video sensors can help assess security or traffic flow in a given environment. System log file analysis can aid in monitoring, securing, and improving its performance. Likewise, web log file analysis can provide various reports such as determining peak traffic hours, bounce rate, session duration, etc., to enhance website performance, attract audiences, and strengthen website security.

Web usage mining is used in a variety of web domain pertinent applications, including security monitoring (such as fraud detection), web personalization, e-commerce (such as targeted advertising and enhancing marketing strategies), and web improvement (such as prefetching and cashing or web design modification).

Usability evaluation is a significant application within both log analysis and web usage mining domains. Although none of the previously mentioned applications in these domains are specifically tailored to usability evaluation, it remains a crucial aspect of log analysis that needs further attention and study. Figure 5 indicates the domain applications of log file analysis, web usage mining, and current study to illustrate their relationships, similarities, and differences.

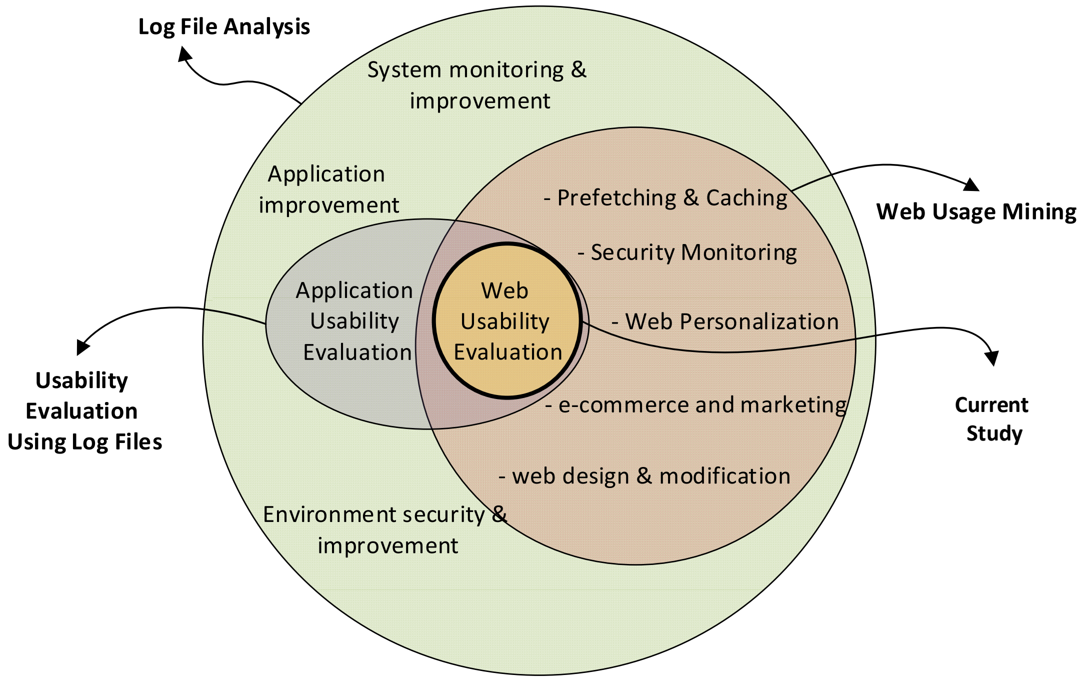

Log File Analysis
System monitoring & improvement
Application improvement
- Prefetching & Caching
Web Usage Mining
- Security Monitoring
Application Usability Evaluation
Web Usability Evaluation
- Web Personalization
Current Study
Usability Evaluation Using Log Files
- e-commerce and marketing
- web design & modification
Environment security & improvement

Figure 5: Comparison of Log file analysis, web usage mining, and current study in terms of their applications

CHAPTER 4

DATA COLLECTION

4.1 Introduction

The aim of this chapter is first to provide a comprehensive overview of the methods used for data collection in previous studies. This will include the different types of data sources that have been used, such as server-side, proxy-side, or client-side, as well as the strengths and limitations of each data type. Moreover, the effort that needs to be taken into account when choosing a particular type of log file is identified. Next, the process of synthesizing data will be examined and the techniques employed for simulating server log data will be discussed in detail. Finally, I will examine the characteristics and sources of real log data that will be used in the subsequent steps of this dissertation.

4.2 Data Collection in the Previous Works

This section reviews the previous methods from the perspective of their log files and data collection methods. In the following, I examine what type of log file each method used for their usability evaluation, what data they extracted from log files, and what restrictions the type of log file imposed on the proposed method. Table 13 shows some of the features of the data collection step for the 13 methods studied. The first column of the table specifies whether a separate file is used to store the transaction data of each user or the data of all users and all sessions are stored in a single file. The second column shows the tools used to generate the log file. Out of 13 works reviewed, seven have produced their own logger, 6 of which are client-side loggers. Column three shows the type of log file (Client Side, Proxy Side, and Server Side), and the last column determines whether the method was used for web or non-web applications.

Table 13: Data collection method in previous work

Evaluation tool	Single file per user/ One file for all users	Language/ Tools	Server/Client/Proxy Side	Web/NonWeb
(Inversini et al., 2011)	One file for all users	Web Server	Server Side	Web
UX-LOG (T. C. Menezes & Nonnecke, 2014)	One file for all users	Web Server	Server Side	Web
(Jorritsma et al., 2016)	Single file per user	Create by authors	Client Side	Non Web
(Babaian et al., 2007)	Single file per user	SQL	Client Side	Non Web
(Bader & Pagano, 2013)	One file for all users	Create by authors	Client Side	Non Web
(Siochi & Hix, 1991)	Single file per user	Create by authors	Client Side	Non Web
WebHint (Vargas et al., 2010)	One file for all users	USAProxy	Client Side	Web
WebQuilt (Hong & Landay, 2001)	Single file per user	WebQuiltProxy Created by Java servlet technology	Proxy Side	Web
GUITESTER (Okada & Asahi, 1999)	Single file per user	Create by authors	Client Side	Non Web
WAUTER (Balbo et al., 2005)	Single file per user	WIMM (XML file)	Proxy Side	Web
WELFIT (V. F. de Santana & Baranauskas, 2015)	Single file per user	WELFIT	Client Side	Web
AWUSA (Tiedtke et al., 2002)	One file for all users	Web Server	Server Side	Web
WebRemUSINE (Paganelli & Paterno, 2002)	Single file per user	implemented in Javascript	Client Side	Web

Table 14 specifies a detailed overview of the information extracted by each of the 13 reviewed methods from the log file. The first column, Page, indicates the name or link of a web page in case of web applications, or the title of a window in non-web applications. The second column, User/IP, displays the username or IP address of the person who performed the transaction, and the third column, Time, displays the timestamp of the transaction (including date and time). The fourth column, Page Item, specifies the name of the items on which the user performed an operation such as buttons, drop-down lists, radio buttons, etc. The fifth and sixth columns, Mouse and Keyboard, indicate the mouse-related (e,g., mouse movement, right or left click, and dragging) and keyboard events performed by the user, respectively. The seventh column, Desktop, provides information about the number of opened windows, frames, and active windows. The last column (Other) is positive only for WebQuilt and includes additional information collected in this method such as the transaction ID (TID), From TID, To TID, Parent ID, Frame ID, Link ID, HTTP Response, HTTP Method, and URL+ Query. From all possible mouse events, WebQuilt only captures clicks on the back and forward buttons on the browser.

Table 14: Type of information extracted from log file

Evaluation tool	Page	User/IP	Time	PageItem	Mouse	Keyboard	Desktop	Other
(Inversini et al., 2011)	y	y	n	n	n	n	n	n
UX-LOG (T. C. Menezes & Nonnecke, 2014)	y	y	y	n	n	n	n	n
(Jorritsma et al., 2016)	y	y	n	y	y	n	n	n
(Babaian et al., 2007)	y	y	y	y	y	y	n	n
(Bader & Pagano, 2013)	y	y	y	n	n	n	n	n
(Siochi & Hix, 1991)	y	y	n	n	n	n	n	n
WebHint (Vargas et al., 2010)	y	y	n	y	y	y	n	n
WebQuilt (Hong & Landay, 2001)	y	y	y	n	Back/Forward	n	n	y
GUITESTER (Okada & Asahi, 1999)	y	y	y	y	y	y	y	n
WAUTER (Balbo et al., 2005)	y	y	n	y	y	y	n	n
WELFIT (V. F. de Santana & Baranauskas, 2015)	y	y	y	y	y	y	n	n
AWUSA (Tiedtke et al., 2002)	y	y	n	n	n	n	n	n
WebRemUSINE (Paganelli & Paterno, 2002)	y	y	y	y	y	y	n	n

Figure 6 shows how different kinds of log files have used various types of information. As we can see, the mouse, keyboard, and page items are employed only by proxy and client which is obvious because they are not available in server logs. Moreover, desktop information is used only in one client logging method.

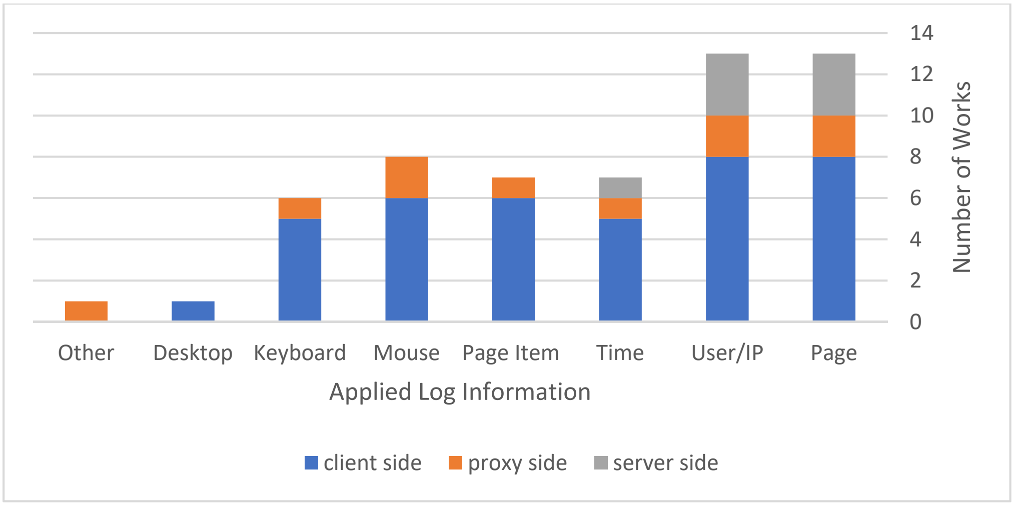


Figure 6: Representation of using log information in different types of log files (client side, server side, and proxy side)

Finally, Table 15 shows the challenges and limitations of data collection in each of the methods studied. Of the thirteen, seven have changed the code or recreated the evaluated software, of which six use the client-side log files. The only client-side method that does not need to change or rebuild a code is GUITESTER, which sends an independent logger application for the user; the user should run this application on her/his system.

Table 15: evaluators and users' efforts in collecting data

Evaluation tool	Evaluators effort	Users effort	Source
(Inversini et al., 2011)	Insertion of code into each Web page element to log abstract events and virtual page-views	None	Server Side
UX-LOG (T. C. Menezes & Nonnecke, 2014)	None	None	Server Side
(Jorritsma et al., 2016)	Recreate a new version of the app to create the log	None	Client Side
(Babaian et al., 2007)	Recreate a new version of the app to create the log	None	Client Side
(Bader & Pagano, 2013)	Embed logging code in the application	Use a modified version of the application	Client Side
(Siochi & Hix, 1991)	Recreate a new version of the app to create the log	None	Client Side

WebHint (Vargas et al., 2010)	Invitation for users to access the proxy every session	Access the proxy	Client Side
WebQuilt (Hong & Landay, 2001)	Invitation for users to access the proxy every session	Access the proxy	Proxy Side
GUITESTER (Okada & Asahi, 1999)	Invitation for users to access the proxy every session	Access the proxy	Proxy Side
WAUTER (Balbo et al., 2005)	Send logger application for users	Run logger application on his/her system	Client Side
WELFIT (V. F. de Santana & Baranauskas, 2015)	Invitation for users to access the proxy every session	Access the proxy	Proxy Side
AWUSA (Tiedtke et al., 2002)	Insertion of code into each Web page	Accept the invitation once	Client Side
WebRemUSINE (Paganelli & Paterno, 2002)	None	None	Server Side
(Inversini et al., 2011)	Task model definition and Insertion of code into each Web page element to log events	Indication of the task being performed via one modality	Client Side

4.3 Synthetic Data Generation

Due to the challenging nature of obtaining real log files, the use of simulation data is quite important. In particular, simulating log files for websites with specific usability features can serve two main purposes. First, it can be used to validate the results obtained from the proposed automatic usability evaluation method. Second, it enables us to repeat tests using different log files, leading to an improvement in the model's accuracy. By simulating various scenarios, it is possible to generate synthetic data that reflects the complexities of real-world usage data, allowing us to evaluate and refine the proposed approach with greater confidence. Therefore, due to the critical importance of simulation data in this study, a log file simulation model that accounts for various usability features is presented in this section.

The present study employs a technique based on probabilistic graphical models, with a particular emphasis on Bayesian networks. Bayesian networks are named after the mathematician and statistician Thomas Bayes, whose work on conditional probability made the foundation for Bayesian statistics (Koller & Friedman, 2009). A Bayesian network is a graphical model that operates on probabilities. It represents a collection of variables and their conditional relationships using a directed acyclic graph (DAG). In these models, each variable is represented as a node in the graph, and the relationships between variables are represented as directed edges between the nodes. The edges indicate the conditional dependencies between the variables, where a variable is dependent on its parent nodes in the graph and the full joint distribution of all variables in the model can be obtained by $P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i | parent(X_i))$ (Koller & Friedman, 2009).

There are several reasons that make Bayesian networks a suitable tool for modeling usability features. First, in log file analysis, we don't have much data to train the model, and preparing a dataset is a time-consuming task. Therefore, we cannot rely on machine learning techniques that rely on large amounts of data in their training phase such as Neural Networks. But instead, in this domain, we have expert knowledge and heuristics which are critical, and we can apply them in building the model. Bayesian networks can be used to incorporate expert knowledge into the model, allowing domain experts to provide input and guidance on the structure and parameters of the model. This can lead to more accurate and effective models. However, it is still possible to use data in these models for learning. Therefore, whenever new information is available, it can be used to improve the model and adopt it with new data over time. Second, Bayesian networks are graphical models that explicitly represent the relationships between variables, making it easy to interpret the model and understand how different variables influence each other. Considering that our work applies usability factors and creates a generalized model that is simplified based on certain assumptions, this aspect holds significant importance. This feature improves the reader's comprehensive understanding of the model, enabling them to optimize and refine it for specific purposes or domains in the future. Therefore, this understanding becomes a valuable foundation for further enhancements and advancements in the field.

Third, Bayesian networks provide a natural framework for effective reasoning under conditions of uncertainty and making probabilistic predictions. This characteristic proves to be particularly useful when dealing with usability attributes and human decision-making processes, as both domains often exhibit inherent noise and uncertainties.

4.3.1 Defining a Bayesian Network Model for Data Generation

In order to generate data using a graphical Bayesian model, the first step is the identification of model variables and their dependencies. To this aim, one should determine how each usability feature impacts user behavior and consequently, user transactions with the system. Thus, it is required to incorporate in the model, not only the variables that capture the user's behavior, but also the variables that represent the website's usability characteristics and their influence on the user's actions within the system. The variables that determine the user's behavior in the system are those variables that can be used to simulate the log file transactions.

During the task completion process, users may access multiple pages with varying parameters, with each page requiring one or more actions to be performed. The website's usability features can affect the speed of completing each action, the frequency of errors, and the type of errors made by users. For instance, overwhelming a user with an excessive amount of information may increase the time spent on a page, and unclear titles and labels may lead to incorrect path selection. My analysis of user behavior has identified four distinct categories of errors: improper usage of the back browser button, reloading the same page due to an error or manually by user request, leaving the task incomplete, and selecting an incorrect path. Four variables of 'Back', 'Loop on the Same Page', 'Incomplete', and 'Wrong Path' are added to the model respectively for each category. Although the selection of an incorrect path has two possible outcomes, either the user may halt task completion after one or more steps along the wrong path or revert to the correct path to resume the task, for the sake of simplicity, only one variable will be included in the model. However, in the implementation phase, both possible modes will be randomly added to the log file to comprehensively capture all user behaviors.

To ensure an accurate representation of user behavior within the system, the model requires four additional variables. The first variable, "Time," reflects the impact of usability features on the speed of task completion. The second variable, "Correct Path," measures the impact of usability features on the user's ability to select the correct path without any error. The third variable, "Learning", measures the level of ease or difficulty involved in learning the task at hand, based on the features of the website. Finally, the fourth variable, "Remembering", reflects the positive and negative impacts of website features on the user's ability to recall how to perform a task after a certain period of inactivity.

If a user is required to request pages A, B, C, D, E, and F from the server to complete a task correctly, the occurrence of any of the aforementioned error categories can impact the user's journey through the task. For example, if the user makes an error in performing an action on page B, they may need to reload the same page from the server, resulting in a request for page B again.

Table 16 illustrates an example of the requested page list in the occurrence of each defined error category.

Table 16: Sample of the requested page list in the occurrence of each type of error

Correct Path	A→B→C→D→E→F
Loop on the Same Page	A→B→C→D→E→E→E→F
Wrong Path	A→B→C→G→H A→B→C→G→H→D→E→F
Incomplete	A→B→C
Back	A→B→C→D→C→D→E→F

As mentioned before, the objective is to generate a log file based on website usability features. Therefore, to determine the second group of variables, a thorough examination of various usability models including ISO Standard (Organizacion Internacional de Normatizacion - ISO, 2018), Scapin & Bastien's model (1997) Shackel's model (Shackel, 2009), and Nielsen's model (Nielsen, 1994b) was conducted. Following this analysis, I found Scapin & Bastien's model (1997) the most appropriate for the intended purpose. Scapin & Bastien proposed a set of usability dimensions (ergonomic criteria), based on available experimental results and large sets of individual guidelines. This set consists of eight main criteria (Figure 7), some of which are divided into sub-criteria (Figure 8). To assess the reliability of this set, the authors asked a set of participants (human factors specialists and non-specialists) to identify which criterion was violated for each usability problem. Results show a one-to-one matching between usability problems and criteria (i.e., all problems were diagnosed under one particular criterion), which supports the independence of the criteria.

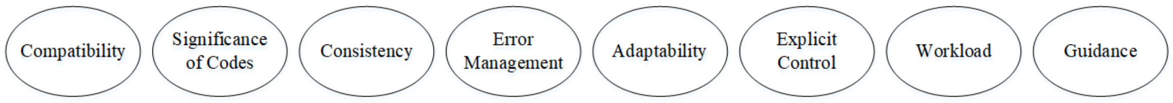


Figure 7: Eight main ergonomic criteria proposed by Scapin & Bastien (1997)

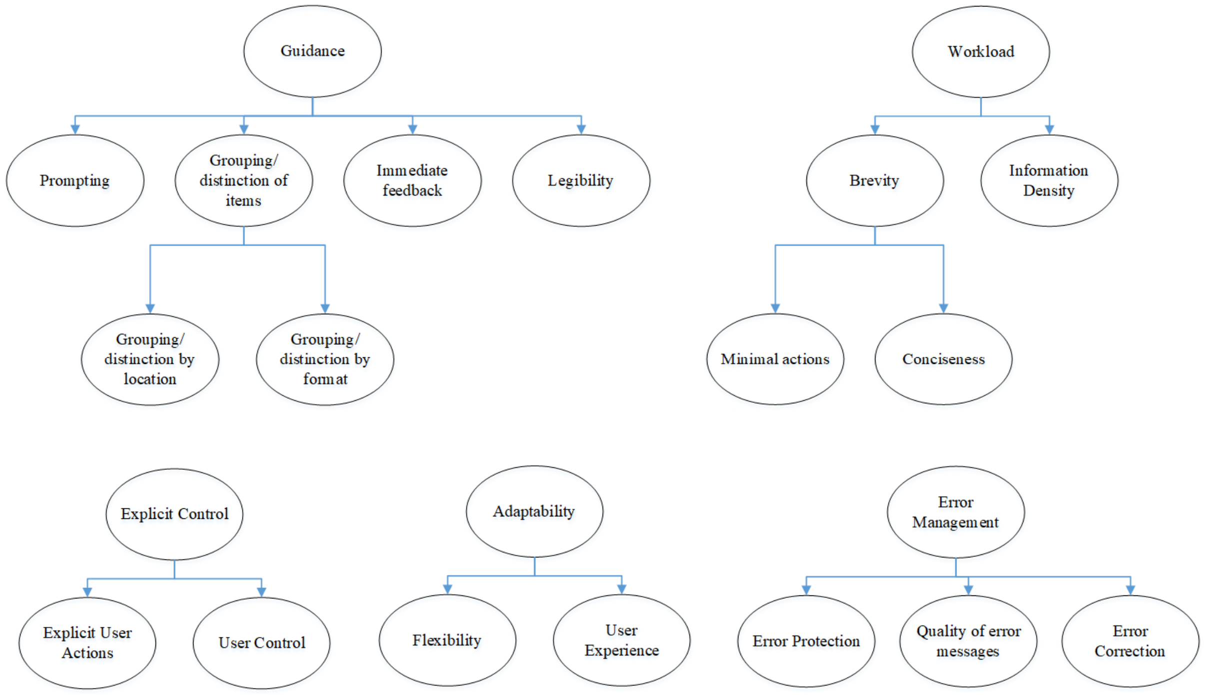


Figure 8: The sub-criteria of each of the main criteria in Figure 7. Three criteria including Consistency, Significance of Codes, and compatibility do not have any subdivision

The primary motivation for developing the proposed Bayesian model based on Scapin & Bastien's model is their unique approach to defining independent sets of usability criteria and determining their hierarchical sub-criteria. As the Bayesian model establishes conditional relationships among variables, using a set of independent groups and knowing the relationships within each group can significantly simplify and enhance the model's comprehensibility. Therefore, I will go through each main criterion and its sub-criteria to find their relationship and their influence on predetermined variables including Time, Back, Loop on the Same Page, Incomplete, Wrong Path, Correct Path, Learning, and Remembering.

Guidance

The first main criterion defined in this model is 'Guidance', which pertains to informing, advising, orienting, and instructing users through various means like messages, labels, and alarms. Effective guidance enhances learning and system utilization, resulting in fewer errors. As shown in Figure 8 it has 4 sub-criteria including 'Grouping', 'Prompting', 'Immediate Feedback', and 'Legibility'.

The criterion 'Grouping' concerns the distinction of items based on their visual organization in relation to one another. It involves considering the location and graphical characteristics of the items to indicate their relationships and differences. There are two subcategories of 'Grouping': criterion 'Grouping by Location', which involves the relative positioning of items, and criterion 'Grouping by Format', which involves graphical features like color and font to indicate class membership or differences (Scapin & Bastien, 1997).

To determine the grouping value, four key factors come into play: G1 (Differences Between Classes), G2 (Positioning of items within a class), G3 (Distinction Between Classes), and G4 (Resemblance within a class). Eq. 1 to Eq. 4 formulates each factor.

$$G1 = \frac{number\ of\ items\ with\ significant\ relative\ distance\ from\ other\ classes}{total\ number\ of\ items} \times 100 \qquad \textit{Eq. 1}$$

$$G2 = \frac{number\ of\ items\ with\ close\ relative\ distance\ within\ the\ same\ class}{total\ number\ of\ items} \times 100 \qquad \textit{Eq. 2}$$

$$G3 = \frac{number\ of\ items\ with\ distinct\ graphical\ features\ from\ other\ classes}{total\ number\ of\ items} \times 100 \qquad \textit{Eq. 3}$$

$$G4 = \frac{number\ of\ items\ with\ similar\ graphical\ features\ within\ the\ same\ class}{total\ number\ of\ items} \times 100 \qquad \textit{Eq. 4}$$

Having G1 to G4, Grouping can be obtained by a weighted average of these factors (Eq. 5)

$$Grouping = \sum_{i=1}^{4} \omega_i \times G_i \qquad \textit{Eq. 5}$$

When items are presented in a well-organized and formatted manner that indicates the similarities or differences between them and their groups, it improves the users' understanding of a display screen and helps them not only learn their relationship(s) better but also remember them later more easily. This resulted in adding two edges to the model (Grouping → Learning, Grouping→ Remembering). Effective grouping also helps the users to find faster what they are looking for and therefore decreases the time of achieving their goal (Grouping → Time).

According to Scapin and Bastien's (1997) model, 'Prompting' refers to the tools used to steer users toward certain actions, such as entering data or completing other tasks. This criterion also encompasses any methods that assist users in understanding their options when multiple actions are available, and it involves providing status updates about the system's current state, as well as information about the availability and accessibility of help resources. Because effective prompting can provide guidance and help users to learn the system better and save users from having to memorize a sequence of commands, we added the causal relationship between the Prompting node and both Learning and Remembering nodes to the model (Prompting→ Learning, Prompting→ Remembering). Moreover, good prompting reduces confusion during data entry and task completion, which in turn reduces errors that lead to reloading the page (Prompting→ Loop on the Same Page). Also, by clarifying available options, adequate prompting can prevent users from using the back button or choosing the wrong path (Prompting→Back, Prompting→Wrong Path).

For better clarification, a formulation of the four principal factors of Prompting is offered, namely P1 (Guiding Users for Action), P2 (Knowledge of Alternatives), P3

(Provision of Status Information), and P4=(Enhanced Help Accessibility) These factors will be represented by Eq. 6 to Eq. 9 respectively.

$$P_1 = \frac{number\ of\ actions\ with\ user\ guide}{total\ number\ of\ actions} \times 100 \qquad Eq.\ 6$$

$$P_2 = \frac{number\ of\ actions\ revealing\ all\ alternatives\ to\ the\ user}{total\ number\ of\ actions} \times 100 \qquad Eq.\ 7$$

$$P_3 = \frac{number\ of\ actions\ after\ which\ the\ users\ know\ their\ status}{total\ number\ of\ actions} \times 100 \qquad Eq.\ 8$$

$$P_4 = \frac{number\ of\ actions\ that\ help\ is\ accessible\ throughout}{total\ number\ of\ actions} \times 100 \qquad Eq.\ 9$$

Then the Prompting can be calculated using Eq. 10 where ω_i represents the weight assigned to each contributing factor.

$$Prompting = \sum_{i=1}^{4} \omega_i \times P_i \qquad Eq.\ 10$$

'Immediate Feedback' refers to the way in which a system responds to any user's actions from a simple keystroke to more complex series of commands. Regardless of the action taken, the system's response should be rapid, consistent, informative, and appropriate for the transaction being requested and its outcome. An illustrative quantification for this variable can be represented as Eq. 11.

$$Immediate\ Feedpack = \frac{number\ of\ actions\ with\ prompt\ response}{total\ number\ of\ actions} \times 100 \qquad Eq.\ 11$$

The quality and speed of this feedback help users to better understand how the system functions. If feedback is not provided or there is a delay in receiving it, users may become skeptical about the system's performance and request for unnecessary refreshing of the page, repeat the action, or take wrong actions that disrupt ongoing processes.

Therefore, this has a causal relationship with the correct path node in the defined Bayesian network model (Immediate feedback → Correct path).

The final criterion within this category is Legibility, which refers to the linguistic features of the content displayed on the screen that could hinder or aid in the reading of the information. We propose the formulation of this variable using Eq. 12.

$$Legibility = \frac{number\ of\ legible\ items}{total\ number\ of\ items} \times 100 \qquad Eq.\ 12$$

This includes factors such as character brightness, the contrast between the text and background, font size, spacing between words, lines, and paragraphs, as well as line length. According to Scapin and Bastien (1997), good legibility makes it easier and faster to read the information. The link between Legibility and Time is for considering this fact in the model (Legibility → Time).

Workload

The next main criterion is Workload, which encompasses all the interface elements that contribute to the user's cognitive or perceptual burden. A higher workload increases the chances of errors occurring. Furthermore, shorter actions lead to quicker interactions. Figure 8 illustrates two sub-criteria related to Workload, including Brevity and Information Density. The Brevity sub-criterion refers to the level of perceptual and cognitive workload for individual inputs and outputs, or a set of inputs needed to accomplish a task. We can introduce three key factors to quantify it: B1 (number of actions), B2 (average number of items per action), and B3 (length of items). Obtaining B1 and B2 is straightforward. However, when considering the length of items, it's important to note that its effect does not increase linearly based on the number of words. To account for this, we can assume that having more than seven words in an item significantly increases the error probability. Based on this assumption, the formulation in Eq. 13 is proposed for B3.

$$B3 = \sum_{\forall\, item} sigmoid(N_{words}, c_1 = 0.5, c_2 = 7) * N_{words} \qquad \text{Eq. 13}$$

Therefore, Brevity can be computed by Eq. 14

$$Brevity = \sum_{i=1}^{3} \omega_i \times B_i \qquad \textit{Eq. 14}$$

Good brevity implies that the entries are short and more succinct, which can reduce not only the reading time (Brevity→Time) but also the probability of making a mistake in filling the forms (Brevity→Loop on the Same Page) or getting tired to leave the task incomplete (Brevity→Incomplete). Effective brevity also means less reliance on the user's short-term memory, which in turn increases learning and memorizing the task process (Brevity→Learning, Brevity→Remembering).

The Information Density sub-criterion focuses on the user's perceptual and cognitive workload with regard to the entire set of information presented to the users rather than each individual element or item. To ensure an optimal information density, it is crucial to avoid providing users with unnecessary information that increases their cognitive load, while still presenting all the essential information they require to complete the task. Thus, we can consider two significant factors for information density: I1 (Irrelevant Items) and I2 (Missing Information), which are respectively defined by Eq. 15 and Eq. 16, and Information Density can simply be calculated based on their values using Eq. 17.

$$I1 = \frac{number\ of\ items\ that\ are\ not\ relevant\ and\ can\ be\ removed}{total\ number\ of\ items} \times 100 \qquad \text{Eq. 15}$$

$$I2 = \frac{number\ of\ actions\ that\ user\ needs\ more\ information\ to\ accomplish\ them}{total\ number\ of\ actions} \times 100 \qquad \text{Eq. 16}$$

$$Information\ Density = \sum_{i=1}^{2} \omega_i \times I_i \qquad \textit{Eq. 17}$$

By providing users with sufficient information without overwhelming them with unnecessary details, they can learn to navigate the website more efficiently and effectively (Information Density → Learning). When users have access to ample information, their ability to make informed choices regarding the correct path increases (Information Density → Correct Path).

Explicit Control

The criterion Explicit Control consists of two distinct sub-criteria, E1 (Explicit User Action), which means the computer must process only those actions requested by the user explicitly and only when requested to do so, and E2 (User Control), which means that users should always be able to control the system processing by interrupting, canceling, pausing, and continuing the process. Eq. 18 and Eq. 19 are the proposed formulas for these factors and Eq. 20 is proposed for quantifying the Explicit Control criterion.

$$E1 = \frac{\text{number of actions that will be processed only based on the user's request}}{total\ number\ of\ actions} \times 100 \qquad \text{Eq. 18}$$

$$E2 = \frac{number\ of\ actions\ that\ users\ have\ all\ kinds\ of\ control\ on\ them}{total\ number\ of\ actions} \times 100 \qquad \text{Eq. 19}$$

$$Explicit\ Control = \sum_{i=1}^{2} \omega_i \times E_i \qquad \textit{Eq. 20}$$

When computer processing results from explicit user actions and lets them control the interactions, users learn and understand better the application functioning and thus diminish the probability of making errors (Explicit Control→Learning).

Adaptability

Scapin and Bastien's perspective is that a single interface cannot cater to all of its potential users effectively. To prevent any negative consequences for the users, the interface must adjust to suit their requirements. The ability of a system to behave contextually and in accordance with the users' preferences and needs is referred to as its Adaptability. Adaptability comprises two sub-criteria: A1 (Flexibility) and A2 (User Experience). Flexibility pertains to the range of available methods for accomplishing a specific objective and refers to the users' ability to customize the interface to fit their habits and task demands. On the other hand, User Experience is concerned with the methods available to consider the users' level of familiarity with the system. There are various possible formulations for these two factors. However, a straightforward approach to quantify Flexibility is by simply counting the number of ways to accomplish the task. For User Experience, the percentage of items and actions that can be omitted for the experienced user is considered. This is shown by Eq. 21

$$A2 = \left[\frac{1}{2}\left(\frac{Number\ of\ items\ that\ are\ removed\ for\ experienced\ users}{total\ number\ of\ items\ in\ the\ task} + \frac{Number\ of\ actions\ that\ are\ removed\ for\ experienced\ users}{total\ number\ of\ actions\ in\ the\ task}\right) \times 100\right] \qquad \text{Eq. 21}$$

Then, Adaptability will be obtained by Eq. 22

$$Adaptabiliyty = \sum_{i=1}^{2} \omega_i \times A_i \qquad \textit{Eq. 22}$$

If there are many different methods available to complete a task, then it is more likely that a specific user will find a method that suits them. This method will be easier for them to learn and remember over time. (Adaptability→Learning, Adaptability→Remembering). Furthermore, users with varying levels of experience have different information requirements. It may be necessary to offer inexperienced users clear and concise step-by-step procedures that adapt the interface to their needs for simplicity and guidance and thus reduce errors or leave tasks incomplete (Adaptability→Correct Path). On the other hand, experienced users may find some procedures tedious and slow down their interaction. In such cases, offering shortcuts can allow them to access system functions more quickly (Adaptability→Time).

Error Management

The next criterion is Error Management which is composed of three sub-criteria: "Error Protection", "Quality of Error Messages", and "Error Correction". They respectively refer to the ways available to prevent, decrease, and recover from errors, such as invalid data entry, incorrect data format, or incorrect command syntax. "Error Protection" relates to the ability to detect and avoid errors in data entry and their format, commands, or actions that could result in harmful consequences. Two have a high Error Protection; it is required to first prevent errors and then if an error occurred prevent destructive consequences. Therefore, it can be quantified by computing the weighted average of two primary factors EP1 (Error Prevention) and EP2 (Destructive Consequences) that are respectively demonstrated by Eq. 23 and 24.

$$EP1 = \frac{number\ of\ actions\ that\ prevent\ data\ entry\ or\ command\ errors}{total\ number\ of\ actions} \times 100 \qquad \text{Eq. 23}$$

$$EP2 = \frac{number\ of\ actions\ that\ may\ cause\ destructive\ consequnces}{total\ number\ of\ actions\ in\ the\ task} \times 100 \qquad \text{Eq. 24}$$

"Quality of Error Messages" on the other hand, pertains to the phrasing and content of error messages, including their accuracy, readability, and specificity in addressing the nature of the error (e.g., syntax or formatting issues) and providing necessary steps to correct it. This criterion is decomposed into four distinct factors: EQ1 (Error Message Relevance), EQ2 (Error Messages Readability), EQ3 (Error Message Specificity), and EQ4 (Present Error Solution); each one can be defined by the following formulas:

$$EQ1 = \frac{Number\ of\ error\ messages\ that\ are\ relevant\ to\ the\ error}{Total\ number\ of\ error\ messages} \times 100 \qquad \text{Eq. 25}$$

$$EQ2 = \frac{Number\ of\ error\ messages\ that\ are\ readable}{Total\ number\ of\ error\ messages} \times 100 \qquad \text{Eq. 26}$$

$$EQ3 = \frac{Number\ of\ error\ messages\ that\ explicitly\ specify\ error}{Total\ number\ of\ error\ messages} \times 100 \qquad \text{Eq. 27}$$

$$EQ4 = \frac{Number\ of\ error\ messages\ in\ a\ task\ that\ present\ solution}{Total\ number\ of\ error\ messages} \times 100 \qquad \text{Eq. 28}$$

The last criterion in this group is "Error Correction" (EC), which pertains to the methods available to users for fixing their errors and can be defined by Eq. 29.

$$EC = \frac{Number\ of\ actions\ allow\ users\ to\ correct\ potential\ errors}{Total\ number\ of\ actions} \times 100 \qquad \text{Eq. 29}$$

Based on all the above factors, Error Management will be calculated by Eq. 30.

$$\text{Error Management} = \sum_{i=1}^{2} \omega_i \times EP_i + \sum_{i=1}^{4} \alpha_i \times EQ_i + \beta \times EC \qquad \text{Eq. 30}$$

When Error Protection is effective, it can minimize system interruptions caused by user errors. For instance, a website that has a strong level of error protection checks the validity of data before transmitting it to the server and therefore reduces the same page reloading. This imposes the "Error Management"→"Loop on the Same Page" relationship to the model. Error Protection also reduces the number of interactions and time by limiting the number of interruptions (Error Management→Time). Moreover, it would be less disruptive if errors can be detected in advance and corrected easily and immediately after the occurrence. Thus, appropriate Error Protection and Correction can prevent interference with task completion, which implies adding Error Management→Incomplete to the model. Moreover, the quality of error messages can enhance users' understanding of systems by explaining the reasons for their errors and teaching them how to avoid or fix these errors (Error Management→Learning).

Significance of Codes

The criterion Significance of Codes(SoC) assesses the correlation between a term or symbol and its intended meaning. Codes and names become meaningful to users when there is a clear semantic connection between the code and the item or action it represents. This criterion can be represented by Eq. 31.

$$SoC = \frac{Number\ of\ items\ with\ names/signs\ semantically\ related\ to\ the\ references}{Total\ number\ of\ items} \times 100 \quad \text{Eq. 31}$$

Meaningful codes are easier to remember and recognize (Significance of Codes→Remembering), while non-meaningful codes or names can cause users to perform incorrect actions, resulting in errors (Significance of Codes→Correct Path).

Consistency

The Consistency criterion pertains to the uniformity of interface design choices, such as codes, naming conventions, formats, and procedures, across similar contexts while ensuring that they are different in distinct contexts. Eq. 32 is proposed for defining consistency.

$$Consistency = \frac{\sum_{\forall\ meaningful\ element} \frac{1}{number\ of\ varient\ of\ that\ element\ in\ all\ pages}}{total\ number\ of\ meaningful\ elements} \times 100 \quad \text{Eq. 32}$$

By keeping procedures, labels, and commands in a consistent format, location, and syntax from one screen to the next and from one session to the next, users are more likely to remember, locate, recognize, and use them effectively. Inconsistent design choices can significantly increase search time and reduce the likelihood of errors. Furthermore, a consistent website is more predictable, making it easier for users to learn and generalize. Therefore, the connection (Consistency→Remembering), (Consistency→Time), (Consistency→Learning), and (Consistency→Correct Path) are added to the model.

Compatibility

Scapin and Bastien (1997) introduced Compatibility as the final main criterion among their independent variables. They defined Compatibility as the evaluation of how well the psychological characteristics of users (memory, perception, habits, skills, age, expectations, etc.) match the task characteristics. However, this criterion overlaps with other criteria such as Legibility, Workload, Significance of Code, and Adaptability. As independent variables are being sought, the Compatibility criterion is excluded from the proposed model.

Moreover, it is important to note that the user's experience with the system can impact the variables of Learning and Remembering. As users become more familiar with the system through regular usage, they are likely to learn more and recall it better. Therefore, the edges (Number of Use→Learning) and (Number of Use→Remembering) are added to the model. On the other hand, the fact that a longer time interval between system usage causes forgetting leads us to add the (Time Gap→Remembering) connection to the Bayesian network. Moreover, both Learning and Remembering can also influence the variables of task completion speed and error rate. When a user knows how to perform a task, they are more likely to do it quickly and without errors (Remembering → Time, Remembering → Correct path, Learning → Time, Learning → Correct path).

The diagram presented in Figure 9 depicts all the interrelationships involved in the complete model design. It represents the finalized Bayesian network model that will be employed to generate log data. During this process, values for each variable associated with the usability criteria will be selected from a predetermined truncated normal distribution. The distribution parameters for each variable including mu, sd, α, and β can be defined by usability experts and based on the formula proposed for each one of them. Moreover, the weights of the network can be initialized by usability experts and if a database is collected in the future, they can be improved using parameter learning methods and based on real data.

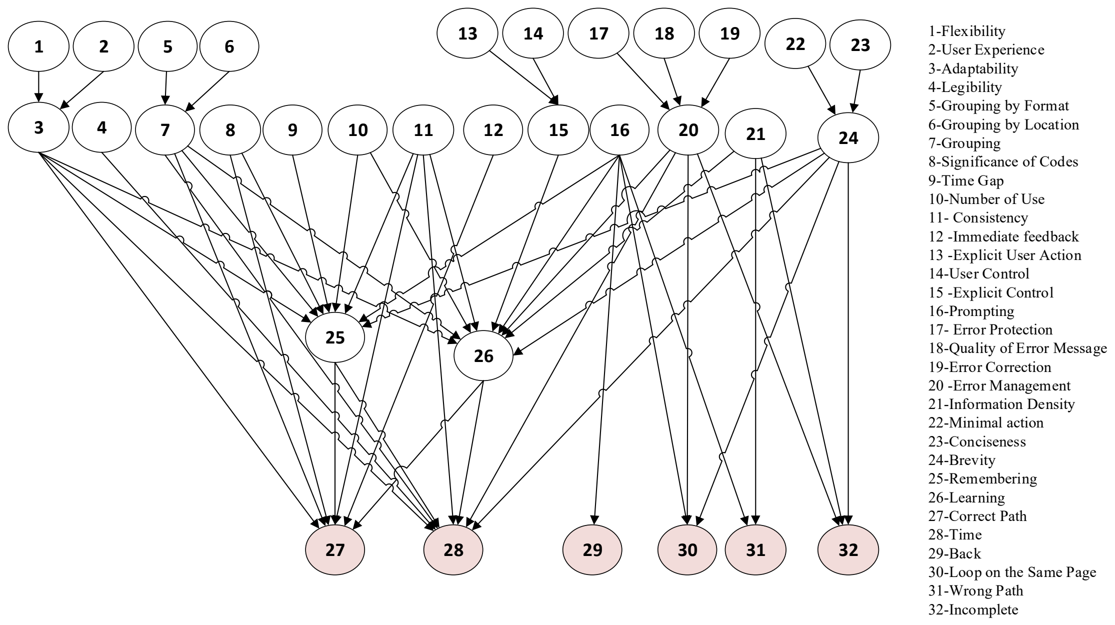


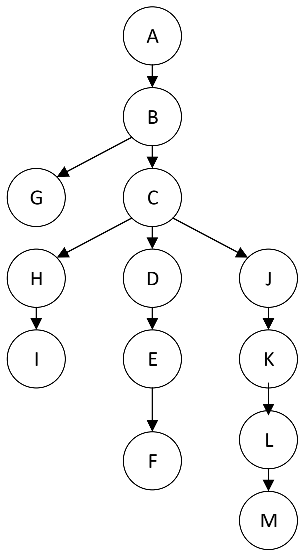
Figure 9: Bayesian network for modeling user behaviors based on usability features of a website

Once a random value is assigned to each usability criterion, the model will be used to derive the possible distribution of the variables that represent the user's behavior, such as time of performing an action (Time), clicking on back (Back), clicking on the correct link (Correct Path), clicking on an incorrect link (Wrong Path), stopping halfway through a task (Incomplete), and Loop on The Same Page. Subsequently, this distribution will be used to produce the simulated log file, as outlined in the next section.

4.3.2 The Procedure of Creating Synthetic Data

In the previous section, a model was introduced that assessed the impact of usability features of web pages on the likelihood of performing correct or incorrect actions, as well as the distribution of the required time for executing these actions. While this information is essential for determining the user's next move, it is still insufficient for simulating users' behavior and therefore log file transactions. For instance, if the Bayesian model suggests that the probability of making a mistake in selecting the right path is high, then a wrong path that represents their error must be created in the simulation process to reflect this information. However, the question remains: how and with what pattern should the error be added to the user's movement model? Randomly including one or more pages to the original path to create this path does not mirror the user's behavior in the real world. In reality, errors do not occur randomly, and various factors, including the website or application's structure, play a crucial role in determining user path patterns. As an example, Figure 10(a) portrays a sample sitemap. Suppose a user is undertaking a task that involves traversing through A→B→C→D→E→F and the Bayesian model indicates a high probability of occurrence error when the user reaches page C. In such a case, Figure 10(b) represents a list of feasible paths that the user can undertake on this website. However, a Path like A→B→C→K→I cannot occur since there exists no link between pages C and K or between pages K and I. Alternatively, in a different scenario, if the Bayesian model indicates a high probability of occurrence of errors when the user reaches page E, the sole possibility is using the back button since page F is the only page accessible based on this website's structure.

The importance of this fact becomes evident when the objective is to identify the task models or detect errors from using analyzing a log file. In practical scenarios, where the user's error paths are influenced by the website's structure, a single error path may be replicated numerous times by different users. Consequently, while automatically identifying the task model, an erroneous path could be inaccurately classified as one of the task models. However, if the simulation procedure introduces these error paths randomly into the user's behavioral model, it would considerably simplify the identification and elimination of errors by machine learning algorithms, and it will not encounter the challenges of analyzing actual log files.



(a) sitemap

Task Model:
A→B→C→ D→E→F

User selected the wrong path on page C and halted after two steps:
A→B→C→ J→K→L

User selected the wrong path on page C but return after two steps:
A→B→C→ J→K→J→C→D→E→F

User reload the page C:
A→B→C→C→ D→E→F

Using clicked on back button on page C:
A→B→C→B→C→ D→E→F

(b) Sample of possible user's path pattern

Figure 10: A sample sitemap and a list of possible users' path

Hence, it is crucial to incorporate a website structure in conjunction with a Bayesian network for log file simulation. Figure 11 outlines an algorithm that generates a randomized website structure according to the number of tasks and the number of pages for each task.

Therefore, this algorithm utilizes three input parameters to generate task models. The first parameter, 'n_task,' denotes the number of tasks within the system. For each task, the algorithm employs the second and third parameters, 'mu_Npage' and 'sd_Npage,' along with a normal distribution to determine the number of pages to request from the server while performing that task. The primary objective of the algorithm is to establish a set of task models that adhere to a unified website structure. To accomplish this, the algorithm initially generates a random list of pages as the task model for the first task. This list outlines the sequence of pages that users must navigate, starting from the home page and concluding at the final goal. However, the approach differs for subsequent tasks. For each new task with 'Np' pages, the algorithm begins by randomly selecting one of the pages whose depth from the root is less than 'Np.' Let's refer to this page as 'P.' Consequently, the clickstream path for this new task is shared from the root to 'P,' while the remaining pages form a new branch extending from 'P'.

```
Random Sitemap Generator (n_task, mu_Npage,sd_Npage)
sitemap = Create an empty Tree to store the website structure.
for i  in range(n_task):
        - n_page[i] = randomly assigning a value from a normal distribution with a mean (mu_Npages) and a
        standard deviation (sd_Npages).

        - Generate a random number between 1 and min(n_page[i] - 1, depth of sitemap). Call this number "k".

        - if k > 0 :
                Select one of the pages in depth k of sitemap and call it parentNode

        - Generate a list of n_page[i]-k random URL using a combination of letters, numbers, and symbols.

        -if k > 0:
                Add this list as an out going link to parentNode
        else:
                Add this list to the depth 0 of the sitemap
return the sitemap
```

Figure 11: Random sitemap generation algorithm[2]

It should be noted that this algorithm does not claim to generate all types of sitemaps or statistically accurate sitemaps reflective of real-world scenarios. However, this is not a requisite for the purposes of this study as well. The primary objective is to have a structure that enables the inclusion of errors into the model based on it.

---

[2] Implementation of this algorithm is available at GitHub

Figure 12 shows two examples of sitemaps created by the above algorithm for 5 tasks while the number of pages follows a normal distribution characterized by a mean of four and a standard deviation of two.

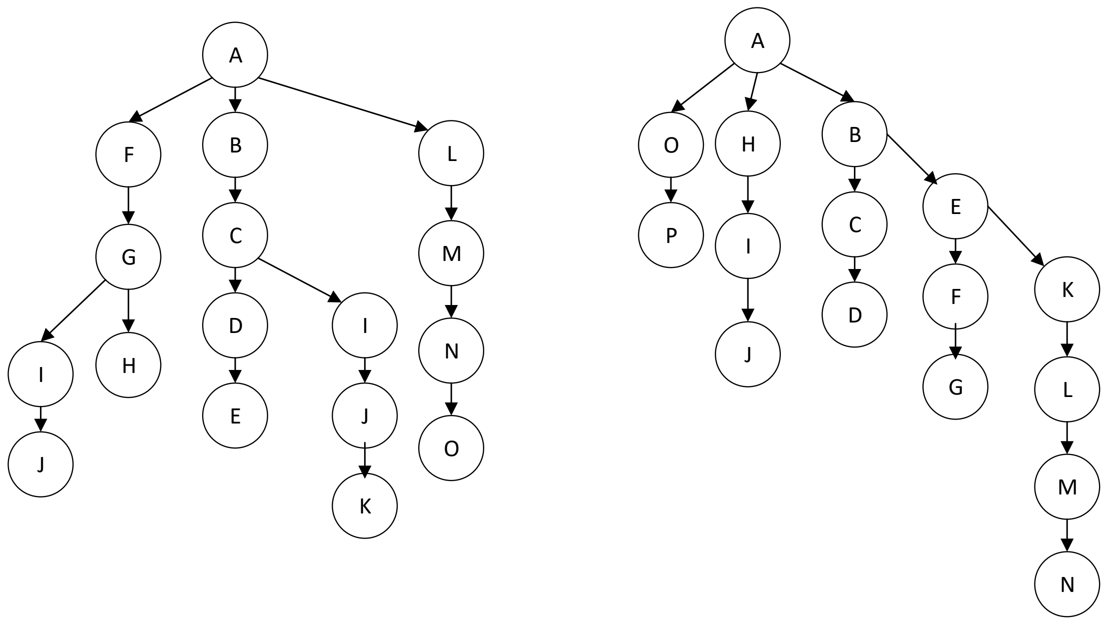


Figure 12: Sample Sitemap created by algorithm represented in Figure 11

Having all the essential tools, the log file simulation is executed in accordance with the following meticulously designed process:

1. *Generate a random sitemap based on the task count.*
2. *Assign usability features randomly to each page in the sitemap, based on their predetermined distributions.*
3. *Randomly select the number of roles, the number of users associated with each role, and the activity frequency of each role in the system, with predefined distributions.*
4. *Determine the distribution of performing each task by each role.*
5. *Randomly select a role based on its activity frequency in the system.*
6. *Randomly select a user from the pool of users assigned to the selected role.*

7. *Randomly select a task for the user, based on the distribution of task performance for the user's role.*
8. *Select the entrance time based on exponential distribution with parameter* $\boldsymbol{\lambda}$ *and the time of the previous entrance.*
9. *Set the page equal to the first page of the task.*
10. *Feed the user and page information into the Bayesian network to determine the next move of the user.*
11. *Based on the user's move determined in step 9 and the sitemap, select the next page that the user will request.*
12. *Repeat steps 9 and 10 until the task is completed or the user leaves the task incomplete.*
13. *For each page visited by the user, create a transaction.*
14. *Sort all transactions based on their time in save the result in the log file.*[3]

In this process, the user login events occur continuously and independently at a constant average rate, following a Poisson point process. Therefore, to determine the time interval between two consecutive logins, an exponential distribution is utilized. A single parameter, $\boldsymbol{\lambda}$, has been used for all types of tasks. However, it is also possible to define different parameters based on the selected task in the previous step, considering the varying time occurrence distribution of tasks within the system.

[3] The implementation of this algorithm is available in GitHub

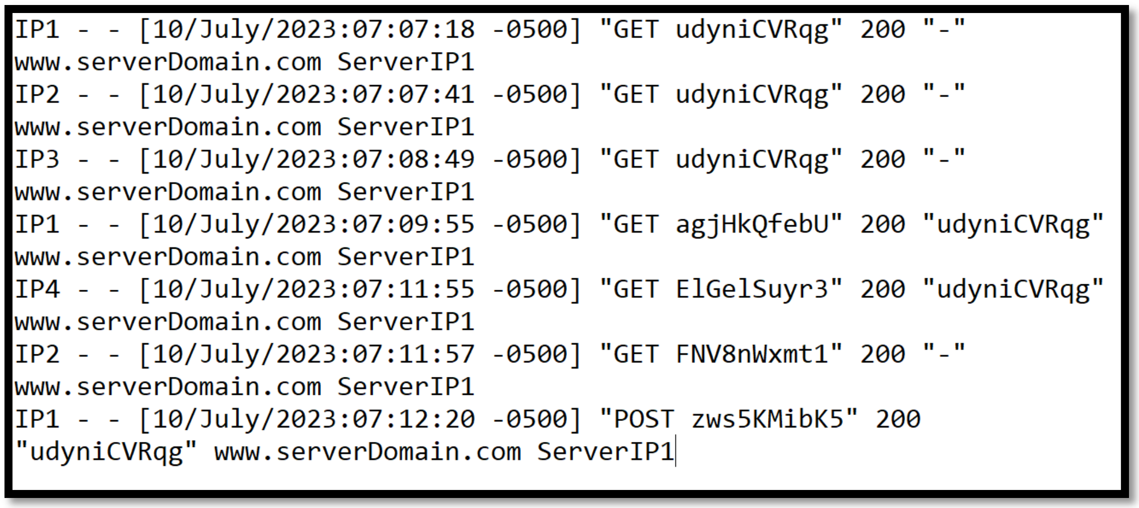
4.3.3 Results

An example of the created log file is shown in Figure 13. As you can see, instead of registering the IP address, the IP code is used and instead of registering the complete HTML link of the page, a randomly generated name is used to determine each page. Replacing these codes with random IP addresses or HTML links is very straightforward. However, for the sake of readability, the step of replacing them is omitted.

In addition, to enhance the possibility of incorporating geographic information and researching on user locations, creating a dataset of random IP addresses and their allocated locations is suggested. Then, when selecting a user, we can consider the distribution of users' activities across various locations to choose a corresponding user to carry out the task.

In an Apache access log file, there are two other fields that are omitted in the final result: (1) the size of the response sent back to the client and (2) the User-Agent string, which provides detailed information about the web browser or client software used by the user to access the web page and its version.

The determination of whether to record the "POST" or "GET" method for each transaction is based on the page specification. If the page includes forms that require the user to input and submit data to the server, the "POST" method will be used. On the other hand, if the user intends to retrieve information from the server without submitting any data, the "GET" method will be recorded for the transaction.



```
IP1 - - [10/July/2023:07:07:18 -0500] "GET udyniCVRqg" 200 "-"
www.serverDomain.com ServerIP1
IP2 - - [10/July/2023:07:07:41 -0500] "GET udyniCVRqg" 200 "-"
www.serverDomain.com ServerIP1
IP3 - - [10/July/2023:07:08:49 -0500] "GET udyniCVRqg" 200 "-"
www.serverDomain.com ServerIP1
IP1 - - [10/July/2023:07:09:55 -0500] "GET agjHkQfebU" 200 "udyniCVRqg"
www.serverDomain.com ServerIP1
IP4 - - [10/July/2023:07:11:55 -0500] "GET ElGelSuyr3" 200 "udyniCVRqg"
www.serverDomain.com ServerIP1
IP2 - - [10/July/2023:07:11:57 -0500] "GET FNV8nWxmt1" 200 "-"
www.serverDomain.com ServerIP1
IP1 - - [10/July/2023:07:12:20 -0500] "POST zws5KMibK5" 200
"udyniCVRqg" www.serverDomain.com ServerIP1
```

Figure 13: Sample of the synthetic log file

Figure 14 illustrates the impact of website structure on users' actions, by comparing two different website structures. The first structure features a simpler design, with fewer outgoing links on each page. The degree of each node in the sitemap graph directly reflects the number of outgoing links found on each page. In this structure, the degree of each node in the sitemap graph which represents the number of outgoing links, is a maximum of 3 with an average degree of 1.02. Tasks within this structure typically span around 7 pages, indicating that users, on average, navigate through 7 pages to complete a task.

On the other hand, the complex structure showcases a more intricate design with a maximum degree of 10 and an average degree of 5.51 for each node in the sitemap graph. Despite the structural complexity, the average task length remains the same at 7 steps. In both cases, the first 1000 user transactions on the websites were examined, monitoring user actions such as back button usage, page reloads (loops), incorrect path choices, and incomplete task performances. Considering that if none of these actions are performed by the users, they have inevitably chosen the correct path, counting and displaying the number of correct path choices in the result was excluded, assuming that any uncounted action implied the user had chosen the correct path.

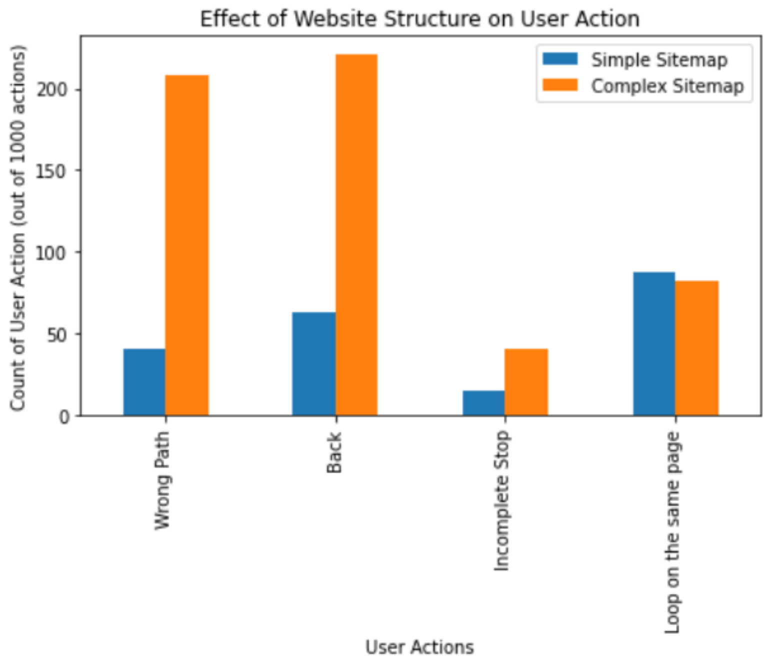


Figure 14:Effect of website structure on users' actions

To ensure fairness, the usability specifications of all pages on both websites are fixed and similar by employing the same probability distribution for all pages. Usability specifications that reflect a suboptimal design were incorporated to amplify the likelihood of user errors and to highlight the impact of the website structure. Figure 14 clearly demonstrates that the complex structure leads to a nearly five-fold increase in incorrect path choices. Similarly, there is a notable rise in both incomplete tasks and back button usage. This outcome aligns with logical reasoning since whenever users deviate from the correct path, they must retrace their steps or abandon the task altogether. To rectify their course, users may rely on the back button or explore alternative links that redirect them to the appropriate pages, ensuring they resume their intended path. Consequently, the upward trend in incorrect path choices correlates with heightened back button usage and a greater occurrence of incomplete paths.

Furthermore, it is worth noting that the occurrence of page reloading remains relatively consistent across both structures, with a slightly higher incidence observed in the simpler structure, which may be due to the randomness of the user's choices. This result is acceptable because the website's structure itself does not directly impact the frequency of page reload requests or loops on the same page. In fact, page reloading commonly arises when users encounter errors while completing form submissions or manually refreshing pages due to a lack of expected response. Hence, it can be inferred that page reloading is predominantly influenced by the usability features of individual pages rather than the overall website structure.

The third experiment delves into examining the influence of three selected usability features on users' actions (Figure 15 (a, b, c)) and the overall completion time of activities (Figure 15 (d)). To conduct this test, a fixed page is employed to isolate and evaluate the impact of each individual usability feature, while all other usability aspects remain consistent. The first test focuses on exploring the significance of Code (SoC), the second on Prompting, and the third on Error Protection.

Considering the formulas assigned to each feature (Eq. 31, Eq. 10, and Eq. 23 & 24, respectively), their values are represented as a percentage, ranging from 0 to 100. To comprehensively assess their effects, specific values of 20%, 40%, 60%, 80%, and 100% are employed for each attribute. Furthermore, the experiments elucidate how modifications in each of these features influence the time required to complete a single step of a task, specifically referring to the completion of an operation performed on a single page. These results provide valuable insights into the relationship between usability features and task completion performance.

Each graph's y-axis represents the mean value derived from the normal distribution of user actions obtained through the implementation of the designed Bayesian network. The Bayesian network model functions by obtaining the probability distribution of each variable, where in this dissertation’s model, the variable corresponds to the likelihood of a specific user action being performed. This probability distribution is determined based on other observed variables and model parameters.

Hence, to determine which specific action will be performed by the user among the available options, a random value is extracted from the respective probability distribution associated with each action in the Bayesian network. This extracted value represents the probability of executing the corresponding action. Ultimately, the action with the highest probability is chosen as the final outcome. For instance, in the first experiment, when the significance of code is set to 20%, the mean value of the normal distribution for the correct path is comparatively lower than that of wrong path choices and loops on the same page. Consequently, when a number is randomly sampled from these distributions, there is a higher probability of selecting either a wrong path or a loop on the same page. Conversely, when the significance of code increases to 100% while keeping other features constant, the probability of successfully following the correct path is expected to notably increase.

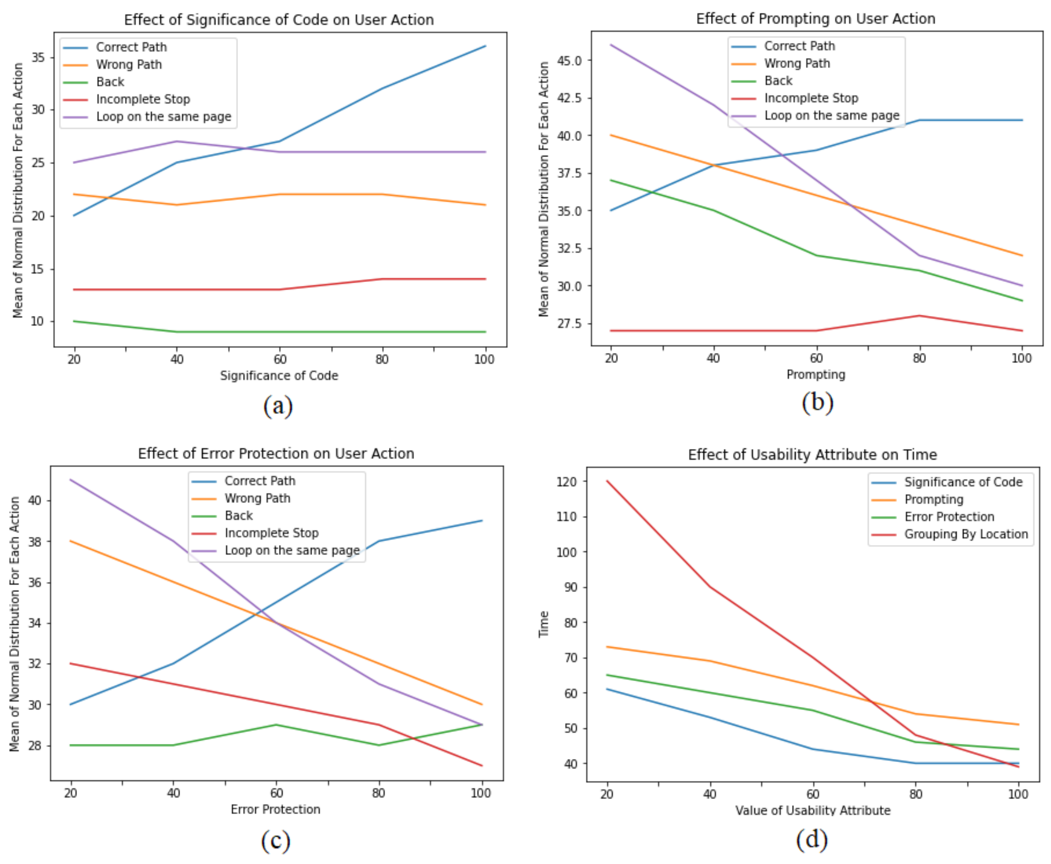


Figure 15: Effect of three selected usability attributes on users' actions (a,b,c) and four selected attributes on task completion time(d)

The results depicted in Figure 15 were obtained using predetermined values for both other usability features and model parameters. Consequently, modifying these parameters can alter the slope and characteristics of the presented plots. To ensure the reliability and relevance of our findings, extensive testing was performed, and the parameters were carefully adjusted in collaboration with 2 web designers and one usability expert. This iterative process helps ensure that the obtained results align with expectations within the field.

As illustrated in Figure 15 (d), incremental enhancements in SoC, Error Protection, and Prompting lead to slight improvements in task completion time. The range of completion time decreases from 80-60 seconds to 60-40 seconds as these variables are refined. Considering that these three attributes have almost the same multiplier effect on reducing the task time, to verify the accuracy of the model, the impact of Grouping on task completion time was also examined. It becomes evident that improving Grouping significantly reduces the overall task completion time from 120 to 40 seconds. This outcome aligns with logical reasoning since a cluttered and chaotic design necessitates users spending considerable time searching for desired buttons and links. By implementing effective Grouping and organization strategies, users can swiftly locate relevant elements, resulting in a substantial reduction in task completion duration.

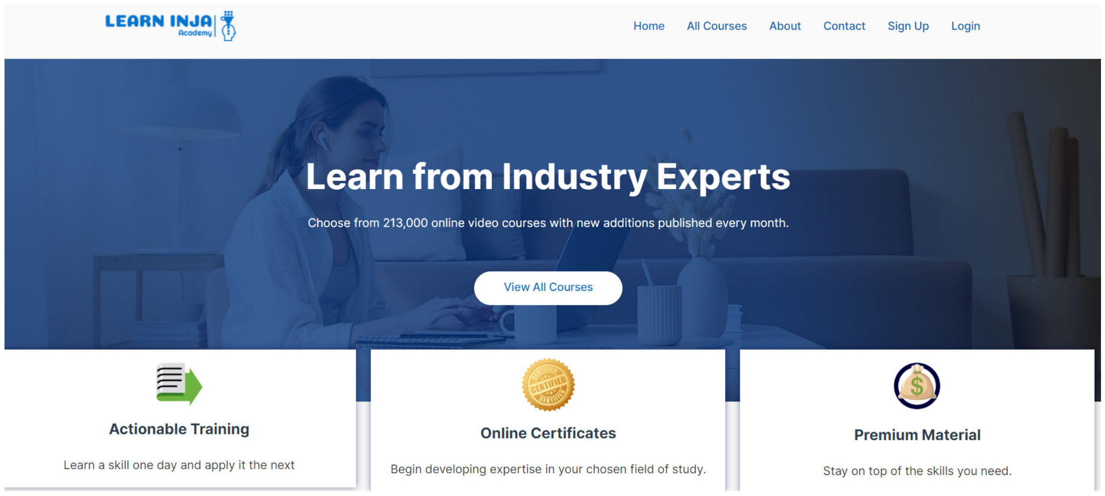
4.4 Real data

In addition to the synthesized data, testing on real data is also essential to assess the performance of the presented methods. To this aim, various real log data samples have been studied such as those from GitHub, Kegel, or different universities. A sample of these datasets are listed in Table 17. However, their usage for the intended purpose of this dissertation can be challenging due to their certain limitations. For instance, some data are restricted to a short time period or do not possess sufficient transactional data to perform the desired methods. Additionally, some source websites may not be accessible, or the available access logs may not be up-to-date, which makes it difficult or even impossible to compare the manual and traditional usability evaluation methods of the website with the outcome of automatic usability evaluation. Moreover, if these data are used, it is not feasible to make modifications to the website, define tasks for users, and collect log files associated with these changes and tasks. To overcome these limitations, I have developed a website called Learninjawebsite.com, which provides easy access to its log file whenever needed.

Table 17: Sample of publicly available access log dataset

Website title	Description	Access Link
NASA-HTTP	Two months of all HTTP requests to the NASA Kennedy Space Center WWW server in Florida	https://www.kaggle.com/datasets/souhagaa/nasa-access-log-dataset
Calgary-HTTP	Approximately one year's worth of all HTTP requests to the University of Calgary's Department of Computer Science WWW server located in Calgary, Alberta, Canada.	https://ita.ee.lbl.gov/html/contrib/ClarkNet-HTTP.html
Saskatchewan-HTTP	Seven months of all HTTP requests to the University of Saskatchewan's WWW server. The University of Saskatchewan is located in Saskatoon, Saskatchewan, Canada.	https://ita.ee.lbl.gov/html/contrib/Sask-HTTP.html
UC Berkeley Home IP Web Traces	This dataset consists of 18 days' worth of HTTP traces gathered from the Home IP service offered by UC Berkeley to its students, faculty, and staff Home IP provides.	https://ita.ee.lbl.gov/html/contrib/UCB.home-IP-HTTP.html
Online Shopping Store	Nginx server access log for an online shopping store (2018-12-10)	https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/3QBYB5

The Learninja website offers four different roles, including public user, student, instructor, and site administrator. Public users have access to a variety of pages, such as Home, About, Contact Us, and All Courses. They can also search for courses based on their titles, categories, levels, and creator names, and send messages via the website. Furthermore, public users can register as students or instructors, granting them access to additional features. As a student, a user can enroll in courses, save courses to their wish list, complete courses and quizzes, ask questions in the course forum, and rate registered courses. Similarly, registered teachers, after approval from the site administrator, can define new courses and new categories, create quizzes for their courses, conduct polls on the lessons, and respond to questions in their course forums. The site administrator has the power to approve teachers, edit website information, reply to messages, and manage student and teacher accounts, along with defined courses and categories. Additionally, all registered users have the ability to view and modify their profile information. Figure 16 and Figure 17 show two sample screenshots of this website.



Learn from Industry Experts

Choose from 213,000 online video courses with new additions published every month.

Actionable Training

Learn a skill one day and apply it the next

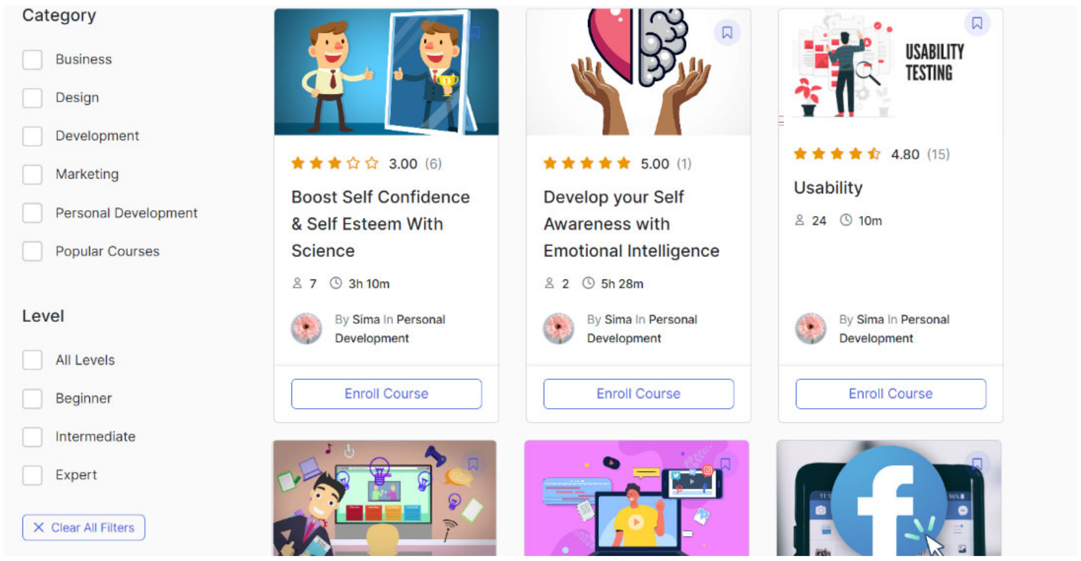
Online Certificates

Begin developing expertise in your chosen field of study.

Premium Material

Stay on top of the skills you need.

Figure 16: Home page of learninjawebsite.com



3.00 (6)
Boost Self Confidence & Self Esteem With Science
7 3h 10m
By Sima In Personal Development

5.00 (1)
Develop your Self Awareness with Emotional Intelligence
2 5h 28m
By Sima In Personal Development

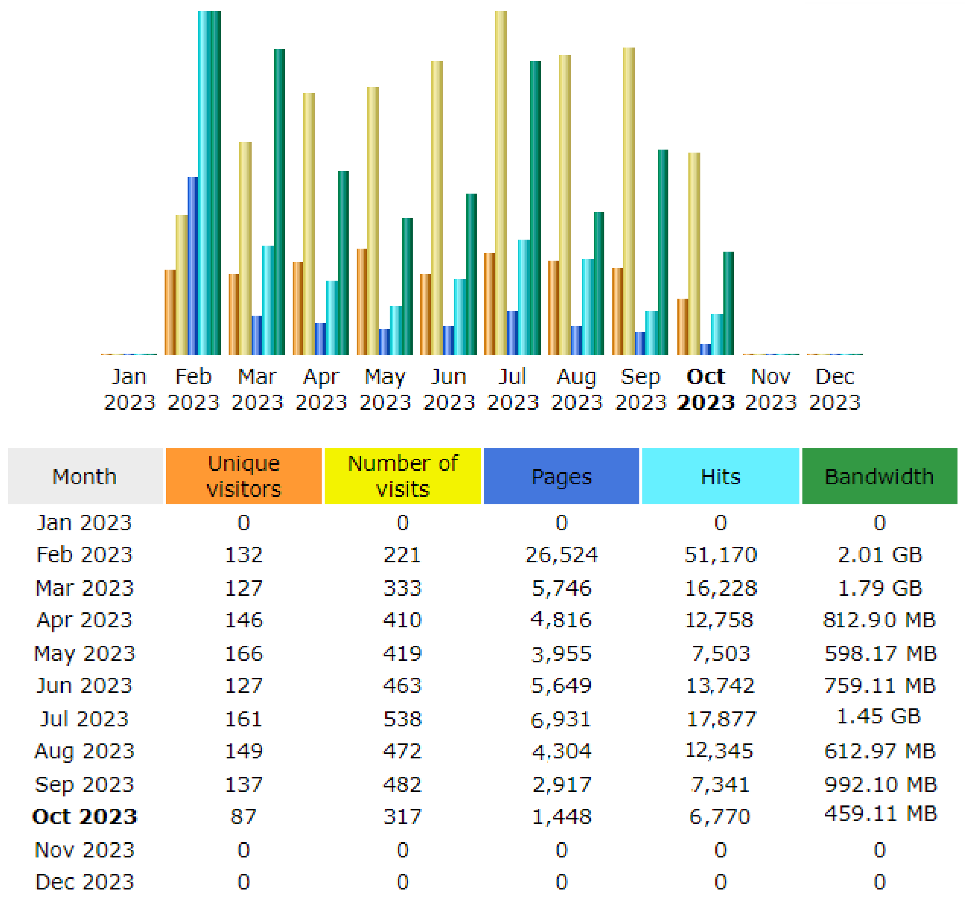
4.80 (15)
Usability
24 10m
By Sima In Personal Development

Figure 17: All Courses page of learninjawebsite.com and available filter for search

This website was created on Jan 2023 and has been available to the public since Feb 2023. So far, 261 people are registered as a student, and 33 people are registered as an instructor.

As a proof of concept, 10 tasks shown in Appendix A were defined and given to students to perform with the web site. Each person was asked to complete two or three tasks from the list. This data collection was approved by the University of Louisville’s Institutional Review Board (IRB); the informed consent is provided in Appendix B.

Several tools such as Awstats, Webalizer, and Google Analytics are utilized to monitor and perform primitive analyses on recorded information in the log file. Figure 18 shows a sample report obtained by Awstats which represents visiting information (from the beginning of Feb 2023 to mid Oct 2023). As evident from Figure 18, the data for February exhibits notable differences from the other months. This discrepancy is attributed to the website's development and launch during this period, resulting in a surge of content and page creation activities. Consequently, for subsequent analyses, the data for this month was excluded. Furthermore, the transaction volume for the months of March and July exceeds that of the other months. This can be attributed to the allocation of specific tasks to a group of students during these months, with each student assigned particular responsibilities. The data from these months will be given greater consideration in the subsequent stages of analysis and evaluation.



Month	Unique visitors	Number of visits	Pages	Hits	Bandwidth
Jan 2023	0	0	0	0	0
Feb 2023	132	221	26,524	51,170	2.01 GB
Mar 2023	127	333	5,746	16,228	1.79 GB
Apr 2023	146	410	4,816	12,758	812.90 MB
May 2023	166	419	3,955	7,503	598.17 MB
Jun 2023	127	463	5,649	13,742	759.11 MB
Jul 2023	161	538	6,931	17,877	1.45 GB
Aug 2023	149	472	4,304	12,345	612.97 MB
Sep 2023	137	482	2,917	7,341	992.10 MB
Oct 2023	87	317	1,448	6,770	459.11 MB
Nov 2023	0	0	0	0	0
Dec 2023	0	0	0	0	0

Figure 18: Summary of AWstats report on LearninjaWebsite.com logfile after excluding bots or spiders

The final dataset consists of 128078 records related to 3655 visits. On average, users had a click length of 12, and the typical time spent by users in a session was 315 seconds. Summary information of this log file is presented in Table 18.

Table 18: Summary information of logfile of learnInjaWebsite.com

Total number of records in log file	128078
Total number of distinct IP addresses	1232
Average pages per visit	12
Average session time	315(s)

CHAPTER 5

IMPLEMENTATION OF AUTOMATIC USABILITY EVALUATION FRAMEWORK

5.1 Introduction

In the previous chapters, we have thoroughly underscored the significance and relevance of the research topic at hand. It is now imperative to delve deeper into the intricacies of implementing the proposed approach, including the detailed characterization of real and synthesized data, the essential pre-processing steps, methodological considerations, and modeling approaches. Subsequently, I will ascertain the efficacy of this approach by formulating a series of experiments and evaluating the outcomes.

The primary objective of this dissertation is to present a comprehensive methodology for the automated evaluation of interactive systems' usability via log files. This chapter will first introduce a general framework in Section 5.2 that can be applied for any type of log file as input and produce any usability attribute as output. However, to restrict the scope of the project, the implementation step will later exclusively concentrate on server log files as input and learnability metrics as the output.

In the subsequent sections of this chapter, we adhere to the presented framework and provide an in-depth explanation of the method's implementation. Section 5.3 elucidates the data preparation and preprocessing steps. Following the completion of preprocessing, the data proceeds to the phase of knowledge discovery and analysis, aimed at extracting relevant usability metrics.

Section 5.4 elaborates on the proposed two-step approach for knowledge extraction. This section's objective is to cluster the records from the log file, aligning them with the user's intended task. Once the task association of each sequence of transactions or visited pages is established, subsequent knowledge analysis such as error analysis and time analysis for each task becomes more streamlined. Then, in Section 5.4.1, a range of metrics is specified that can be extracted from the log files, along with their respective calculation algorithms. Finally, the experiments, evaluation, and validation of results are presented in Section 5.6 to determine the accuracy and effectiveness of the proposed method.

5.2 The Proposed Framework

The proposed framework, as illustrated in Figure 19, builds upon the web usage mining procedure outlined in Section 3.3. I have introduced an additional step to this framework, specifically for modeling usability attributes, in order to separate this more targeted analysis from other intermediate knowledge analyses. Therefore, this model comprises five main phases including Data Collection, Preprocessing, Knowledge Discovery, Knowledge Analysis, and Data Modeling.

- Data Collection involves gathering real log files from various sources such as clients, proxy servers, and web servers, in addition to generating synthetic log data. The process of generating synthetic server log files is detailed in the previous chapter.
- The preprocessing procedure is highly dependent on the type of log file being used. For example, the preprocessing required for application log files may

differ from that for web log files. Even web log files from different sources may not have identical preprocessing steps. For instance, client log files require log file retrieval as the first preprocessing step in order to collect log files from different clients. On the other hand, they do not undergo the path completion step mentioned in Section 3.3.1., as no interaction is lost due to the caching process.

Therefore, to maintain the generality of this model that can be applied to all types of log files, I have included only the least options that are common among all types of log files as preprocessing steps including Data Fusion or aggregation, Parsing, Data Cleaning, Normalization, Filtering, and Data Formatting. However, higher levels of preprocessing can still be incorporated into this model through the loop in the model and repetitive execution of steps 3 and 4.

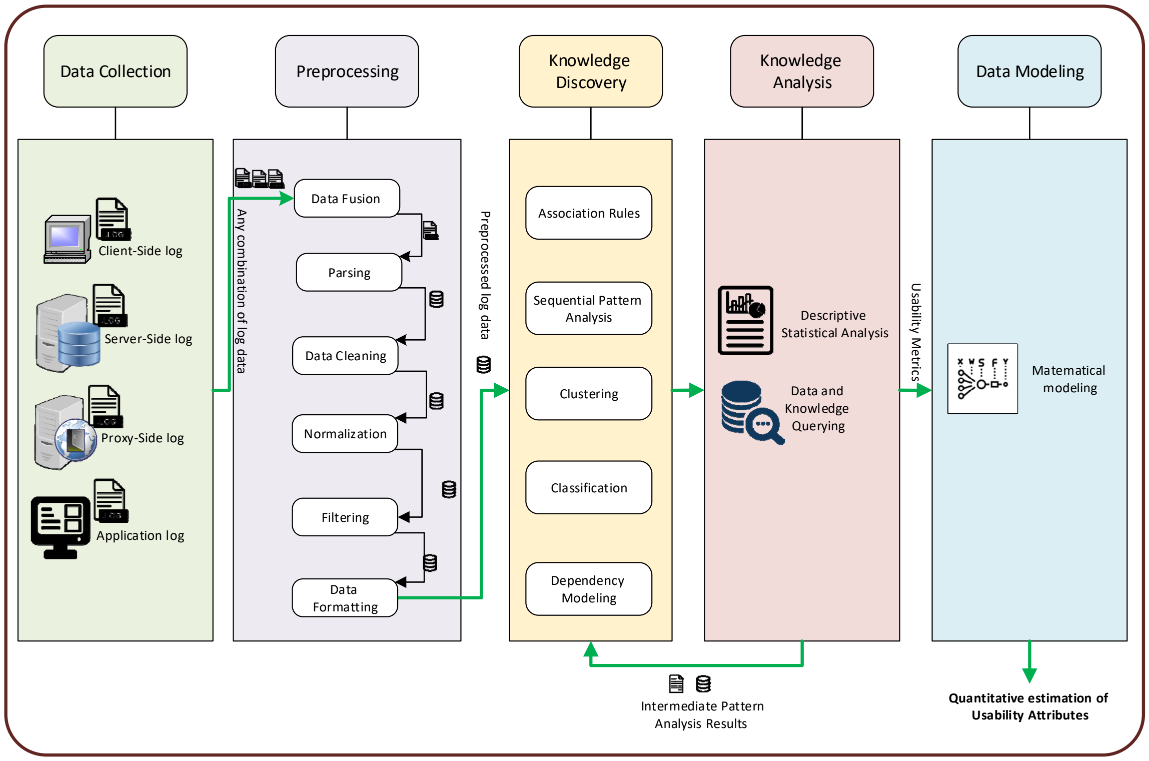


Figure 19: Proposed methodology for automatic usability evaluation using log files

- Knowledge Discovery is the third step that involves the knowledge extraction process, which entails performing a series of operations and processes on the log files to calculate the value of various usability metrics. Similar to Step 3 in web usage mining, these operations can be a combination of methods categorized into association rules, sequential pattern recognition, dependency modeling, clustering, and classification. The proposed methods for knowledge discovery of this study is described in Section 05.4

- After extracting the necessary information for calculating desired metrics, the next step is Knowledge Analysis in order to interpret this information and use it to derive the numerical value of the metrics. For instance, if the goal is to determine the "time on tasks" metric, in the knowledge discovery step (Step 3), task models can be automatically extracted or predetermined by the usability expert. Then, all the actions performed in the system can be categorized and labeled according to these tasks. However, it is the responsibility of the knowledge analysis step to calculate the duration time of each task instance using a database query and then interpret these values and convert them into a single number as the time on task value. This can be achieved through various methods such as considering the maximum time, the mode value in reported times, or the average time after removing outliers (Figure 20). The knowledge analysis methodology is elaborated in Section 5.4.1

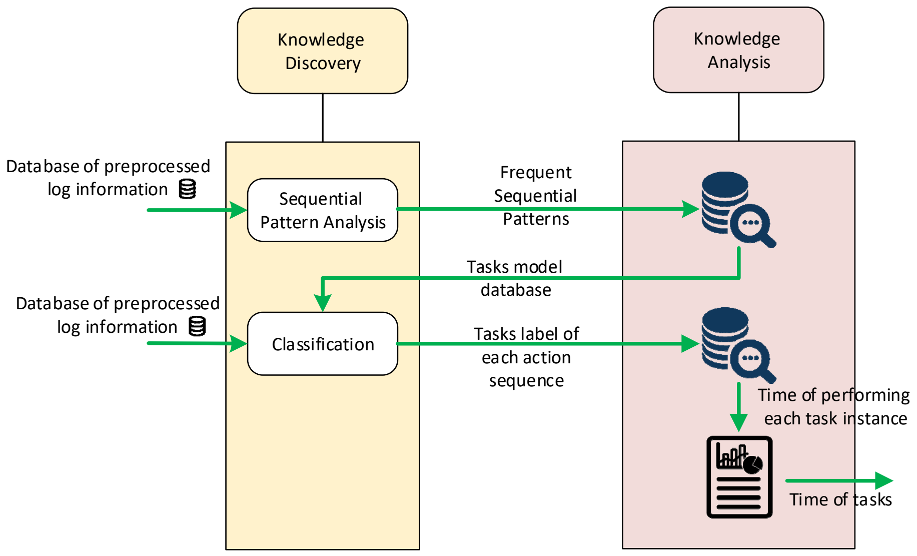

Knowledge Discovery
Knowledge Analysis
Database of preprocessed log information
Sequential Pattern Analysis
Frequent Sequential Patterns
Tasks model database
Database of preprocessed log information
Classification
Tasks label of each action sequence
Time of performing each task instance
Time of tasks

Figure 20: The procedure of calculating the "time on task" metric

Finally, the last step is Data Modeling, which receives the value of different usability metrics and determines the value of usability attributes such as efficiency, effectiveness, learnability, memorability, and user satisfaction. This mathematical model can be obtained by various methods from completely manually by heuristic and expert knowledge to completely automatically using various machine learning methods such as regression analysis, genetic algorithms, neural networks, and Bayesian networks. This dissertation does not focus on the modeling of usability attributes. Therefore, it concludes its scope at the fourth stage of the presented framework.

5.3 Data preparation

From all possible data preprocessing steps explained in Section 5.2, the following steps have been taken for log data preprocessing and preparation:

1) Remove all false visiting records in the log file
    a. Remove duplicate entries.
    b. Remove all records with incomplete or nonstandard HTTP headers: Because bots may not always provide complete HTTP headers, or use non-standard headers, we can check for inconsistencies or missing headers in the log entries to identify bots or crawlers.
    c. Some bots and crawlers respect the rules set in the "robots.txt" file or the "robots" meta tag on our website. Therefore, we can simply recognize records related to these bots and remove them.
    d. Identify and remove all visits from known suspicious IP addresses.
    e. Remove all single interaction visits. A single interaction visit indicates that visitor is not finding what they expected or that the page did not engage or fulfill their needs, so they abandon the website without performing any task.
    f. Remove requests for non-web page resources like images, stylesheets, scripts, or video files. These resources are often not directly relevant to our aim for user path analysis, and they can clutter the data and make analysis more complex. However, there may be cases where we want to analyze

the paths leading to specific resources, so we may keep those relevant records.

2) IP address-based segmentation: Grouping log file entries into separate sets of visitor records based on users' IP addresses.
3) Session identification and reconstruction which involves the following steps:
    a. Arranging all records in chronological order according to their visit times.
    b. Detecting and extracting customer sessions from the visitor record collection. If the time gap between two consecutive visiting records is more than the specified time threshold (20 minutes), they are recognized as different customer sessions.

Table 19 presents the refined dataset, after the removal of false visit records and non-web page entries.

Table 19: Summary of dataset after preprocessing steps

Number of Records	35,766
Number of Sessions	2103
Number of Unique IP address	701
Number of pages per visit	17
Average Session Time	716

5.4 Knowledge Discovery

To derive usability metrics and measurements from the recorded log file data, several steps are necessary. Following log file preprocessing and the extraction of web request streams within each session, the initial phase involves discerning the user's intended tasks for each session. This scenario aligns with unsupervised learning problems, with sequential data. The primary objective is to assign task labels to the relevant segments of the web request streams. An assumption is that task models have been pre-established. In other words, for each task, the sequence of required web requests from initiation to completion has been specified. However, in situations where task models are not available, it becomes necessary to extract them from the log file by recognizing frequent patterns.

A two-step hierarchical clustering approach has been introduced to achieve this goal (Figure 21). In the initial phase, the focus lies in identifying sessions devoted to a single task with a reasonable rate of task execution errors. This step enables the creation of a labeled dataset of web request sequences (or page view sequence) for each task that will later help to train the model in the next step.

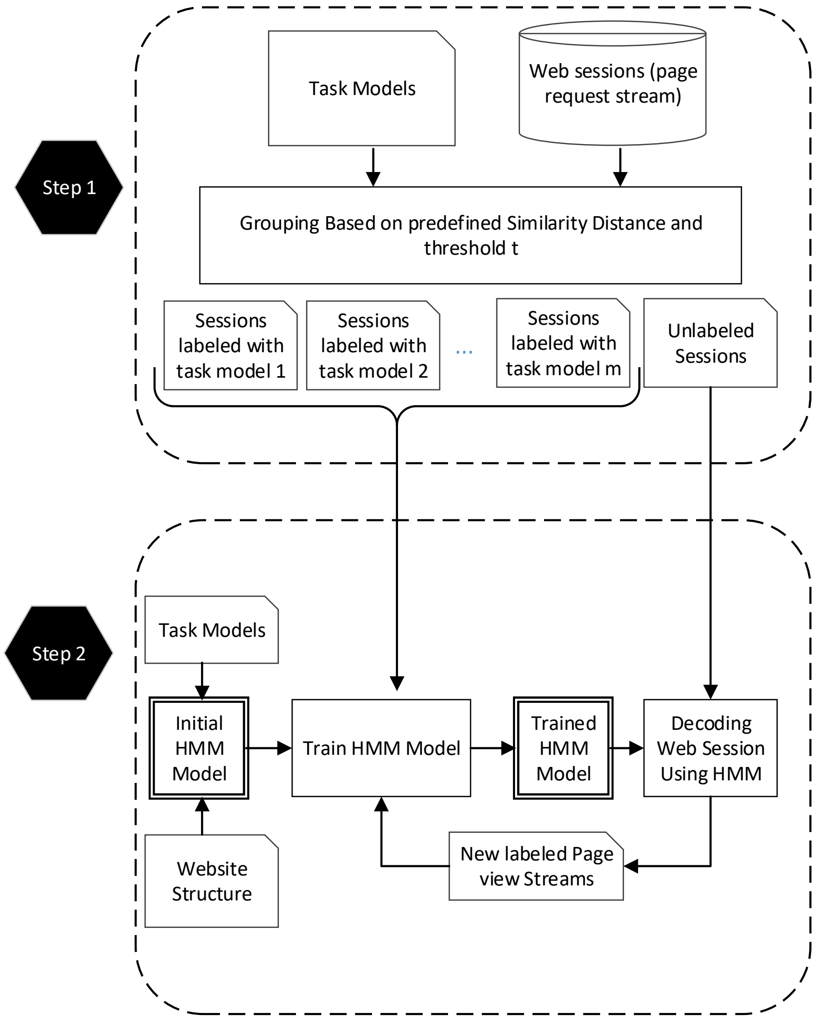

Step 1
Task Models
Web sessions (page request stream)
Grouping Based on predefined Similarity Distance and threshold t
Sessions labeled with task model 1
Sessions labeled with task model 2
...
Sessions labeled with task model m
Unlabeled Sessions
Step 2
Task Models
Initial HMM Model
Train HMM Model
Trained HMM Model
Decoding Web Session Using HMM
Website Structure
New labeled Page view Streams

Figure 21: two-step hierarchical clustering approach for extracting page view sequence related to each task.

The second step leverages Hidden Markov Models (HMMs) to cluster the remaining unlabeled sessions. First, we construct an initial HMM model for each task based on its predefined task model and the website's structure. Subsequently, these models are trained using the labeled sequences from the first step. Further details regarding the creation of this model are explained in Section 5.4.1.

In the final step, the HMM model is utilized to cluster sessions that remained unlabeled in the preceding stage. To refine the model during the clustering of new sessions and provide an incremental learning method, sessions are grouped into batches of 100.

Following the clustering of each batch, the freshly labeled data are employed to retrain and update the HMM models.

5.4.1 Step 1: Grouping based on Similarity Distance

In the approach represented in Figure 21, the initial step involves clustering sessions into pre-defined task models. During this phase, the focus is solely on identifying sessions that bear a strong resemblance to one of these task models. Hence, it's likely to pinpoint sessions where users predominantly engage in a single task, and the occurrence of errors in these sessions is relatively minimal. This enables confidence in assigning a specific task label to such sessions. To facilitate this process, the selection of a suitable distance metric becomes crucial. In this context, three distinct distance measures have been taken into account**.**

1) **Common Subsequences**:

This metric, which is borrowed from Wang et al. (2013) and Inspired by the Jaccard Coefficient (Levandowsky & Winter, 1971), involves identifying shared subsequences of a specific length (k) between two streams of page views. A page view stream can be defined as a sequence P, represented as $P = (p_1, p_2, \dots, p_n)$, where p_i represents the i$^{\text{th}}$ page in the sequence. To formalize this, T_k is introduced as the collection of all possible k-grams (consecutive sequences of k pages) within sequence P: $T_k(P) = \{k\text{-}gram | k\text{-}gram = (p_i p_{i+1} \dots p_{i+k-1}), i \in [1, n+1-k]\}$. Ultimately, the distance between two sequences is calculated based on the count of shared subsequences between the two sequences. we establish the distance between sequences P1 and P2 as follows.

$$D_k(P_1, P_2) = 1 - \frac{|T_k(P_1) \cap T_k(P_2)|}{|T_k(P_1) \cup T_k(P_2)|}$$

2) **Levenshtein distance (Edit Distance)**

This is a string metric for measuring the dissimilarity between two sequences. It quantifies the smallest number of necessary single-character modifications, (insertions, deletions, or substitutions), required to transform one string into another. Originally developed for assessing string similarity, the Levenshtein Distance can also be effectively employed in the context of other sequences, such as DNA sequences, page view streams or click streams. Kumar, Vibha, & Venugopal (2016) have leveraged a modified version of the Levenshtein distance to cluster web sessions. This distance can be formulized as follows:

$$D_{lev}(P_1, P_2) = \begin{cases} |P_1| & if\ |P_2| = 0, \\ |P_2| & if\ |P_1| = 0, \\ D_{lev}(tail(P_1), tail(P_2)) & if\ head(P_1) = head(P_2) \\ 1 + \min \begin{cases} D_{lev}(tail(P_1), P_2) \\ D_{lev}(P_1, tail(P_2)) \\ D_{lev}(tail(P_1), tail(P_2)) \end{cases} & otherwise \end{cases}$$

3) McBride Quimby Shih(MQS) Distance

This metric is described by McBride, Quimby, & Shih, (2005) and is an adaptation of the Levenshtein Distance, specifically tailored for its application in path analysis. This article introduces two key modifications to enhance its relevance.

In the first adjustment, the authors highlight that distinctions in shorter paths (e.g., AA and AB) carry more significance than those in longer paths (e.g., two paths with 20 requests, differing in just one). To account for this, they normalize the computed distance by dividing it by the average length of the two paths (P1 and P2). This normalization effectively scales the calculated distance in accordance with the lengths of the paths. The second change is to consider the repetition between the nodes within a path. The authors emphasize that although the Levenshtein distances between path ABB and the two paths

ABCDCB and ABABAB might be identical, paths that predominantly traverse the same nodes inherently share a greater degree of similarity compared to those exploring significantly distinct sets of nodes. To account for this, an additional condition has been introduced for the insertion, deletion, and substitution of nodes. Specifically, if the symbol to be added, removed, or replaced exists in both input strings, the edit cost is set at one. However, if the symbol is exclusive to one of the original strings, the edit cost is elevated to two.

5.4.2 Step 2: Clustering Using an HMM model

Hidden Markov Models (HMMs) are used in different areas where the aim is to figure out a sequence of data that we can't see directly. Instead, we can observe other data that rely on this hidden sequence; it is like solving a puzzle where we can only see some pieces, but we need to guess and piece together the whole picture.

HMM is a well-established and extensively utilized statistical method for characterizing spectral properties in various domains. It offers notable advantages such as a short algorithm and high detection accuracy and has been successfully applied to various domains such as signal processing, natural language processing, image processing, pattern recognition, and web mining.

For example, Xia & Tingjie (2009) proposed an intelligent predicting model based on the hidden Markov model (HMM), which mines the latent information requirement concepts that exists in the customer visiting path and makes business information predicting decisions. Moreover, classification of visitors based on click-streams is a reasonably well explored area. Markov chains and Hidden Markov models (HMMs) are commonly applied to model click data. Scott & Hann (2006) classified visitor sessions into one of predefined categories, namely decisive shoppers, deliberators, and never buyers, with a nested HMM. In this approach, the higher level HMM models visit across sessions, while the nested lower level HMM captures clicks in a given session. HMMs are also used by Ypma & Heskes (2003) to categorize web-pages. Further, they clustered users based on the observed click-streams.

In another study, Cao, Qiao, & Lyu (2017) proposed an anomaly detection system for web log files, which adopts a two-level machine learning algorithm. This paper first preprocessed the log files by applying decision tree algorithms based on patterns by rule set. The decision tree model classifies normal and anomalous data sets. Then, it models the normal data set by using HMM to build a model that can represent normal data status as the abnormity detector.

5.4.2.1 HMM Description

A Markov chain is a valuable tool in situations where we need to calculate the likelihood of a sequence of observable events. However, in numerous scenarios related to web mining and log file analysis, the events of interest are concealed or not directly observed. For instance, when examining log files, we typically don't directly observe user behaviors or specific actions. Instead, we observe log entries, and we need to deduce the underlying user actions or patterns from these log entries. In this context, we refer to the user actions or behaviors as "hidden" because they are not directly observable in the log files. (Xia & Tingjie, 2009).

A hidden Markov model (HMM), proposed first by Baum and his colleagues in the late 60s, is a stochastic process built upon the foundation of a Markov chain. It provides a framework for discussing both observable events (such as log entries in the input data) and unobservable events (like underlying user behaviors or actions) that are considered as influencing factors in the probabilistic model. The main idea is that an HMM is a finite model designed to represent the probability distribution across an infinite array of potential sequences (Eddy, 1996).

An HMM is specified by the following components λ =(Q,A,O,B, π)

- Q: a set of n states in the model. The individual states are labeled as {1,2,…,n} and denote the state at time t as q_t.
- A: a transition probability matrix, $A = \{a_{ij}\}$ where each a_{ij} represents the probability of moving from state i to state j, s.t. $\sum_{i=1}^{n} a_{ij} = 1$

$$a_{ij} = P[q_{t+1} = j|q_t = i]$$

- O: a sequence of m distinct observation symbols per state. The individual symbols are denoted as $\{v_1, v_2, \ldots, v_m\}$
- B: a sequence of observation likelihoods $B = \{b_i(k)\}$, in which

$$b_i(k) = P[o_t = v_k|q_t = i] \qquad 1 \leq k \leq m$$

  This is also called emission probabilities, each expressing the probability of an observation o_t being generated from a state i.
- The initial state distribution $\pi = \{\pi_i\}$, in which $\pi_i = P(q_1 = i)\ 1 \leq i \leq n$.

  π_i is the probability that the Markov chain will start in state i. Some states j may have πj = 0, meaning that they cannot be initial states. Also$\sum_{i=1}^{n} \pi_i = 1$

Based on a tutorial by Rabiner (1989) hidden Markov models should be defined by three core challenges or fundamental problems:

- Problem 1 (Likelihood): Given an HMM $\lambda = (A, B, \pi)$ and an observation sequence O, determine the likelihood $P(O|\lambda)$.
- Problem 2 (Decoding): Given an observation sequence O and an HMM $\lambda = (A, B)$, discover the best hidden state sequence Q.
- Problem 3 (Learning): Given an observation sequence O and the set of states in the HMM, learn the HMM parameters A and B.

In this specific scenario, the challenges primarily align with Problem 3 (Learning) before transitioning into Problem 2 (Decoding).

5.4.2.2 Problem Formulation

As previously mentioned, in order to define the Hidden Markov Model (HMM) for a given problem, we must specify five essential components denoted as $\lambda = (Q, A, O, B, \pi)$.

- **States of the model (Q):**

Within this model, a distinct part is established for each task. Consequently, each step of the task's execution introduces a state to the model. Figure 22 represents the part of model related to task t which involves N(t) steps.

In cases where the number of steps required to complete a task becomes considerably large, potentially complicating the model, it is possible to group steps together. This involves defining a step to encompass multiple page views, instead of assigning a single step for each individual page view.

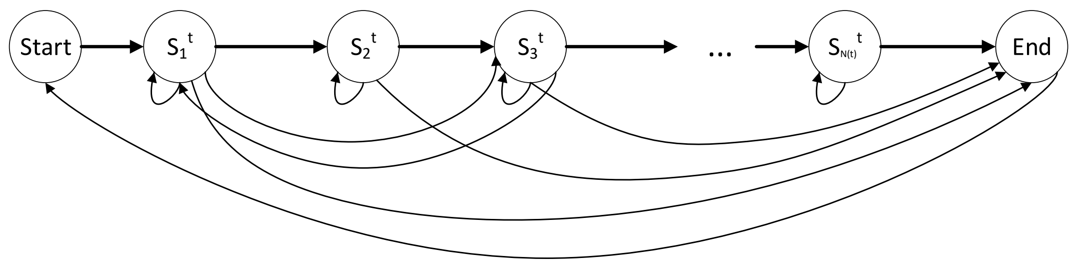


Figure 22: The part of model related to task t which involves N(t) steps

Furthermore, to account for scenarios in which users engage with the website randomly, without a specific task in mind, a new part to the model is introduced that enables browsing through the website's primary pages. Figure 23 shows this part of the model. For the selection of the top K pages, we identify the most frequently visited page (or category) on the website based on information extracted from the log file.

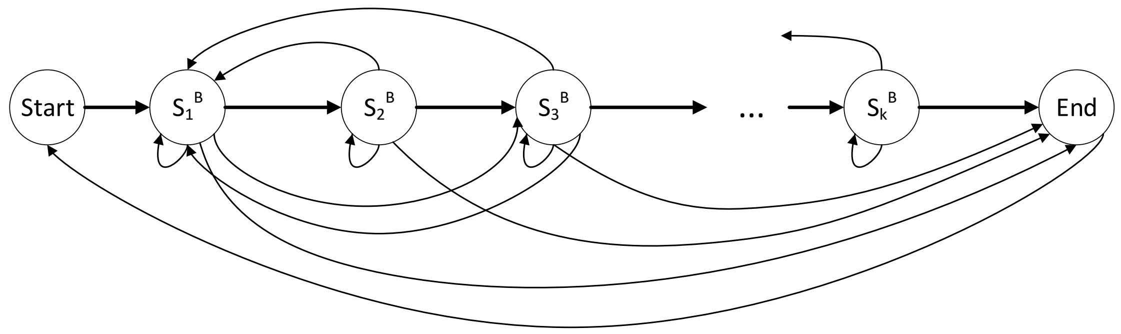


Figure 23: part of the model that relates to random website browsing

An initial state and a final state have also been added within the model to specify the start and end of tasks. Consequently, the set Q is defined as follows:

$$Q = \{Start, END, S_1^1, \dots, S_{N(1)}^1, \dots, S_1^{N_{task}}, \dots, S_{N(N_{task})}^{N_{task}}, S_1^B, \dots, S_K^B\}$$

Given N_{task} as the total number of tasks on the website and N(t) as the number of steps necessary to complete task t, the count of model hidden states (N) is determined as follows:

$$N = K + 2 + \sum_{t=1}^{N_{task}} N(t)$$

- **Observations (O):**

In this problem, the observations are the records stored in the log file. Considering that the focus is on requests related to page views, the number of O states is equal to the total count of website pages. However, given that websites frequently feature a very large number of web pages, they can be effectively grouped and categorized, streamlining the observation process according to the page categories requested.

- **Transition probability matrix (A):**

In the process of establishing the transition probability matrix, the HMM model can be acquired through data from the preceding step. However, there may be cases where some tasks lack any instances, or the available labeled sessions are insufficient to effectively learn the HMM model. Consequently, all tasks are classified into two categories. The first group comprises tasks that have obtained more than R labeled sessions (or page sequences). The second group encompasses tasks for which the count of identified page sequences falls below R (Figure 24). In this context, the threshold value R is set to 30, taking into account both the number of sessions in the log file and the complexity of the task.

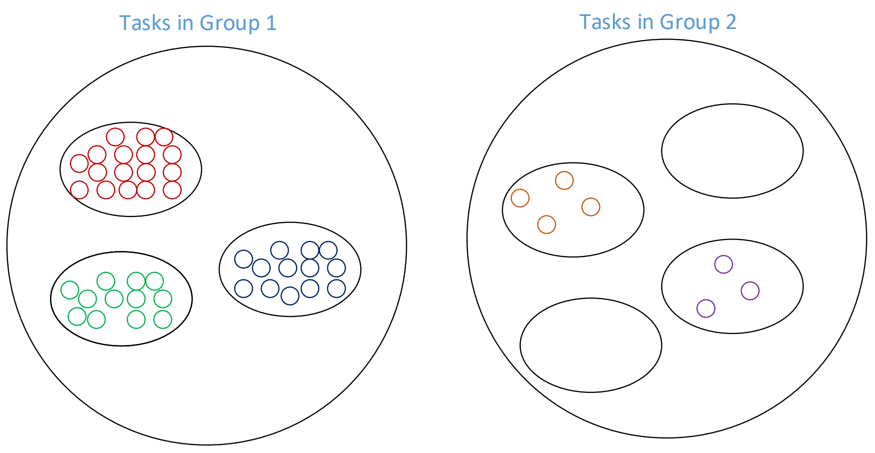


Figure 24: All tasks will be categorized into two groups based on whether the number of labeled sessions for each task is below or above the specified threshold value, denoted as R

The proposed approach involves initially constructing a model for tasks in the first group. Subsequently, utilizing the transition probabilities obtained from group 1, the initial model for the tasks in the second group is established. This method stems from the understanding that, although the complexity of different tasks on a website may be different, obtaining information about a few tasks can serve as a valuable foundation for estimating the complexity or ease of performing other tasks. Hence, by modeling a set of tasks and determining the transition probabilities between their states, we gain the capacity to predict various transitions in other tasks, such as the likelihood of proceeding down the correct path, employing the back button and revisiting previous pages, or making wrong choices.

When establishing the model for tasks in group 1, it is essential to keep in mind that the initial data selection prioritizes samples with fewer errors. Consequently, certain errors may not be prominently evident in the data, but it is important not to assume their probability is zero. Figure 22 illustrates all potential transitions. In the context of task t:

- The transition from S_i^t to S_{i+1}^t represents the moment when the user selects the correct path during the ith step of task execution.
- The transition from S_i^t to S_{i-1}^t is associated with the user's choice to either utilize the browser's back button or take an incorrect path during the ith step, leading to a return to the previous step.
- The loop transition from S_i^t to itself encompasses situations where the user employs the browser's refresh button in the i-th step, encounters errors while completing a page's form, leading to a page reload, or remains stuck in the same step due to choosing the wrong path and being unable to proceed to the next step of the task.
- The transition from S_i^t to S_j^t, where j < i-1, occurs when the user mistakenly goes back or utilizes the back button. In some cases, client-side or proxy-side caching may cause the absence of server requests for several pages in between. The likelihood of transitioning to states further from i decreases as the gap between j and i widens. For the sake of model simplicity, we restrict the analysis of backward transitions to a maximum of three steps.
- The transition from S_i^t to S_j^t, with j>i+1, occurs when the user inadvertently bypasses a task step or when intermediate static pages are available in either the user's or proxy server's cache, therefore they are not requested from the server. As in the previous scenario, the probability of transitioning to states farther from i decreases as the gap between j and i increases. For the sake of model simplicity, we have limited our analysis to forward transitions spanning a maximum of three steps.

- The transition from any state, except for the state $S_{N(i)}^{t}$, to the final state signifies the abandonment of an incomplete task.

For all the aforementioned transitions:

- For tasks in the first group, where an adequate number of samples is available within their respective clusters, we can derive transition probabilities from the dataset. However, given a selective approach in favor of web sessions with low error rates within each cluster, certain transitions may never appear in the available sample sequences. Therefore, it is imperative to refine the model to preserve these transitions and accommodate their potential occurrence in future scenarios. To address this issue, a default percentage value is introduced, denoted as α, to be added to all transition values within the model. In our experimental configuration, we set α to 1%. Subsequently, all updated values are normalized to ensure that the sum of transition probabilities for each state equals 1.
- For tasks belonging to the second group, the average transition probabilities of the first group tasks are calculated and then normalized. These normalized averages are then assigned to equivalent transitions within the second group. For instance, the probability of transitioning from S_{i}^{t} toS_{i+1}^{t} in all tasks within the second group is the average of probabilities obtained from S_{i}^{t} to S_{i+1}^{t} in all tasks from the first group. The same approach is applied to determine probabilities for actions such as choosing the wrong path or navigating backward.

- **Observation likelihood (B)**

In this section, we must specify what observations will happen in each state, or in other words, what is the probability of requesting each page from the server in each state. During the i-th step of task t, the user has the option to progress in the correct direction, resulting in a request for the associated page of that particular step, denoted as P_i^t. Additionally, the user may choose an incorrect path, leading to visiting other pages.

To determine the values of the matrix B, again for Group 1 and Group 2 tasks two distinct approaches are employed.

- For every task (t) within the first group, we employ the data in its cluster to train the respective segment of the HMM. As previously discussed, it's imperative to adjust the acquired values post-learning to account for potential observations that, while feasible, did not appear within the available sample sequences in cluster t.

  In state i of task t ($\boldsymbol{S_i^t}$), for each page (P) that remained unvisited, the distance between P_i^t to P within the website's structure is employed as a pivotal criterion for establishing the probability of transitioning to page P.

$$b_{\boldsymbol{S_i^t}}(\mathrm{P}) = \frac{\alpha}{distance(P_i^t, P)}$$

  Following the assignment of new values, all the acquired values are subjected to normalization, ensuring that their cumulative sum equals one.

$$b_{\boldsymbol{S_i^t}}(\mathrm{P}) = \frac{b_{\boldsymbol{S_i^t}}(\mathrm{P})}{\sum_{\mathrm{P} \in \{\boldsymbol{all\ pages\ of\ website}\}} b_{\boldsymbol{S_i^t}}(\mathrm{P})}$$

- For tasks within the second group, the average error percentage is calculated across all steps of tasks in the first group, denoted as ($\bar{E}$).

$$\bar{E} = \frac{\sum_{t \in first\ group} \sum_{i=1}^{N(t)} 1 - \dfrac{number\ of\ visiting\ P_i^t\ in\ S_i^t}{numbe\ of\ all\ pages\ visited\ in\ S_i^t}}{\sum_{t \in first\ group} N(t)}$$

Subsequently, in all steps of the tasks in this group, the probability of encountering P_i^t is set to 1 - $\bar{E}$, while the probability of visiting other pages is determined in relation to their distance from P_i^t in a manner that ensures the sum of their probabilities equals $\bar{E}$.

- **Initial state distribution (π)**

To derive the values of π, the conversion rates for each task are calculated. Given that conversions typically occur at the final stage of a task and that, in most instances, the last page of a task directly corresponds to the task itself, the assumption that task completion is primarily marked by the request for the last page associated with that task is simplified. Consequently, the π value corresponding to the initiation of task t is determined based on the number of times $\boldsymbol{P_{N(t)}^t}$ is requested and the total requests for the final pages of all tasks. Moreover, the possibility of browsing the website (π($\boldsymbol{S_1^B}$)) can be determined according to the nature of the website, its value is considered to be 10% and the π value for other states which are not the first state of any task is considered zero.

$$\pi(S_i^t) = \begin{cases} \dfrac{\#\left(\boldsymbol{P_{N(t)}^t}\right)}{\sum_{\boldsymbol{i=1}}^{\boldsymbol{Ntask}} \#(\boldsymbol{P_{N(i)}^i})} \times (\mathbf{1} - \pi(\boldsymbol{S_1^B})) & \boldsymbol{i = 1} \\ \mathbf{0} & \boldsymbol{i > 1} \end{cases}$$

5.5 Knowledge Analysis

As depicted within the framework, the subsequent phase encompasses knowledge analysis with the aim of deriving usability metrics. This section offers a comprehensive and explicit explanation of the approach employed for data analysis and the acquisition of usability metrics from log files. To achieve this aim, it begins by introducing the parameters that need to be extracted from the log file and presents the proposed extraction methodology for each. Subsequently, numerical formulas for several specific metrics are provided.

5.5.1 Metrics calculation methods

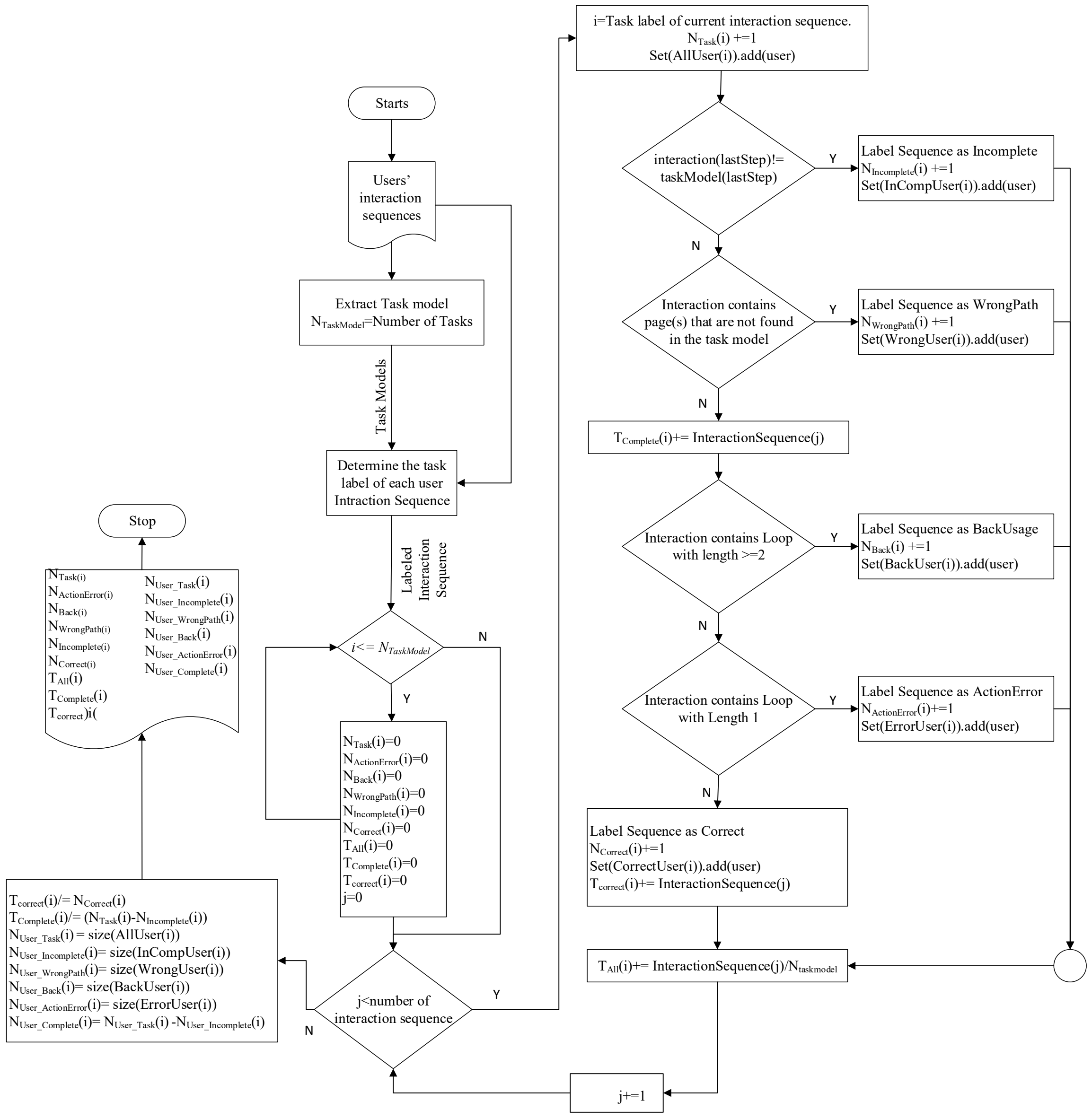
According to the metrics and their corresponding definitions outlined in Table 3, it is necessary to analyze the log file and extract some fundamental information from it, which will serve as the foundation for computing the majority of the metrics in Table 3. A part of the data that needs to be extracted from the log file is explained in the table below and a name has been assigned to each of them, which will be used later in the program and flowchart as their variable name.

Table 20: A sample of information needs to be extracted from the log file

Description	Variable Name
Number of Tasks Models	$N_{TaskModel}$
The number of times that a task has been executed in the system.	$N_{Task(i)}$
The Number of times a task started to be performed but never ended.	$N_{Incomplete(i)}$
The number of times the user completed a task successfully but got confused in the middle of the way and returned to the main task routine after taking a few wrong steps.	$N_{WrongPath(i)}$
The Number of times a task was performed and completed successfully but the user used the back button of the browser.	$N_{Back(i)}$

The Number of times a task was performed and completed successfully but the user tried several times to complete several specific actions.	$N_{ActionError(i)}$
The Number of times a task is performed and completed successfully without error.	$N_{Complete(i)}$
The Number of users that performed a particular task.	$N_{User\_Task(i)}$
The Number of users that started a particular task completely and without error	$N_{User\_Incomplete(i)}$
The Number of users that performed a particular task successfully but got confused in the middle of the way and returned to the main task routine after taking a few wrong steps	$N_{User\_WrongPath(i)}$
The Number of users that performed and completed a task successfully but user used the back button of the browser.	$N_{User\_Back(i)}$
The Number of users that performed and completed a task successfully but user tried several times to complete several specific actions.	$N_{User\_ActionError(i)}$
The Number of users that performed a particular task completely and without error	$N_{User\_Complete(i)}$
The Average time of performing all task instances with or without Error	$T_{All}(i)$
The Average time of performing all task instances without Error.	$T_{correct}(i)$
The Average time of performing all task instances that have been completed successfully with or without Error.	$T_{Complete}$ (i)
The number of differences between the interaction sequence performed by the user and the task model(i) in a particular execution (instance(j))	$N_{error(i)(j)}$
The Number of users that performed a particular task completely and without error for the first time	$N_{First\_Correct(i)}$
The Number of users that performed a particular task for the first time and had an error	$N_{First\_Error(i)}$
The Number of users that performed a particular task for the first time.	$N_{First\ (i)}$
The Number of users that performed a particular task for the first time and complete it with or without error	$N_{First\_Complete\ (i)}$

Figure 25 represents a flowchart for extracting the information outlined in Table 20. Various methods can be applied in different steps of this flowchart. For example, the task model can be estimated by applying sequential pattern analysis, clustering, association rules, or a combination of them. Similarly, for determining the task label of each user interaction sequence, different classification methods can be employed.



Starts
Users' interaction sequences
Extract Task model
$N_{TaskModel}$=Number of Tasks
Task Models
Determine the task label of each user Intraction Sequence
Labeled Interaction Sequence
$i <= N_{TaskModel}$
Y
N
$N_{Task}(i)=0$
$N_{ActionError}(i)=0$
$N_{Back}(i)=0$
$N_{WrongPath}(i)=0$
$N_{Incomplete}(i)=0$
$N_{Correct}(i)=0$
$T_{All}(i)=0$
$T_{Complete}(i)=0$
$T_{correct}(i)=0$
j=0
j<number of interaction sequence
Y
N
i=Task label of current interaction sequence.
$N_{Task}(i)$ +=1
Set(AllUser(i)).add(user)
interaction(lastStep)!= taskModel(lastStep)
Y
N
Label Sequence as Incomplete
$N_{Incomplete}(i)$ +=1
Set(InCompUser(i)).add(user)
Interaction contains page(s) that are not found in the task model
Label Sequence as WrongPath
$N_{WrongPath}(i)$ +=1
Set(WrongUser(i)).add(user)
$T_{Complete}(i)$+= InteractionSequence(j)
Interaction contains Loop with length >=2
Label Sequence as BackUsage
$N_{Back}(i)$ +=1
Set(BackUser(i)).add(user)
Interaction contains Loop with Length 1
Label Sequence as ActionError
$N_{ActionError}(i)$+=1
Set(ErrorUser(i)).add(user)
Label Sequence as Correct
$N_{Correct}(i)$+=1
Set(CorrectUser(i)).add(user)
$T_{correct}(i)$+= InteractionSequence(j)
$T_{All}(i)$+= InteractionSequence(j)/$N_{taskmodel}$
j+=1
$T_{correct}(i)$/= $N_{Correct}(i)$
$T_{Complete}(i)$/= ($N_{Task}(i)$-$N_{Incomplete}(i)$)
$N_{User\_Task}(i)$ = size(AllUser(i))
$N_{User\_Incomplete}(i)$= size(InCompUser(i))
$N_{User\_WrongPath}(i)$= size(WrongUser(i))
$N_{User\_Back}(i)$= size(BackUser(i))
$N_{User\_ActionError}(i)$= size(ErrorUser(i))
$N_{User\_Complete}(i)$= $N_{User\_Task}(i)$ -$N_{User\_Incomplete}(i)$
$N_{Task(i)}$
$N_{ActionError(i)}$
$N_{Back(i)}$
$N_{WrongPath(i)}$
$N_{Incomplete(i)}$
$N_{Correct(i)}$
$T_{All}(i)$
$T_{Complete}(i)$
$T_{correct}$)i(
$N_{User\_Task}(i)$
$N_{User\_Incomplete}(i)$
$N_{User\_WrongPath}(i)$
$N_{User\_Back}(i)$
$N_{User\_ActionError}(i)$
$N_{User\_Complete}(i)$
Stop

Figure 25: Flowchart of Extracting a part of required information for usability metric calculation

Once the information has been extracted, the calculation formula for each metric can be proposed to determine its value using the available information. For example, the Time on Task and the Task Completion Time metrics can respectively be obtained by

$T_{All}(i)$ and $T_{Complete}(i)$. In addition, the Task Completion Rate metric represents the percentage of users who were able to successfully complete a task can be computed by:

$$Task\ Completion\ Rate = \frac{\mathrm{N_{User\_Complete(i)}}}{\mathrm{N_{User\_Task(i)}}} \times 100$$

The Error Rate is the total number of errors made by users while completing a task and is obtained by:

$$Error\ Rate = \frac{\sum_j N_{error(i)(j)}}{N_{Task(i)}}$$

Similarly, the Task Completion Rate on the First Attempt and the Error Rate on the First Attempt can be calculated by the following formula:

$$\text{Task Completion Rate on the First Attempt} = \frac{\mathrm{N_{First_{Correct(i)}}}}{\mathrm{N_{First(i)}}} \times 100$$

$$\text{Error Rate on the First Attempt} = \frac{\sum_{j \in First\ Usage} N_{error(i)(j)}}{\mathrm{N_{First(i)}}} \times 100$$

5.6 Experiments and Results

5.6.1 Clustering with Similarity Distance on Synthetic Data

The initial experiment was conducted to determine the optimal value of 'k' in the common sub-sequences method. Three distinct 'k' values were explored, 3, 4, and 6, tailored to the task lengths of the models. These values were assessed through clustering on a synthetic dataset comprising 10 unique tasks, each with an average page visit length of 10 and a standard deviation of 4. This dataset was generated for 2000 sessions, 1203 of which were single-task sessions. The primary aim of the initial clustering step was to identify and label these single-task sessions. Altogether, these sessions encompassed a total of 2901 tasks.

Table 21 provides a summary of the key details pertaining to this synthesized dataset.

Table 21: Synthesized dataset characteristics

Feature	Value
Number of records	43515
Number of sessions	2000
Number of users	400
Number of Taks	2901
Average Task Length	10 page
Average session time	717(s)

Upon computing the common sub-sequences distance for all sessions with respect to each task model, a label for each session was assigned based on its closest task model, provided that the distance value fell below the threshold value 0.3. As previously mentioned, a substantial number of sessions do not fall into any category at this stage and only those sessions exhibiting low errors and clearly associated with a single task are categorized.

To select the best 'k' value, we calculated precision, recall, and F-Score for each value. Figure 26 and Table 22 illustrate that with a larger 'k' value, the labeling becomes more stringent. This means that the precision is higher, so the labeled samples are correctly classified with greater confidence. However, this also leads to reduced sensitivity, causing the omission of a portion of samples that may be useful for subsequent model learning.

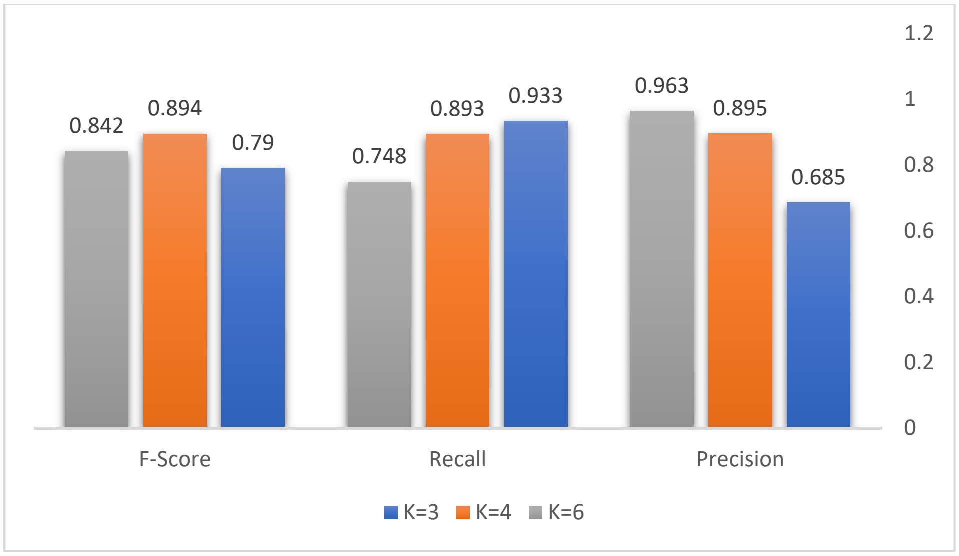


Figure 26: Clustering Results on Synthetic Data Using the Common Sub-sequences Method for k = 3, 4, and 6

Table 22: Precision, Recall, and F-Score Values for Various 'k' Values in the Common Sub-sequences Method

	Precision	Recall	F-Score
K=3	0.6847561	**0.93349958**	0.79001055
K=4	0.89508743	0.89359933	**0.89434276**
K=6	**0.96359743**	0.74812968	0.84230229

When working with sufficiently large log files, opting for a larger 'k' becomes feasible. However, in cases where the log file is relatively smaller, preserving accurate data becomes crucial. The website used in this dissertation is a creation aligned with the project's objectives and not an actual commercial site. As a result, the transaction volume, and subsequently, the log file size, are limited. Therefore, 'k=4' is chosen since it yields the highest F-score.

The next experiment aims to determine the most suitable distance measure. In this test, precision, recall, and F-Score is again calculated for all three methods. For the Common Sub-sequences method, a threshold value of 0.5 is used. In the case of Levenshtein, the Edit distance threshold is set to 4, while for MQS, the threshold is 0.4.

The choice of threshold values should be tailored to the complexity and length of the tasks. For example, with an average task length of 10, a value of 4 for Levenshtein or 0.4 for MQS is reasonable. However, if the average task length is 6, a smaller value should be chosen. The results of the test are presented in Figure 27.

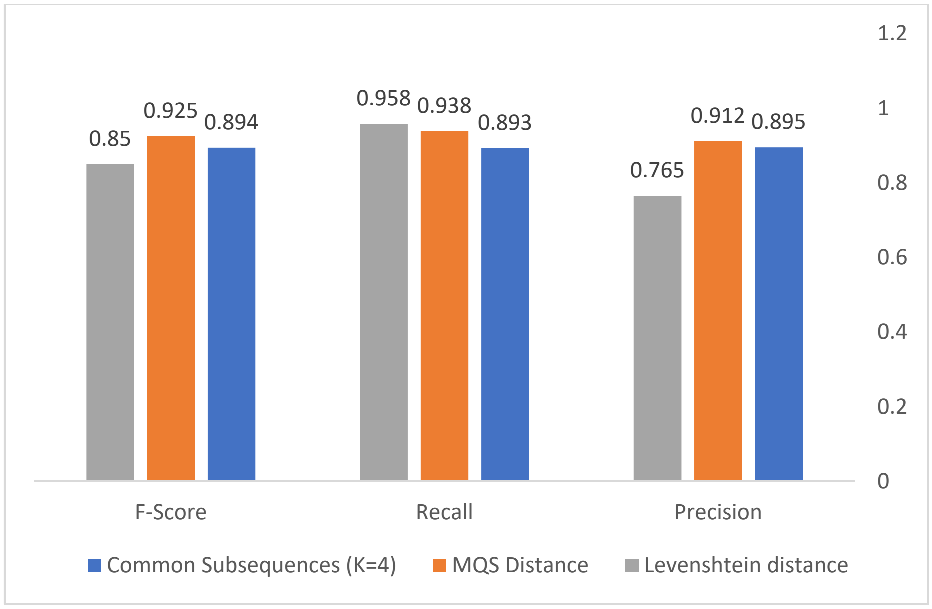


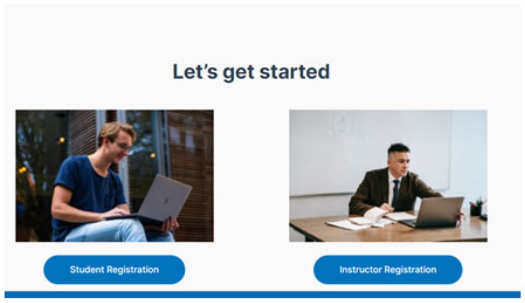
Figure 27: Clustering Results on Synthetic Data Using the Common Sub-sequences Method, Levenshtein and MQS Distance

As shown, the accuracy of Levenshtein is notably low. upon closer examination of its results, it became evident that Levenshtein performs significantly poor when applied to tasks of shorter length. This discrepancy arises from the ineffectiveness of the chosen threshold for shorter tasks. The solution to this issue lies in updating the threshold in accordance with the task's length, a problem that the MQS method has successfully addressed. The results show that among the three distance measures, MQS outperforms the other two.

5.6.2 Clustering with similarity distance on real data

For testing on real data, eight primary tasks were selected from the website, which encompass:

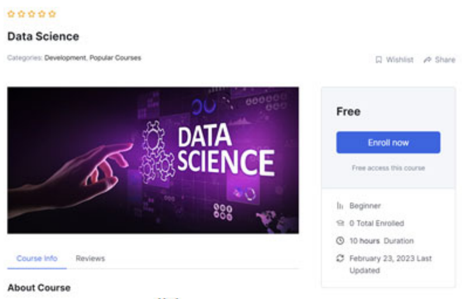
1. **Submit as a Student**: This task pertains to the creation of a student account on the website. (Figure 28 (a))
2. **Submit as a Teacher:** This task involves registering as a teacher on the website. (Figure 28 (a))
3. **Course Enrolment:** It encompasses the process of enrolling in one of the courses available on the website for learning. (Figure 28 (b))
4. **Take a Quiz**: This task is for participating in quizzes associated with specific lessons. (Figure 28 (c))
5. **Send a Message**: It revolves around sending messages to the website's administrator. (Figure 28 (d))
6. **Submit a Review**: This task involves writing reviews for courses that users have enrolled in. (Figure 28 (e))
7. **Create a Course**: This task is intended for teachers and involves the creation of new courses along with the uploading of their content. (Figure 28 (f))
8. **Edit the Profile**: This task centers on updating user account information, including actions like changing the cover photo, email address, or expertise details. (Figure 28 (g))


Let's get started

Student Registration

Instructor Registration

(a)


Data Science

Categories: Development, Popular Courses

Wishlist Share

Free

Enroll now

Free access this course

Beginner

0 Total Enrolled

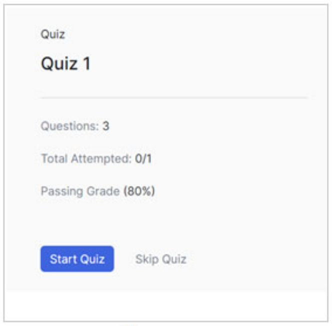
10 hours Duration

February 23, 2023 Last Updated

Course Info Reviews

About Course

(b)


Quiz

Quiz 1

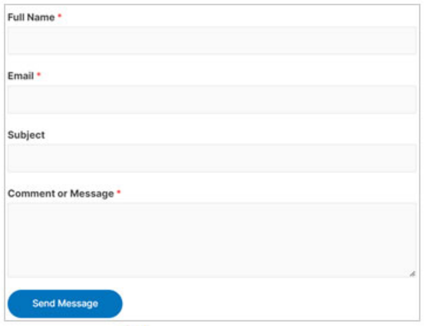
Questions: 3

Total Attempted: 0/1

Passing Grade (80%)

Start Quiz Skip Quiz

(c)


Full Name *

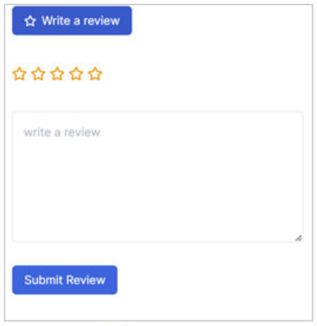
Email *

Subject

Comment or Message *

Send Message

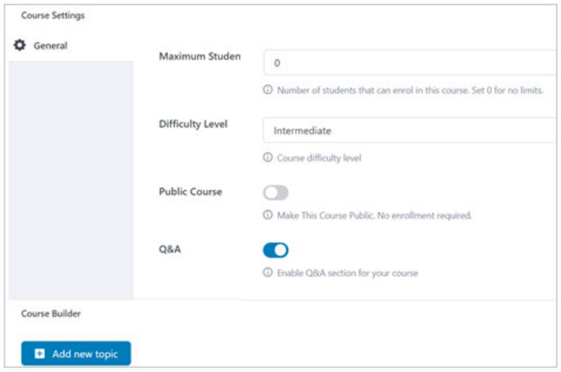
(d)


Write a review

write a review

Submit Review

(e)


Course Settings

General

Maximum Studen: 0

Number of students that can enrol in this course. Set 0 for no limits.

Difficulty Level: Intermediate

Course difficulty level

Public Course

Make This Course Public. No enrollment required.

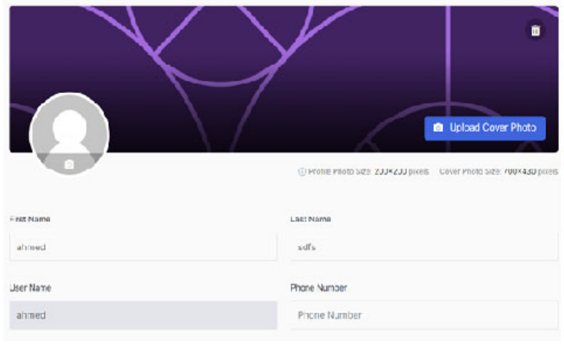
Q&A

Enable Q&A section for your course

Course Builder

Add new topic

(f)


Upload Cover Photo

Profile Photo Size: 200×200 pixels Cover Photo Size: 700×430 pixels

First Name: ahmed

Last Name: sdfs

User Name: ahmed

Phone Number: Phone Number

(g)

Figure 28: Sample Pages for 8 Selected Tasks

The Table 23 shows the number of main steps required to perform each task.

Table 23: The number of main steps required to perform each selected task

Tasks	Number of required steps
Submit as a Student	5
Submit as a Teacher	6
Course Enrolment	9
Take a Quiz	11
Send a Message	4
Submit a Review	8
Create a Course	15
Edit the Profile	5

Figure 29 illustrates the number of sessions labeled for each task using the MQS method on real data. An evaluation of the results underscores the initial clustering's success. For example, the limited number of instances related to recording reviews for lessons can be attributed to several factors. Firstly, there is a relatively low number of reviews submitted for all courses on the website. Additionally, this task is typically not carried out in isolation; most users tend to submit reviews after enrolling in a course or completing a section of the course.

Furthermore, the task ratio between "Submit as a Student" and "Submit as a Teacher" accurately reflects the real-world scenario, as the number of students registered on the website significantly exceeds the number of teachers. An additional sign of the model's effectiveness is the number of sessions detected for the last task. Since March 2023, 25 new courses have been added to the website, and this method successfully identified 16 sessions out of those 25 course creations, which is a notable achievement.

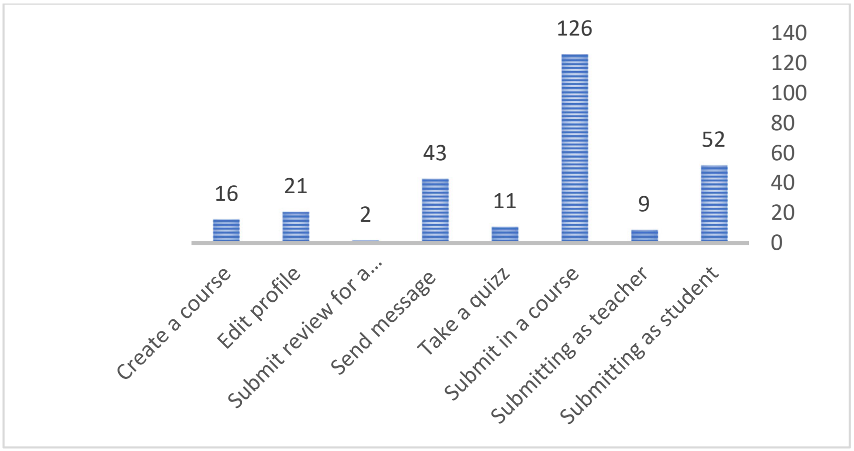


Figure 29: number of sessions clustered in each task category using the MQS method

5.6.3 HMM Model Clustering on Synthetic Data

To assess the performance of the HMM model, the synthesized dataset was initially used. Because in this data set, the true labels of each page view sequence are known, it provides a reference point for evaluating the model's accuracy. As mentioned earlier, the synthesized data set consists of ten distinct tasks, so the sequences related to these ten tasks have to be identified, along with the sequences related to random website browsing. The data generation process did not include an option to generate data for random website browsing. Consequently, no data was created or classified within this category, leading to its exclusion from the evaluation.

As evidenced by the confusion matrix, an evaluation reflects the model's commendable performance across most instances. However, it's important to acknowledge that the synthesized data lacks the complexities and challenges of real-world data, and the level of accuracy achieved here may not be directly translatable to real-world scenarios. Nonetheless, these results instill confidence in the effectiveness of the designed HMM model. Notably, the most significant misclassification occurred between tasks 7 and 1. Upon closer examination, we discovered that task 7 encompasses nine primary steps, while task 1 comprises ten steps, and they have two common subsequences of length 4. Given their high similarity, this misclassification is deemed reasonable and understandable.

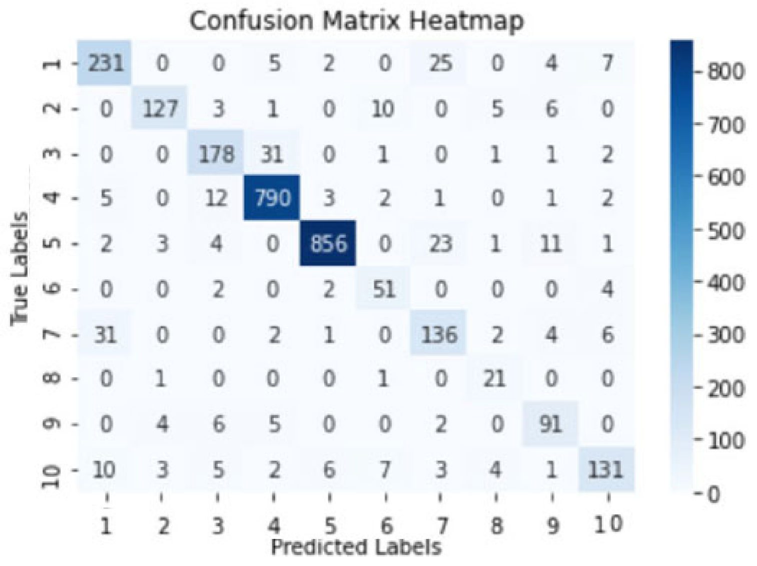


Figure 30: confusion matrix of clustering page view sequences into 10 task using HMM model

For a more comprehensive evaluation, Figure 31 presents precision, recall, and F-scores for the clustering method, individually calculated for each class.

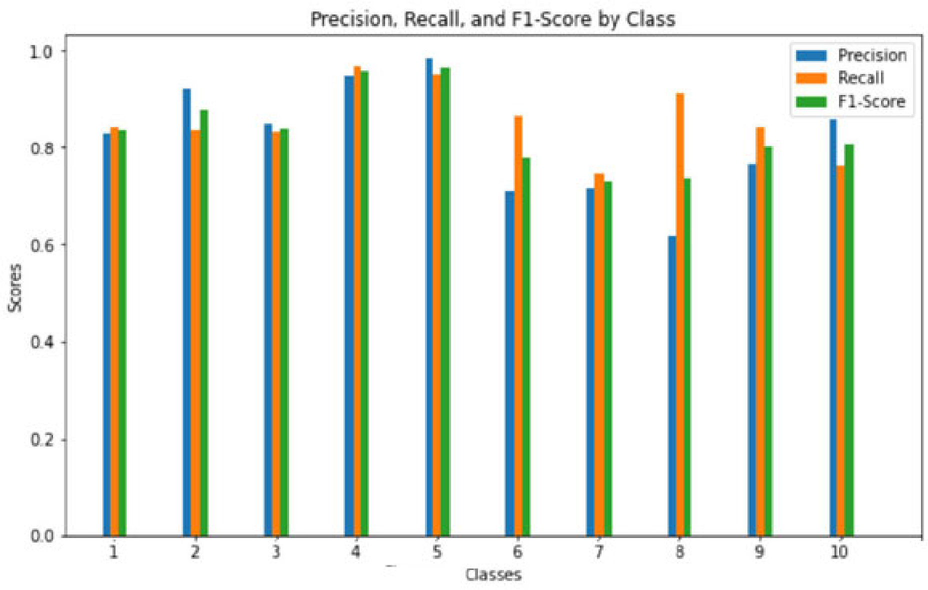


Figure 31: Precision, Recall and F-Score of each class of the clustered synthetic data using the HMM model

5.6.4 HMM Model Clustering on Real Data

To assess the results on real data, a silhouette plot was employed that created based on MQS distance, which measures the dissimilarity between page view sequences (Figure 32). A silhouette plot is a graphical representation used in data analysis and clustering to assess the quality of clustering results. It provides insight into the separation and cohesion of data points within each cluster. The silhouette score ranges from -1 to 1, where a high positive value (close to 1) indicates that the data point is well matched to its own cluster and clearly separated from other clusters. On the other hand, a negative value (close to -1) indicates that the data point may have been assigned to the wrong cluster. A value near 0 suggests that the data point is on or very close to the decision boundary between two neighboring clusters.

Given that MQS distance alone may not serve as a definitive classification criterion, we anticipated that the average silhouette score might not reach a very high value. Nonetheless, the achieved score remains within an acceptable range. Furthermore, this chart aids in evaluating the model's performance from multiple perspective.

The majority of the data points fall within the categories of website browsing and course enrollment. This occurrence can be attributed to the nature of the website, which, as an eLearning platform, naturally witnesses a substantial volume of searches on the website and page views of course information. It is not uncommon for users to conduct searches without proceeding to registration. Additionally, for simplification purposes, specific models were not defined for certain tasks, such as adding a lesson to the wish list or sharing a course link. As they lack distinct clusters, they may have been categorized under website browsing. This fact might have contributed to a lower silhouette score for this cluster.

The second-largest cluster pertains to course enrollment, which is logically expected to have a higher number than other designated tasks, given that the primary objective of visiting an eLearning website is to enroll in a course. Furthermore, since there are shared steps between course enrollment, taking a quiz, and registering a review, there exists the possibility of occasional misclassification among these classes as well. The sizes of the other clusters align with expectations. For instance, registering as a teacher or creating a lesson exhibits a lower count, as evidenced in the figure, resulting in smaller cluster sizes for these categories.

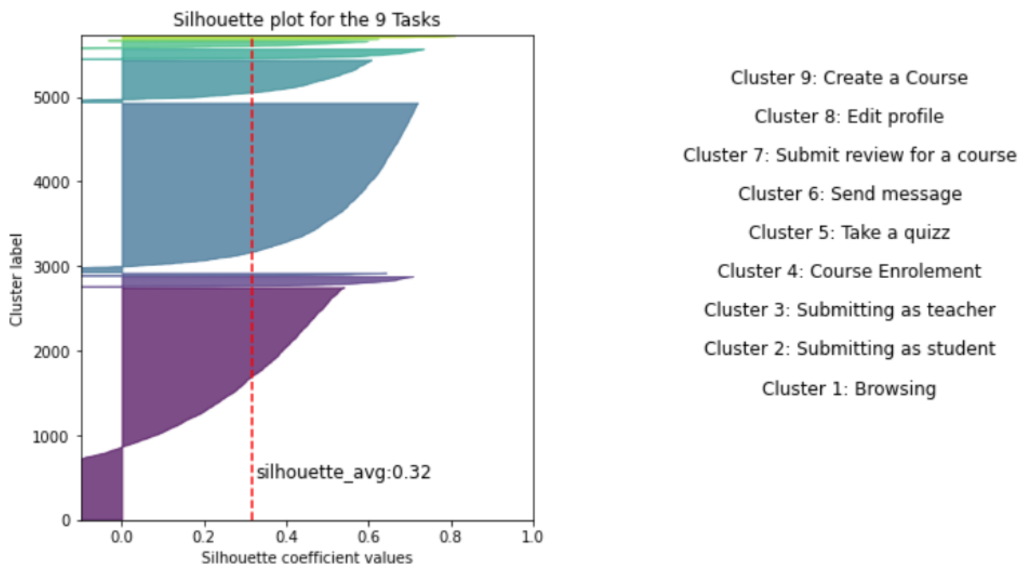


Figure 32: Silhouette plot for clustering real data with HMM model

To ensure the model's accuracy, a distinct silhouette plot was generated for the month of March (Figure 33). During March, the website was newly established, resulting in a reduced appearance in search results. Furthermore, a set of specific tasks on this month was defined; a group of students were asked to complete those tasks on the website.

Consequently, due to the lower volume of miscellaneous visits and more uniform task execution, it is reasonable to expect a reduction in the proportion of browsing and course enrollment relative to other tasks. The observed graph confirms this trend, indicating a higher silhouette value for this particular month as well. Notably, the category "creating a course" maintained a relatively consistent size when compared to the overall dataset, primarily because a significant number of courses were created during this period.

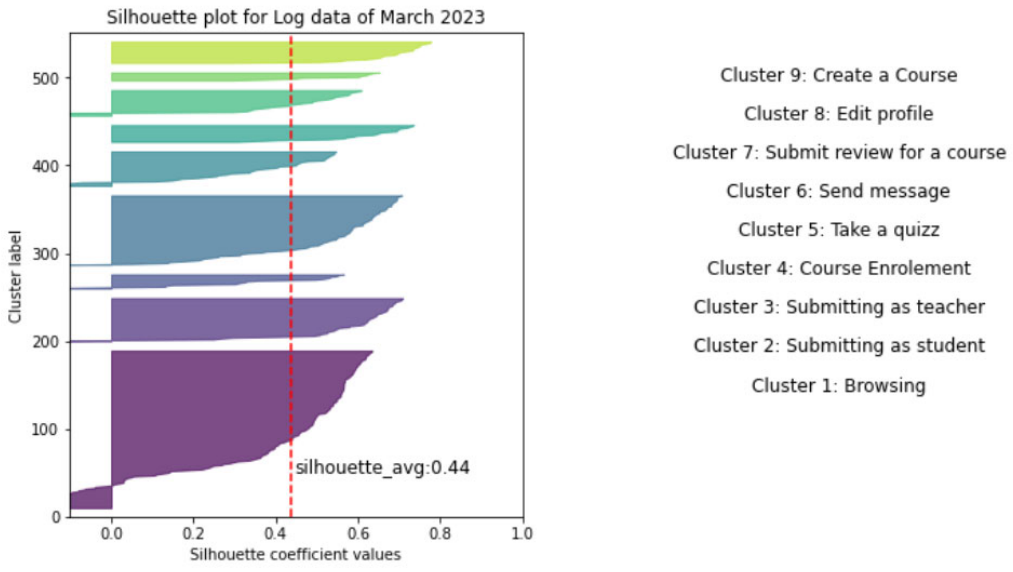


Figure 33: Silhouette plot for clustering March 2023 data records using the HMM model.

5.6.5 Examining Usability Metrics

The final set of tests aimed to evaluate the performance of usability metrics. In particular, the focus was on assessing the error rate, as this metric plays a fundamental role in computing various usability attributes. To conduct these tests, five synthetic datasets were generated. These datasets were kept consistent in terms of the number of sessions and the initial task, with the primary difference lying in the varying levels of information density they contained.

The concept of information density, as defined in Chapter IV, dictates that as information density increases, the likelihood of encountering errors increases. A parallel set of experiments is perpormed by creating an additional five datasets. In this case, the 'significance of the Code' variable was heightened in each dataset. The resulting graphs illustrate the shifting error rates in response to these modifications, consistently reflecting that as the data complexity and the number of errors increase, the model's error rate calculations also rise. (Figure 34, Figure 35)

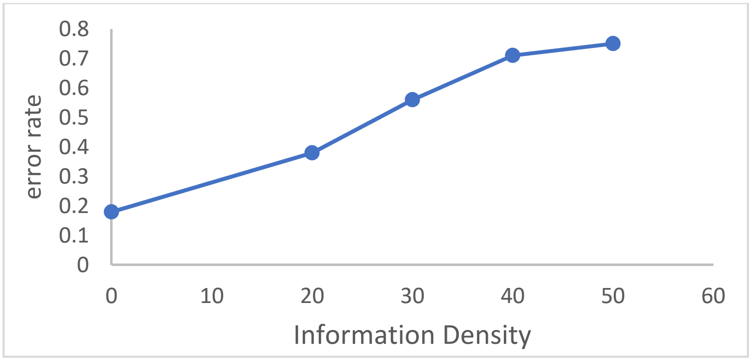


Figure 34:the effect of changing information density on error rate

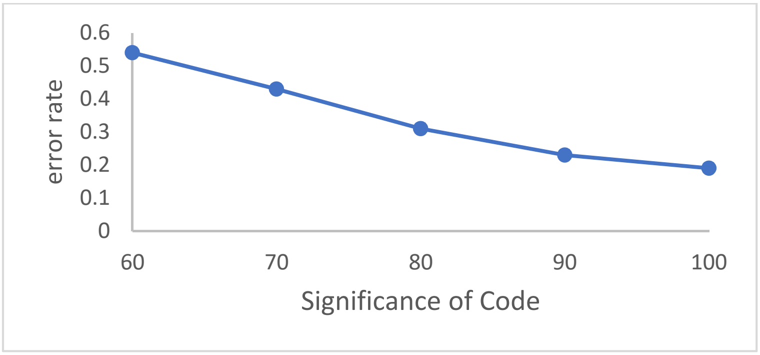


Figure 35: the effect of changing Significance of Code on error rate

Furthermore, in order to assess usability metrics with real data, a specific evaluation was conducted by calculating the error rates for two distinct tasks: course enrollment and student submission. The error rate for student submission was determined to be 0.08, while for course creation, it was found to be 0.21. Notably, this outcome aligns with expectations, as course creation typically involves more complexity, making the slightly higher error rate entirely consistent with reality.

5.7 Conclusion

This dissertation employs a combination of data mining and machine learning techniques to extract valuable insights from log files. Initially, it clusters visited page sequences into task models and subsequently quantifies usability metrics based on user performance. The evaluation of this model involves the use of both real data collected from an eLearning website and simulated data created using a Bayesian network model. It presents several significant contributions to the realm of automatic usability evaluation through log file analysis.

1. **Comprehensive Comparison Study and Categorization**: This study commences by conducting a thorough review of the existing literature in this field, offering a meticulous comparison and categorization based on diverse criteria, including the type of analysis they have applied, the range of calculable usability attributes, their requisite data and assumptions, and their scalability and reusability.
2. **Usability Metric and Attributes Extraction Framework**: A comprehensive framework is introduced that outlines all the necessary steps for extracting usability metrics and attributes from log file analysis. This framework not only serves as a roadmap for the majority of studies in the field but also provides a list of possible approaches at each juncture of the analysis process.
3. **Bayesian Model for Log Data Simulation**: To overcome the challenge of collecting log data for various scenarios, this dissertation introduces a Bayesian

model for simulating log data based on the usability features of web pages and website structure. The obtained simulated log file has a wide range of applications in various web usage mining studies. In the context of this research, it played a pivotal role in aiding with model selection, parameter estimation, and evaluating model performance.

4. **Task Identification**: This dissertation presents a novel two step clustering model to automatically identify users' intended tasks based on their sequence of page visits and website structure. The first step is based on similarity distance and the second step uses the information from the first step to create and learn an HMM model.
5. **Usability Metric Algorithm and Calculation Method**: Finally, this dissertation offers detailed algorithms and calculation methods for several examples of usability metrics derived from extracted knowledge from log file, enhancing the usability assessment.

Several potential research directions and extensions of this work include:

- **Modeling Usability Attributes**: While this dissertation focuses on estimating usability metrics, the next step involves using these metrics to numerically model usability attributes, providing a more comprehensive usability evaluation. To achive this goal, after obtaining numerical values for each metric, the significance and weight of each metric in the usability features should be determined.
- **Automatic Task Model Extraction**: While manual task model definition was assumed for simplicity, future work can explore techniques to

automatically extract task models from large log files, reducing the need for manual intervention.

- **Incorporating Domain Knowledge**: Incorporating domain-specific knowledge into the model can lead to more tailored and accurate usability evaluations, enhancing the applicability of the framework. Many prior studies in the realm of web usage mining have focused on integrating domain information or adapting evolving web usage data. (Nasraoui & Krishnapuram, 2002; Nasraoui, Rojas, & Cardona, 2006) The integration of these approaches with the method proposed in this research holds the potential to enhance overall accuracy significantly.

These proposed future works have the potential to advance the field of automatic usability evaluation and further improve the practicality and effectiveness of log file analysis in web usability assessment.

REFERENCES

Agrawal, R., Srikant, R., & others. (1994). Fast algorithms for mining association rules. *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, *1215*, 487–499.

Amazon CloudWatch. (2023). Amazon CloudWatch. Retrieved from https://aws.amazon.com/cloudwatch/

Anderka, M., Klerx, T., Priesterjahn, S., & Büning, H. K. (2014). Automatic ATM Fraud Detection as a Sequence-based Anomaly Detection Problem. *ICPRAM*, 759–764.

Atterer, R., Wnuk, M., & Schmidt, A. (2006). Knowing the user's every move: user activity tracking for website usability evaluation and implicit interaction. *Proceedings of the 15th International Conference on World Wide Web*, 203–212.

Babaian, T., Lucas, W. T., & Topi, H. (2007). A Data-Driven Design for Deriving Usability Metrics. *ICSOFT (ISDM/EHST/DC)*, 154–159.

Bader, D., & Pagano, D. (2013). Towards automated detection of mobile usability issues. *Software Engineering 2013-Workshopband*.

Baker, S., Au, F., Dobbie, G., & Warren, I. (2008). Automated usability testing using HUI analyzer. *19th Australian Conference on Software Engineering (Aswec 2008)*, 579–588.

Balbo, S., Goschnick, S., Tong, D., & Paris, C. (2005). Leading web usability evaluations to WAUTER. *Proceedings of the 11th AusWeb, Gold Coast, Australia*, 23.

Burton, M. C., & Walther, J. B. (2006). The Value of Web Log Data in Use-Based Design and Testing. *Journal of Computer-Mediated Communication*.

https://doi.org/10.1111/j.1083-6101.2001.tb00121.x

Cao, Q., Qiao, Y., & Lyu, Z. (2017). Machine learning to detect anomalies in web log analysis. *2017 3rd IEEE International Conference on Computer and Communications (ICCC)*, 519–523.

de Santana, V. F., & Baranauskas, M. C. C. (2015). WELFIT: A remote evaluation tool for identifying Web usage patterns through client-side logging. *International Journal of Human-Computer Studies*, *76*, 40–49.

de Santana, V. F. de F., & Baranauskas, M. C. C. (2008). A prospect of websites evaluation tools based on event logs. *IFIP Human-Computer Interaction Symposium*, 99–104.

Delias, P., Doumpos, M., Grigoroudis, E., Manolitzas, P., & Matsatsinis, N. (2015). Supporting healthcare management decisions via robust clustering of event logs. *Knowledge-Based Systems*, *84*, 203–213.

Eddy, S. R. (1996). Hidden markov models. *Current Opinion in Structural Biology*, *6*(3), 361–365.

ElasticSearch. (2023). ElasticSearch. Retrieved from https://www.elastic.co/

Facca, F. M., & Lanzi, P. L. (2003). Recent developments in web usage mining research. *Data Warehousing and Knowledge Discovery: 5th International Conference, DaWak 2003, Prague, Czech Republic, September 3-5, 2003. Proceedings 5*, 140–150.

Fournier-Viger, P., Gomariz, A., Campos, M., & Thomas, R. (2014). Fast vertical mining of sequential patterns using co-occurrence information. *Advances in Knowledge Discovery and Data Mining: 18th Pacific-Asia Conference, PAKDD 2014, Tainan, Taiwan, May 13-16, 2014. Proceedings, Part I 18*, 40–52.

Fournier-Viger, P., Gomariz, A., Gueniche, T., Soltani, A., Wu, C.-W., Tseng, V. S., &

others. (2014). Spmf: a java open-source pattern mining library. *J. Mach. Learn. Res.*, *15*(1), 3389–3393.

Galitz, W. O. (1994). *It's time to clean your windows: Designing GUIs that work*. John Wiley \& Sons, Inc.

Gomariz, A., Campos, M., Marin, R., & Goethals, B. (2013). Clasp: An efficient algorithm for mining frequent closed sequences. *Advances in Knowledge Discovery and Data Mining: 17th Pacific-Asia Conference, PAKDD 2013, Gold Coast, Australia, April 14-17, 2013, Proceedings, Part I 17*, 50–61.

GrayLog. (2023). GrayLog. Retrieved from https://www.graylog.org/

Guzdial, M. (1993). *Deriving software usage patterns from log files*.

Han, J., Pei, J., & Yin, Y. (2000a). Frequent pattern tree: design and construction. *Proc. Conf. on the Management of Data (SIGMOD'00, Dallas, TX)*, 1–12.

Han, J., Pei, J., & Yin, Y. (2000b). Mining frequent patterns without candidate generation. *ACM Sigmod Record*, *29*(2), 1–12.

Hartson, R., & Pyla, P. S. (2012). The UX Book: Process and Guidelines for Ensuring a Quality User Experience. In *The UX Book: Process and Guidelines for Ensuring a Quality User Experience*. https://doi.org/10.1016/C2010-0-66326-7

Hashim, N. L., & Isse, A. J. (2019). Usability evaluation metrics of tourism mobile applications. *Journal of Software Engineering and Applications*, *12*(7), 267–277.

Hong, J. I., Heer, J., Waterson, S., & Landay, J. A. (2001). WebQuilt: A proxy-based approach to remote web usability testing. *ACM Transactions on Information Systems*, *19*(3), 263–285.

Hong, J. I., & Landay, J. A. (2001). WebQuilt: A framework for capturing and visualizing the web experience. *Proceedings of the 10th International Conference on World Wide Web, WWW 2001*. https://doi.org/10.1145/371920.372188

Hussain, A., Mkpojiogu, E. O. C., & Jasin, N. M. (2017). Usability metrics and methods for public transportation applications: a systematic review. *Journal of Engineering Science and Technology (JESTEC)*, 98–105.

International Organization for Standardization. (1998). *ISO 9241-11:Ergonomic requirements for office work with visual display terminals (VDTs): Part 11: Guidance on usability*.

Inversini, A., Cantoni, L., & Bolchini, D. (2011). Connecting usages with usability analysis through the user experience risk assessment model: A case study in the tourism domain. *International Conference of Design, User Experience, and Usability*, 283–293.

Jansen, B. J. (2006). Search log analysis: What it is, what's been done, how to do it. *Library \& Information Science Research*, *28*(3), 407–432.

Jorritsma, W., Cnossen, F., Dierckx, R. A., Oudkerk, M., & van Ooijen, P. M. A. (2016). Pattern mining of user interaction logs for a post-deployment usability evaluation of a radiology PACS client. *International Journal of Medical Informatics*, *85*(1), 36–42.

Kandpal, N., Singh, H. P., & Shekhawat, M. S. (2019). Application of web usage mining for administration and improvement of online counseling website. *Int J Appl Eng Res*, *14*(7), 1431–1437.

Kandpal, N., Sinha, R. R., & Shekhawat, M. S. (2017). A survey on web usage mining: process, application and tools. *Suresh Gyan Vihar University Journal of Engineering \& Technology*, *3*(1), 19–25.

Kibana. (2023). Kibana. Retrieved from https://www.elastic.co/kibana/

Koller, D., & Friedman, N. (2009). *Probabilistic graphical models: principles and techniques*. MIT press.

Kopanitsa, G. D., Tsvetkova, Z., & Veseli, H. (2012). Analysis of metrics for the usability evaluation of EHR management systems. *MIE*, 358–362.

Kumar, B. T. H., Vibha, L., & Venugopal, K. R. (2016). Web page access prediction using hierarchical clustering based on modified Levenshtein distance and higher order Markov model. *2016 IEEE Region 10 Symposium (TENSYMP)*, 1–6.

Kumar, V., & Thakur, R. S. (2017). A brief investigation on web usage mining tools (WUM). *Saudi J. Eng. Technol*, *2*(1), 1–11.

Le, V.-H., & Zhang, H. (2022). Log-based anomaly detection with deep learning: How far are we? *Proceedings of the 44th International Conference on Software Engineering*, 1356–1367.

Levandowsky, M., & Winter, D. (1971). Distance between sets. *Nature*, *234*(5323), 34–35.

Lewis, J. R. (1994). Sample sizes for usability studies: Additional considerations. *Human Factors*, *36*(2), 368–378.

Lin, H. X., Choong, Y.-Y., & Salvendy, G. (1997). A proposed index of usability: a method for comparing the relative usability of different software systems. *Behaviour \& Information Technology*, *16*(4–5), 267–277.

Liu, X., Liu, W., Di, X., Li, J., Cai, B., Ren, W., & Yang, H. (2021). LogNADS: Network anomaly detection scheme based on log semantics representation. *Future Generation Computer Systems*, *124*, 390–405.

LogStash. (2023). LogStash. Retrieved from https://www.elastic.co/logstash/

López, J. M., Fajardo, I., & Abascal, J. (2007). Towards Remote Empirical Evaluation of Web Pages' Usability. *International Conference on Human-Computer Interaction*, 594–603.

Malik, S. K., & Rizvi, S. A. M. (2011). Information extraction using web usage mining, web scrapping and semantic annotation. *2011 International Conference on Computational Intelligence and Communication Networks*, 465–469.

McBride, S., Quimby, R., & Shih, B. (2005). *Understanding Web Usage Patterns Through Path Analysis*.

Menezes, C., & Nonnecke, B. (2014). UX-Log: understanding website usability through recreating users' experiences in logfiles. *Journal ISSN*, *2368*, 6103.

Menezes, T. C., & Nonnecke, B. (2014). UX-Log: Understanding Website Usability through Recreating Users' Experiences in Logfiles. *International Journal of Virtual Worlds and Human Computer Interaction*, (November). https://doi.org/10.11159/vwhci.2014.006

Mobasher, B. (2006). 12 web usage mining. *Encyclopedia of Data Warehousing and Data Mining Idea Group Publishing*, 449–483.

Moran, K. (2018). Quantitative User-Research Methodologies: An Overview. Retrieved from Nielsen Norman Group website: https://www.nngroup.com/articles/quantitative-user-research-methods/

Nasraoui, O., & Krishnapuram, R. (2002). A new evolutionary approach to web usage and context sensitive associations mining. *International Journal on Computational Intelligence and Applications-Special Issue on Internet Intelligent Systems*, *2*(3), 339–

348.

Nasraoui, O., Krishnapuram, R., Joshi, A., & Kamdar, T. (2002). Automatic web user profiling and personalization using robust fuzzy relational clustering. *E-Commerce and Intelligent Methods*, 233–261.

Nasraoui, O., Rojas, C., & Cardona, C. (2006). A framework for mining evolving trends in web data streams using dynamic learning and retrospective validation. *Computer Networks*, *50*(10), 1488–1512.

Nielsen, J. (1994a). Estimating the number of subjects needed for a thinking aloud test. *International Journal of Human-Computer Studies*, *41*(3), 385–397.

Nielsen, J. (1994b). *Usability engineering*. Morgan Kaufmann.

Nielsen, J., & Molich, R. (1990). Heuristic evaluation of user interfaces. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 249–256.

Nielsen Norman Group. (2023). When to Use Which UX Research Method. Retrieved from https://www.youtube.com/watch?v=OtUWbsvCujM

NXLog. (2023). NXLog. Retrieved from https://nxlog.co/

Okada, H., & Asahi, T. (1999). Guitester: A log-based usability testing tool for graphical user interfaces. *IEICE Transactions on Information and Systems*, *82*(6), 1030–1041.

Oppermann, R., & Reiterer, H. (1997). Software evaluation using the 9241 evaluator. *Behaviour \& Information Technology*, *16*(4–5), 232–245.

Organizacion Internacional de Normatizacion - ISO. (2018). ISO 9241-11:2018(en), Ergonomics of human-system interaction — Part 11: Usability: Definitions and concepts.

Oztekin, A., Kong, Z. J., & Uysal, O. (2010). UseLearn: A novel checklist and usability evaluation method for eLearning systems by criticality metric analysis. *International Journal of Industrial Ergonomics*, *40*(4), 455–469.

Paganelli, L., & Paterno, F. (2002). Intelligent analysis of user interactions with web applications. *Proceedings of the 7th International Conference on Intelligent User Interfaces*, 111–118.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, *77*(2), 257–286.

Rice, R. E., & Borgman, C. L. (1983). The use of computer-monitored data in information science and communication research. *Journal of the American Society for Information Science*, *34*(4), 247–256.

Rohrer, C. (2014). When to use which user-experience research methods. *Nielsen Norman Group*, *12*.

Rojas, C., & Nasraoui, O. (2007). Summarizing evolving data streams using dynamic prefix trees. *IEEE/WIC/ACM International Conference on Web Intelligence (WI'07)*, 221–227.

Rubin, J., & Chisnell, D. (2008). *Handbook of usability testing: how to plan, design and conduct effective tests*. John Wiley \& Sons.

Scapin, D. L., & Bastien, J. M. C. (1997). Ergonomic criteria for evaluating the ergonomic quality of interactive systems. *Behaviour \& Information Technology*, *16*(4–5), 220–231.

Scholtz, J. (2006). Beyond usability: Evaluation aspects of visual analytic environments. *2006 IEEE Symposium On Visual Analytics Science And Technology*, 145–150.

Scott, S. L., & Hann, I.-H. (2006). A nested hidden markov model for internet browsing behavior. *Marshall School of Business*, 1–26.

Shackel, B. (2009). Usability--Context, framework, definition, design and evaluation. *Interacting with Computers*, *21*(5–6), 339–346.

Shackel, B., & Richardson, S. J. (1991). *Human factors for informatics usability*. Cambridge university press.

Singh, B., & Singh, H. K. (2010). Web data mining research: a survey. *2010 IEEE International Conference on Computational Intelligence and Computing Research*, 1–10.

Siochi, A. C., & Hix, D. (1991). A study of computer-supported user interface evaluation using maximal repeating pattern analysis. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 301–305.

SplunkInc. (2005). Splunk. Retrieved from http://www.splunk.com

Srivastava, J., Cooley, R., Deshpande, M., & Tan, P.-N. (2000). Web usage mining: Discovery and applications of usage patterns from web data. *Acm Sigkdd Explorations Newsletter*, *1*(2), 12–23.

Suneetha, K. R., & Krishnamoorthi, R. (2009). Identifying user behavior by analyzing web server access log file. *IJCSNS International Journal of Computer Science and Network Security*, *9*(4), 327–332.

Tarby, J.-C., & Barthet, M.-F. (1996). The DIANE+ Method. *CADUI*, *96*, 95–119.

Tiedtke, T., Märtin, C., & Gerth, N. (2002). AWUSA--a tool for automated website usability analysis. *Proceedings of 9th International Workshop on Design, Specification and Verification DSV-IS*.

Van Dongen, B. F., de Medeiros, A. K. A., Verbeek, H. M. W., Weijters, A., & van Der

Aalst, W. M. P. (2005). The ProM framework: A new era in process mining tool support. *International Conference on Application and Theory of Petri Nets*, 444–454.

Vargas, A., Weffers, H., & Da Rocha, H. V. (2010). A method for remote and semi-automatic usability evaluation of web-based applications through users behavior analysis. *Proceedings of the 7th International Conference on Methods and Techniques in Behavioral Research*, 1–5.

Varnagar, C. R., Madhak, N. N., Kodinariya, T. M., & Rathod, J. N. (2013). Web usage mining: a review on process, methods and techniques. *2013 International Conference on Information Communication and Embedded Systems (ICICES)*, 40–46.

Wang, G., Konolige, T., Wilson, C., Wang, X., Zheng, H., & Zhao, B. Y. (2013). You are how you click: Clickstream analysis for sybil detection. *22nd USENIX Security Symposium (USENIX Security 13)*, 241–256.

Wronikowska, M. W., Malycha, J., Morgan, L. J., Westgate, V., Petrinic, T., Young, J. D., & Watkinson, P. J. (2021). Systematic review of applied usability metrics within usability evaluation methods for hospital electronic healthcare record systems: Metrics and Evaluation Methods for eHealth Systems. *Journal of Evaluation in Clinical Practice*, *27*(6), 1403–1416.

Xia, C., & Tingjie, L. (2009). Intelligent business prediction in context-awareness services based on hidden Markov model (HMM). *2009 Second Pacific-Asia Conference on Web Mining and Web-Based Application*, 116–119.

Yan, T. W., Jacobsen, M., Garcia-Molina, H., & Dayal, U. (1996). From user access patterns to dynamic hypertext linking. *Computer Networks and ISDN Systems*, *28*(7–11), 1007–1014.

Ypma, A., & Heskes, T. (2003). Automatic categorization of web pages and user clustering with mixtures of hidden markov models. *WEBKDD 2002-Mining Web Data for Discovering Usage Patterns and Profiles: 4th International Workshop, Edmonton, Canada, July 23, 2002. Revised Papers 4*, 35–49.

Zaki, M. J., Parthasarathy, S., Ogihara, M., Li, W., & others. (1997). New algorithms for fast discovery of association rules. *KDD*, *97*, 283–286.

APPENDIX

Appendix A: User Tasks for Read Data Collection

Task #	Task Detail
Task 1	• Go to the learninjawebsite.com • Login if you have an account or register as a student if you are new. • Set a photo for your profile • Search and save 3 courses about Python programming to your wish list. The wish list is represented by: 🔖 • Log out • Login with your username and password • Find your saved wish list and enroll in one of the courses that you added before.
Task 2	• Go to the learninjawebsite.com • Find all courses in the business category. • Find the 'Usability' and enroll in the course (log in or register on the website if it is required) • Finish the course and complete the quiz. • Write a comment on the first lesson. • Write a review for the course.
Task 3	• Go to the learninjawebsite.com • Find the 'Usability' course and enroll in it (log in or register on the website if it is required) • Finish the course and complete the quiz. (you don't need really to watch the whole video to complete it. You can fast-forward it manually) • Ask a question about the course in the course forum (any question is acceptable)
Task 4	• Go to the learninjawebsite.com • Login if you have an account or register as a student if you are new. • Find a course about Python programming and enroll in one of the courses • Finish the course (you don't need really to watch the course material to complete it.). • Write a review for the course

	• Ask a question about the course in the course forum (any question is acceptable).
Task 5	• Go to the learninjawebsite.com • Find their contact information and send a message to them and ask what the requirements are to be an instructor. • Register as an instructor • Search for Python courses. • Enroll in one of the courses in the search result.
Task 6	• Go to the learninjawebsite.com • Login if you have an account or register as a student if you are new. • Search for courses that are created for beginners. • From the search result, add 5 courses to your wish list. The wish list is represented by: 🔖 • Logout • Login again with your user-id and password • Find your saved wish list on the website • Enroll in one of the courses • Complete the course and write a review on it
Task 7	• Go to the learninjawebsite.com • Signup as a student • After signing up as a student, apply to Become an instructor.
Task 8	• Go to the learninjawebsite.com • login with the following username and password to the website: username: instructor1 password: instructor1620 • create a new course with any subject you want (you can use youtube videos or even define an empty course) • add a category for your course • If you had any questions send a message through the website and ask your questions
Task 9	• Go to the learninjawebsite.com • Find which universities and companies are the partners of learninjawebsite.com • Ask them "How UofL can be a partner of them?"
Task 10	• Go to the learninjawebsite.com • Find the profile and all courses created by 'Sima'

	• Enroll in one of the courses in the 'Personal Development' category (log in if you already have an account or register as a student if you are new on the website) • Complete the course. (you don't need really to watch the course material to complete it) • Write a review for the course.

Appendix B: The Approved IRB Consent Document

UofL Institutional Review Boards
IRB NUMBER: 23.0339
IRB APPROVAL DATE: 06/16/2023

TITLE OF RESEARCH STUDY: Automated Usability Evaluation Utilizing Log Files and Data Mining Techniques

Introduction and Background Information
You are invited to take part in a research study about how to automate the whole process of usability evaluation by utilizing data mining and machine learning techniques on data recorded in log files. The study is being conducted under the direction of Dr. Jason Saleem at the University of Louisville.

Why is this study being done?
The purpose of this study is to create a model that can be used for automated usability evaluation of log files. We wish to develop a general comprehensive framework that covers all the usability evaluation steps through the analysis of log files. By automating usability evaluation, knowledge from this research may save usability teams substantial resources since "traditional" usability evaluation is quite labor-intensive (e.g., recruiting participants and hours spent conducting live user sessions).

What will happen if I take part in the study?
Your participation in the study will involve completing a couple of tasks, which we will assign to you, on a fictitious e-learning Web site that we created. Your participation in this study will last for up to 20 minutes.

What are the possible risks or discomforts from being in this research study?
There are no known risks for participation in this research study.

What are the benefits of taking part in the study?
You may or may not benefit personally by participating in this study. The information collected may not benefit you directly; however, the information may be helpful to others.

Will I be paid?
You will not be paid for your time, inconvenience, or expenses while you are in this study.

How will my information be protected?
The data collected about you will be kept private and secure by a password-protected computer in the Industrial Engineering (IE) graduate student office (JBS 302). The IE graduate student office is a secure space with a keypad locked door, which gives an additional level of protection to the password-protected computer. We will follow the University of Louisville's information security policies and standards for safeguarding your information.

Individuals from the Department of Industrial Engineering, the Institutional Review Board (IRB), the Human Subjects Protection Program Office (HSPPO), the University of Louisville, and other regulatory agencies may inspect these study records. In all other respects, however, the data will be held in confidence to the extent permitted by law. Should the data be published, your identity will not be disclosed.

Will my information be used for future research?
Your data will be stored and shared for future research without additional informed consent if identifiable private information, such as your name, are removed. If identifying information is removed from your data,

Version Date: 06/12/23
Version Number: 1.1
Page **1** of **2**

the data may be used for future research studies or given to another investigator for future research studies without additional consent from you.

Can I stop participating in the study at any time?
Taking part in this study is completely voluntary. You may choose not to take part at all. If you decide to be in this study, you may change your mind and stop taking part at any time. You will not be penalized or lose any benefits for which you qualify.

Who can I contact for questions, concerns and complaints?
If you have any questions about the research study, please contact Dr. Jason Saleem at Jason.saleem@louisville.edu or (502) 852-2274.

If you have concerns or complaints about the research or research staff and you do not wish to give your name, you may call this toll free number: 1-877-852-1167. This is a 24 hour hot line answered by people who do not work at the University of Louisville.

If you have any questions about your rights as a research participant, you may call the Human Subjects Protection Program Office at (502) 852-5188. You may discuss any questions about your rights as a research participant, in private, with a member of the Institutional Review Board (IRB).

Acknowledgment
This document tells you what will happen during the study if you choose to take part. By starting the first assigned task with the Web site, you agree to take part in this study.

You are not giving up any legal rights to which you are entitled by consenting to this study. You can save this consent form for your records.

[illegible]

Dr. Jason J. Saleem

CURRICULUM VITA

NAME:	Sima Shafaei
ADDRESS:	Department of Industrial Engineering 220 Eastern Pkwy, Louisville, KY 40292
DOB:	Hamedan, Iran - April 7, 1981
EDUCATION & TRAINING:	B.S., Software Engineering Isfahan University of Technology 2001-2005
	M.Sc. Artificial Intelligence University of Isfahan 2006-2008
	Ph.D. Industrial Engineering University of Louisville 2019-2023
AWARDS:	Recipient IE Research Competition and Innovation Award, University of Louisville, 2023
	Recipient IE Outstanding GTA Award, University of Louisville, 2023
	Recipient Doctoral Dissertation Completion Award, University of Louisville, 2023

Annual Award for INFORMS Student Chapter as the Web Master,
INFORMS Institution,
2021

Outstanding Student Award,
University of Isfahan,
2008

Outstanding Student Award,
Isfahan University of Technology,
2008

PUBLICATIONS: Shafaei, Sima, and Nasser Ghasem Aghaee. "Biological network simulation using holonic multiagent systems." Tenth International Conference on Computer Modeling and Simulation (uksim 2008). IEEE, 2008.