User Behavioral Modeling for Web-based Systems

by

Saeedeh Sadat Sajjadi Ghaemmaghami

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Software Engineering & Intelligent Systems

Department of Electrical and Computer Engineering
University of Alberta

Abstract

Today's users are spending more time on web applications. Many users browse web applications and navigate through different web pages. They may have different interests, especially when it comes to large-scale applications. The more the developers of the applications know about their users' needs and interests, the smarter choices they will make for their application's development. Inferring a behavioral model from users' navigation patterns in a web application helps application providers to understand their users' interests. A navigational pattern is a record of where a user visits; the pattern is extracted from the start to the end of a user session. User navigation information is obtained by collecting the data in a log file.

Some studies instrumented the application's web pages to collect data and then model user navigational behaviors. To instrument a web page, the source code of the program is modified with additional commands. However, this can be difficult when the source code is inaccessible. Ideally, a user behavioral inference process should not be required to instrument the application's web pages to generate a user behavioral model.

Also, a model generation approach needs to support the evolution of web applications. A behavioral model should be generated incrementally during its evolution and should play a role in the application's evolution (upgrading) procedure. This can help sustain web applications.

Inferring a model by predicting and analyzing users' navigational behaviors is necessary to understand users' interests. Developers can identify interesting (from users' perspective) or problematic pages of applications and therefore improve the application design. Analyzing the behavioral model helps to detect design anomalies such as dead-ends; pages in which users are being prevented from leaving the page without closing it. Detecting dead-ends can significantly help in addressing design anomalies and providing solutions to retain users. Satisfied customers are more likely to stay with the company and contribute to its success.

It is ideal to analyze web pages to model user behaviors. Web page analysis methods utilize web page segmentation which is the process of segmenting a web page into different blocks, where each block contains similar components in terms of structural, visual, or contextual similarity. Current segmentation methods use the Document Object Model (DOM) structure of a web page and vision-based techniques to segment a web page. However, current methods do not consider semantic analysis to categorize pages. Semantic analysis includes extracting text from segmented blocks, computing textual similarity, and regrouping blocks.

In this research, we attempt to bridge several gaps in all the above-mentioned areas. Firstly, we provide an automated approach, with no instrumentation, to generate user behavioral models. We evaluate the utility of our approach by using it on a large-scale mobile and desktop application. Also, we evaluate the evolving properties of interaction patterns against the inferred behavioral models using an analysis engine.

Next, we present a new combination model of web page segmentation, namely

Fusion-Block, by dividing the content of a web page into blocks by initially considering human perception (inspired by Gestalt laws of grouping) and subsequentially re-segmenting initial similar blocks using semantic text similarity.

Hence, in the next part of our research, we improve the segmentation model, namely Integrated-Block, by merging the DOM structure, vision-based, and text-based similarity metrics of web pages. Finally, to verify the effectiveness of our approach, we applied it to the public datasets and compared it with the five existing state-of-the-art algorithms. We demonstrate the value and novelty of the presented solutions using extensive evaluations throughout the thesis.

Preface

This thesis is an original work by Saeedeh Sadat Sajjadi Ghaemmaghami submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Software Engineering & Intelligent Systems. The thesis is written in paper-based format with the details as follows:

Chapter 3 of this thesis contains the article: Saeedeh Sadat Sajjadi Ghaemmaghami and James Miller, "Automatically Inferring User Behavior Models in Large-Scale Web Applications", In Information and Software Technology, Elsevier, Volume 141, January 2022, Article 106704.

Chapter 4 of this thesis contains the following two articles:

Saeedeh Sadat Sajjadi Ghaemmaghami and James Miller, "A New Semantic Approach to Improve Web page Segmentation", In Journal of Web Engineering, Volume 20_4, June 2021, Pages: 963–992, DOI: 10.13052/jwe1540-9589.2042.

Saeedeh Sadat Sajjadi Ghaemmaghami and James Miller, "Integrated-Block: A New Combination Model to Improve Web Page Segmentation", Under the second round of revision at the Journal of Web Engineering.

Acknowledgments

First and foremost, I wish to express my deepest gratitude to my supervisor Prof. James Miller for his guidance, motivation, and support. His constant encouragement and immense knowledge helped me throughout the years of my Ph.D. research.

Secondly, I would like to thank the members of my supervisory committee: Dr. Marek Reformat and Dr. Cor-Paul Bezemer for their thoughtful comments and constructive feedbacks during my doctoral internal exam.

Finally, I would like to thank my husband Mohammadreza, my parents Akram and Mohammadbagher, my brother Salman, and my sister Sareh for their constant love, encouragement, and support.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

As a detailed list of articles is provided in the preface, in this section we elaborate on the motivation and a brief overview of our research. Moreover, this section also summarizes the primary contributions of our overall thesis followed by a discussion on the organization of the thesis documentation.

1.1 Motivation

Nowadays users are spending more time on web applications. They browse web pages and navigate through different pages. They may have different and evolving interests and preferences, especially when it comes to large-scale applications. Knowing and predicting the different user behaviors are crucial factors that may directly affect the success of the application. Underestimating the importance of these factors may lead to software limitations which can easily lead to lost audiences; dead-ends and navigational anomalies are some examples of the limitations.

However, the presence of a huge number of users with different behaviors makes it almost impossible to accurately predict and model all of them, and to design applications that can answer all possible needs. Moreover, the population of users is seldom homogenous and, typically, several classes of users with distinct user behaviors coexist at the same time. Also, no matter how well they are initially captured, user behaviors change over time. This leads to the need for learning and refining our understanding of how users interact with the system.

The mainstream approach towards capturing the user behaviors consists of monitoring the usage of the system and subsequently mining possible interaction patterns [1]. Some existing solutions instrument web pages to track users' navigation actions, for example, Google Analytics [2], while others analyze log files such as [3]. However, current solutions suffer from several limitations. Some approaches lack general applicability, for example, they need to infer users' navigational behaviors to support specific tasks [4, 5]. On the other hand, the general frameworks simply return a set of statistics or patterns that are useful to understand the preferences of the system's users but cannot be directly used to evaluate and analyze the web page design.

Inferring a model by predicting and analyzing users' navigational behaviors (trajectories) is necessary to understand users' interests. Developers can identify popular or problematic pages of applications and therefore improve the application design and retain users. In addition, a model generation approach needs to support the evolution of web applications. A behavioral model should be generated incrementally during its evolution procedure. This can help sustain web applications.

The mainstream approach to model user behaviors is to analyze web pages of applications. Web page analysis methods utilize web page segmentation that is the process of segmenting a web page into different blocks, where each block contains similar components in terms of structural, visual, or textual similarity [6]. Current segmentation methods use the Document Object Model (DOM) structure of a web page and/or vision-based techniques to segment a web page [7-11]. However, these methods do not consider semantic analysis to categorize pages. Semantic analysis includes extracting text from segmented blocks, computing textual similarity, and regrouping blocks.

1.2 Research Overview

In this research, we attempt to bridge several gaps in all the above-mentioned areas. At first, we study different behavioral model generation techniques. Some research utilizes user navigational behavior in mobile applications. For example, [12] and [13] exploit application usage records to characterize patterns and discover user behavior. Some other research models mobile and desktop application usage patterns such as [14, 15]. The rest of the research uses web usage mining on desktop applications [16, 17].

Next, we generate user behavior models using Markov models and Reinforcement Learning to automatically and incrementally, learn users' interests. We evaluate the utility of our approach by using it on a large-scale mobile and desktop application. Then, we analyze the models using a model checker. By analyzing the inferred model, the application limitations are found. Next, we perform a survey of different academic behavioral model generation's algorithms and choose the 19 most popular algorithms to compare our proposed approach in terms of 9 factors such as whether an approach supports real-life application or not, or whether or not a method uses an instrumenting approach. A detailed list of factors is proposed in Section 3. It is ideal to analyze web pages of an application to model user behaviors. Web page analysis methods utilize web page segmentation to categorize the structure and content of a web page.

In the next phase, we shift our focus towards the challenges associated with web page segmentation. There are two major factors in segmenting a web page into different blocks, (1) how the content of a web page is extracted and (2) how the extracted content is processed to retrieve distinct information [7-10]. In this research first, we study different web page segmentation algorithms. There has been much research that focuses on segmenting a page based on the DOM structure of an HTML page [8, 18]. Some researchers prefer to segment web pages using visual information in a web page. This vision-based segmentation method focuses on the analysis of visual features of the document content as they are perceived by a human reader. It exploits visual clues such as font size, font color, background color, spaces between paragraphs, etc. [19]. Some segmentation methods have been carried out using Natural Language Processing (NLP) techniques [10, 20]. These methods consider text density metrics such as text formats and words' frequency of a document but do not consider the semantic analysis to categorize pages. Semantic analysis includes extracting text from segmented blocks, computing textual similarity, and regrouping blocks.

Next, we propose a new method of web page segmentation (Fusion-Block) by combining the DOM structure, visual features, and text similarity metrics to improve the segmentation performance. Our approach generates a segmentation model by utilizing human perception and semantic analysis of a web page. To achieve this, our model merges the content of a web page into basic-blocks and identifies similar blocks using text similarity, and regroups these similar blocks as fusion blocks. Thus, a fusion block is composed of related blocks in terms of similar text contents using NLP techniques. To evaluate the accuracy of our approach, we apply the approach on three datasets and compare the approach with 4 state-of-the-art methods.

In the next phase of this research, we focus on improving the web page segmentation algorithm. Towards achieving this goal, we provide a new semantic method of web page segmentation named Integrated-Block by merging the DOM structure, vision-based, and text-based similarity metrics of web pages. To improve the segmentation accuracy, we demonstrate the utility of transformer technology as a vehicle for the text-based process [21].

Finally, we evaluate our approach on three datasets and compare the approach with 5 state-of-the-art methods. The results represent that our proposed approach outperforms 5 other existing web page segmentation methods, in terms of higher accuracy.

1.3 Summary of Contributions

The primary contributions of this research are as follows:

- This research represents the first to our knowledge, 'state-of-the-art' automatic, non-instrumented behavioral models that have been applied to a mission-critical, real-world large-scale application (cf. Section 3.4).

- This research compares our model with the 19 popular behavioral model inference algorithms in terms of 9 factors (cf. Section 3.2).
- This research presents Fusion-Block, a novel algorithm that can provide a new method of web page segmentation by combining the DOM structure, visual features, and text similarity metrics (cf. Section 4).
- To improve the segmentation accuracy, this research presents Integrated-Block that provides a new semantic method of web page segmentation by merging the DOM structure, vision-based, and semantic similarity metrics of web pages. Also, this study demonstrates the utility of transformer technology as a vehicle for the semantic similarity process (cf. Section 4).
- Finally, this research evaluates our approaches on three public datasets and compares the approach with state-of-the-art methods. The results represent that our proposed approach (Integrated-Block) outperforms other existing web page segmentation methods, in terms of higher accuracy (cf. Section 4.4).

1.4 Thesis Organization

This thesis has been prepared in a paper-based format and is organized as follows:

Chapter 2 of this thesis provides background information about the Gestalt laws of grouping that are used in Chapter 3 and Chapter 4.

Chapter 3 of this thesis presents a new method to infer probabilistic user behavioral models using Markov models and Reinforcement Learning (RL). A probabilistic model checker is used to analyze the inferred models. The proposed method is applied to and evaluated on a mobile and desktop application.

Chapter 4 presents two new methods of web page segmentation named Fusion-Block and Integrated-Block, by combining the DOM structure, visual features, and text similarity metrics. These methods are applied to three datasets and evaluated by comparing the approach with 4 state-of-the-art methods.

Finally, Chapter 5 concludes the thesis and presents a set of directions for future work.

Chapter 2

Background Information

In this section, we explain the Gestalt laws of grouping that have been used in web page analysis in Chapter 3 and Chapter 4. In this research, the Gestalt laws of grouping are interpreted as the rules for block detection in web page segmentation. During interpretation, the normalized Hausdorff distance, the CIE-Lab color difference, the normalized compression distance, and the series of visual information are used the same as [22] to operationalize these Gestalt laws.

People can recognize related web page content fast and correctly even before reading it, regardless of the complexity of the web pages. According to Gestalt psychology, this is because that humans group objects based on a series of laws – the Gestalt laws of grouping [23-25].

The original Gestalt laws of grouping include eight items (each item represents a single law), i.e., the Gestalt laws of (a) simplicity, (b) closure, (c) proximity, (d) similarity, (e) continuity, (f) common fate, (g) symmetry, and (h) past experience [25, 26]. In the context of web page similarity, the Gestalt laws of symmetry and the Gestalt laws of past experience are not employed. This is because the former considers symmetric elements that are in widely scattered locations (which are very rare in web pages), and the latter refers to a higher level of human perceptions (i.e., it requires knowledge that is beyond the scope of web page analysis). Hence, we focus here on the remaining six laws the same as [22].

Xu and Miller [22] propose the Gestalt layer merging (GLM) model for merging web page content into semantic blocks based on human perception. Three components are included in this model, namely, the layer tree constructor, the Gestalt laws translator, and the web page block identifier.

According to the first component of this model, a layer tree is constructed to remove hierarchical inconsistencies between visual layout and DOM tree of web pages. The DOM tree is a fast and precise representation of a web page; however, it cannot be directly used as the input in this model [22]. People read only visible content from the web pages, so the invisible DOM elements are useless, i.e., they are noise to this model. Meanwhile, the visual hierarchy of a web page sometimes differs from the corresponding DOM hierarchy, causing perception errors to this model. Such noise and errors must be eliminated before analyzing. Thus, the layer tree constructor takes the DOM tree of a web page as a prototype to build up its layer tree. The construction includes removing the invisible DOM elements and fixing the hierarchy.

Definition: Given a web page WP, the layer tree LT of WP is a finite set where each element n (that is, layer tree node) of LT is a layer representing a visible element e from DOM tree DT of WP ($LT = \{n \mid n \leftarrow e, e \in DT\}$) and all elements follow the visual hierarchy of WP [22].

The second component of the GLM model is the Gestalt laws translator. In this step, the Gestalt laws of grouping are translated to computer-compatible rules that can train a classifier to combine the laws to a unified rule to detect semantic blocks (the last component of the GLM model). Thus, these laws need to be converted into machine-compatible rules using a Gestalt laws translator.

2.1 Gestalt Laws Translator

The Gestalt laws translator interprets the Gestalt laws of grouping into machine compatible rules within the domain of web pages. The Gestalt laws explain the mechanisms of how humans perceive and understand things. The Gestalt laws of grouping contain 6 laws described in the following paragraphs [22].

2.1.1 Gestalt Law of Simplicity

This law states that people tend to break down content into the simplest units when reading a web page. Although a web page can be split as small as a single pixel, we will not follow this method. This is because when we read the page, we focus on useful information such as a single image or a piece of text instead of pixels. The useful information corresponds to the DOM elements of a web page. Figure 2.1 shows the logo of the University of Alberta, "https://www.ualberta.ca". The logo contains multiple parts: the figure, the phrase "UNIVERSITY OF", and the big bold "ALBERTA". However, these elements have different types and styles, they are considered as a single group according to the Gestalt law of simplicity. This law helps to make the process of reading and understanding a page more straightforward.

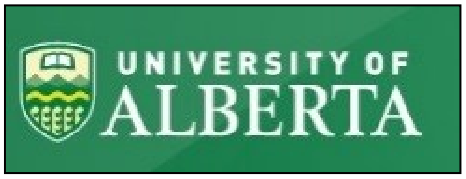
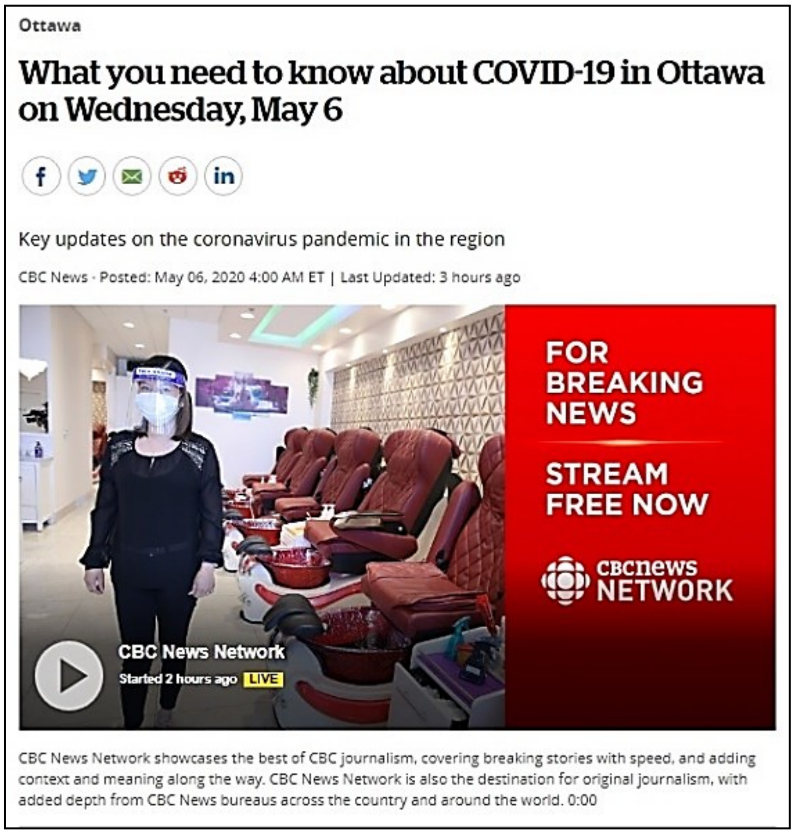


Figure 2.1: Gestalt Law of Simplicity ("www.ualberta.ca")

As another example, Figure 2.2 shows a part of the page of "cbc.ca/news/". In this figure, the middle image between the texts contains multiple elements (i.e., the text "FOR BREAKING NEWS", an image, and video). However, they have various styles, they are grouped as a single image.



Ottawa

What you need to know about COVID-19 in Ottawa on Wednesday, May 6

Key updates on the coronavirus pandemic in the region

CBC News · Posted: May 06, 2020 4:00 AM ET | Last Updated: 3 hours ago

CBC News Network showcases the best of CBC journalism, covering breaking stories with speed, and adding context and meaning along the way. CBC News Network is also the destination for original journalism, with added depth from CBC News bureaus across the country and around the world. 0:00

Figure 2.2: Gestalt Law of Simplicity ("www.cbc.ca/news/")

2.1.2 Gestalt Law of Closure

This law states that humans tend to perceive incomplete shapes as complete ones. As an example, Figure 2.3 represents the homepage of FedEx, "https://www.fedex.com/". However, the middle part of the background image is covered by a search and three other boxes, it is believed that the background image is complete [22].

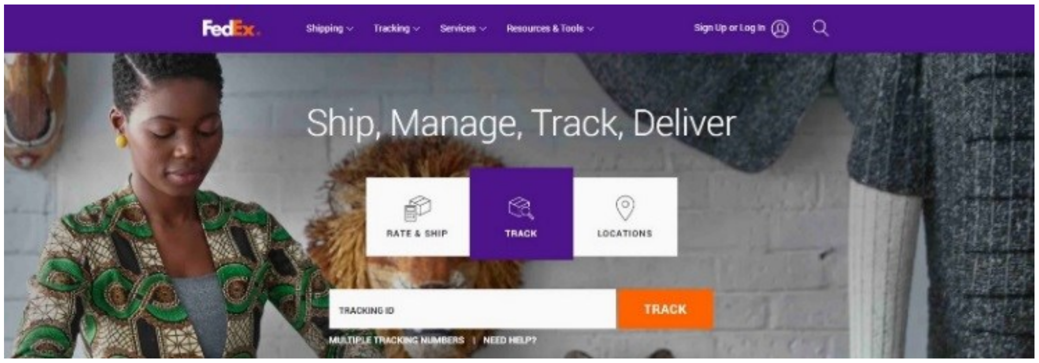


Figure 2.3: Gestalt Law of Closure (“www.fedex.com/”)

As another example, in Figure 2.4 (“ualberta.ca/admissions-programs”), the middle part of the background image is covered by a search box but according to this Gestalt law, the image is considered as a complete one.

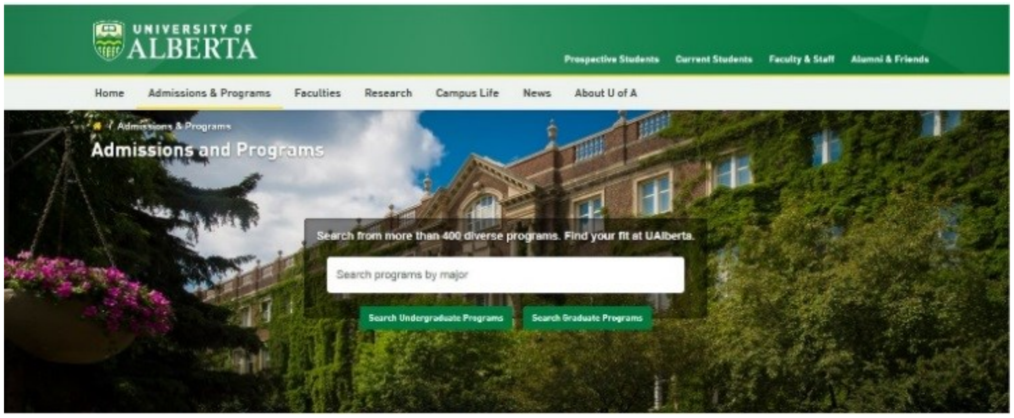


Figure 2.4: Gestalt Law of Closure (“www.ualberta.ca/admissions-programs/”)

2.1.3 Gestalt Law of Proximity

According to this law, humans tend to group close objects. This law groups elements based on their distances. To determine proximity, the distance in the GLM model is defined as the Normalized Hausdorff Distance (NHD) between layers, which provides the best performance as a proximity estimation [27]. NHD aims to group elements if their distances with adjacent elements are similar. Further details can be found in [27].

$$NHD(L_1, L_2) = \max\left\{(\frac{hd_{1,2}}{Re_{L_1}}, \frac{hd_{2,1}}{Re_{L_2}})\right\} \quad (2.1)$$

Re_{L_1} and Re_{L_2} are the relevant lengths of layers L_1 and L_2, and $hd_{1,2}$ and $hd_{2,1}$ are the Hausdorff distance from L_1 to L_2 and L_2 to L_1, respectively. Using the sign-in page of LinkedIn (https://www.linkedin.com/) as an example shown in Figure 2.5, the two boxes regarding sign-in ("Email or phone number" and "Password") are regarded as a group.

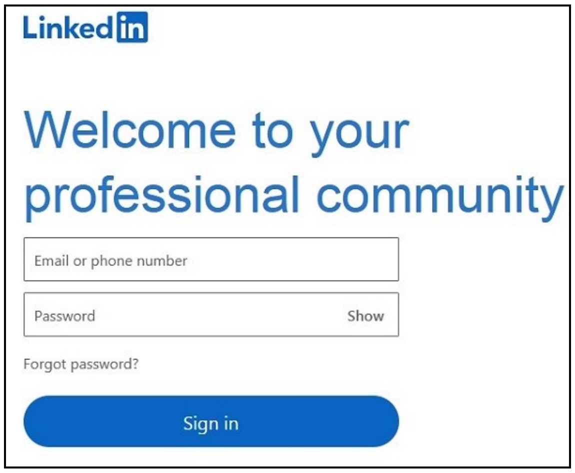


Figure 2.5: Gestalt Law of Proximity ("www.linkedin.com/")

As another example, considering the sign-up page of Instagram (https://www.instagram.com /accounts/emailsignup/) shown in Figure 2.6, the four boxes regarding sign up ("Mobile Number or Email", "Full Name", "Username", and "Password") are related and regarded as a group.

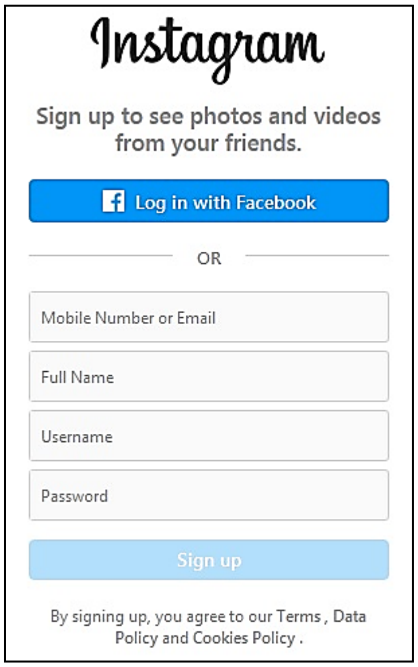

Instagram
Sign up to see photos and videos from your friends.
Log in with Facebook
OR
Mobile Number or Email
Full Name
Username
Password
Sign up
By signing up, you agree to our Terms , Data Policy and Cookies Policy .

Figure 2.6: Gestalt Law of Proximity ("www.instagram.com /accounts/emailsignup/")

2.1.4 Gestalt Law of Similarity

The Gestalt law of similarity indicates that humans perceive similar elements as a single group. To compare elements, this law considers their visual features such as background similarity, foreground similarity, and size similarity. Background similarity includes both the color and the image; foreground similarity compares textual and paragraph styles; and size similarity checks if the two blocks share the same width or height. A more precise set of definitions can be found in [28].

In this research, we use CIE-Lab color space in the same manner as [27] to simulate human vision. We select ΔE_{00}^{12} as the color difference metric same as [27], calculated by Equation (2.2). The parameter list can be found in [27], and it is omitted in this work for brevity.

$$\Delta E_{00}^{12} = \sqrt{\left(\frac{\Delta L'}{k_L s_L}\right)^2 + \left(\frac{\Delta C'}{k_C s_C}\right)^2 + \left(\frac{\Delta H'}{k_H s_H}\right)^2 + R_T\left(\frac{\Delta C'}{k_C s_C}\right)\left(\frac{\Delta H'}{k_H s_H}\right)} \qquad (2.2)$$

Structural Similarity Index (SSIM) [25] is employed to calculate digital image comparison to imitate human understandings. SSIM is capable of distinguishing between similar pages and dissimilar pages [27]. This research uses SSIM in the same manner as [27]. Figure 2.7 shows six objects grouped into three groups in terms of styles. The top two objects are in one group, the next two objects are included in a second group, and the bottom two objects belong to a third group.

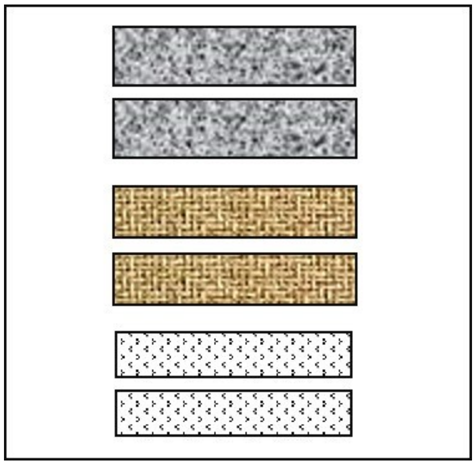
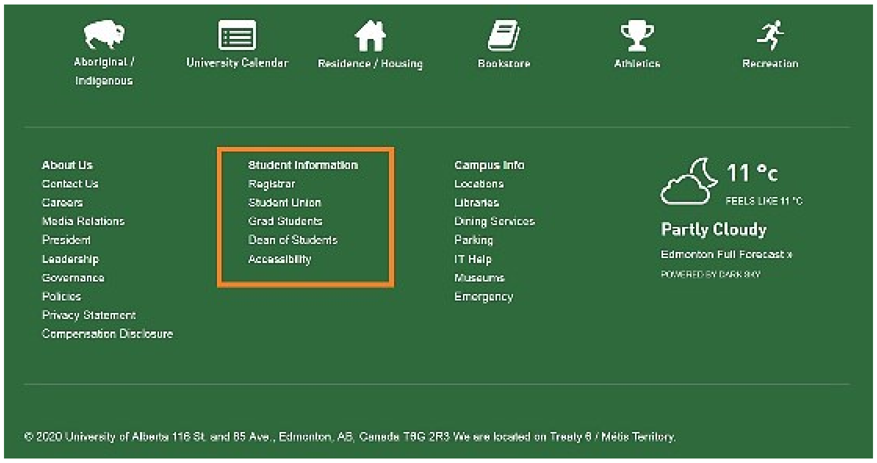


Figure 2.7: Gestalt Law of Similarity

2.1.5 Gestalt Law of Continuity

This law expresses that humans tend to judge the elements on a web page as related in a situation where they are aligned, and as dissimilar when they are not aligned. Using a part of the homepage of the University of Alberta (https://www.ualberta.ca/) as an example shown in Figure 2.8, the paragraphs in the orange rectangle ("Student Information", "Register", "Student Union", etc.) are left-aligned and categorized as a single group, indicating they are related content. To evaluate continuity, we compare the left, top, right, and bottom coordinates of the two elements. If any of the four coordinates of two elements are the same, we conclude that they are related and that they are dissimilar, otherwise [27].

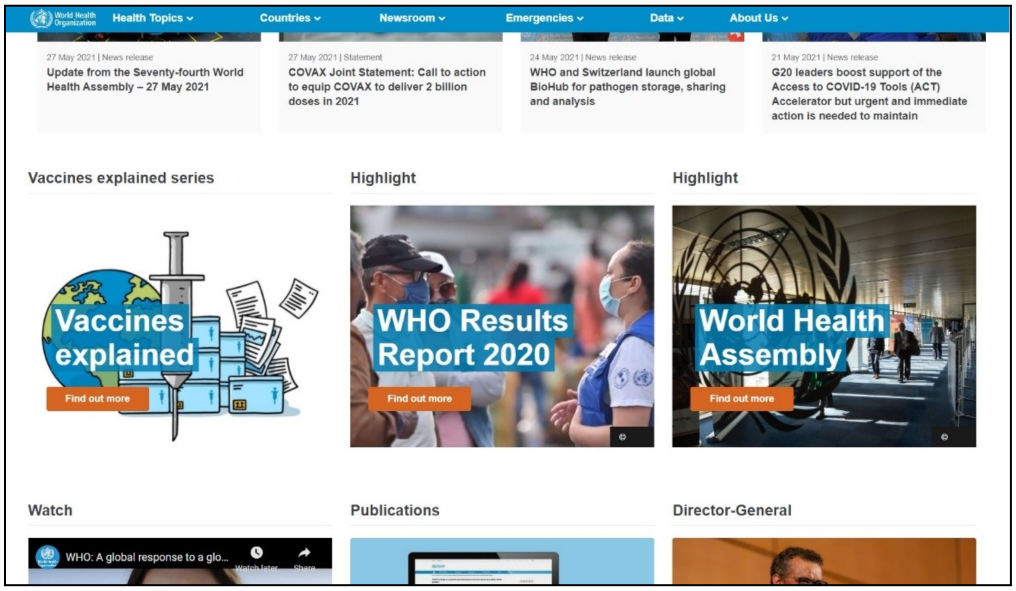


© 2020 University of Alberta 116 St. and 85 Ave., Edmonton, AB, Canada T6G 2R3 We are located on Treaty 6 / Métis Territory.

Figure 2.8: Gestalt Law of Continuity ("www.ualberta.ca")

2.1.6 Gestalt Law of Common Fate

This law describes that people tend to regard elements with the same motion trend as related. For example, the upper ribbon with the blue background color in Figure 2.9 (the homepage of World Health Organization, "www.who.int") hangs at the top and does not move with scrolling the page, but other content moves accordingly.



27 May 2021 | News release
Update from the Seventy-fourth World Health Assembly – 27 May 2021

27 May 2021 | Statement
COVAX Joint Statement: Call to action to equip COVAX to deliver 2 billion doses in 2021

24 May 2021 | News release
WHO and Switzerland launch global BioHub for pathogen storage, sharing and analysis

21 May 2021 | News release
G20 leaders boost support of the Access to COVID-19 Tools (ACT) Accelerator but urgent and immediate action is needed to maintain

Figure 2.9: Gestalt Law of Common Fate (Homepage of "www.who.int")

According to these six laws, a model can allow elements to be categorized whether in a group or not. This group of similar elements includes the results of six laws merged. Such merged groups represent the blocks. In this research, we use these laws in Chapter 3 and Chapter 4 as part of the web page analysis.

Chapter 3

Automatically Inferring User Behavior Models in Large-Scale Web Applications

Nowadays users are spending more time on web applications [29, 30]. Many users browse these applications and navigate through different web pages. They may have different interests, especially when it comes to large-scale applications. Software limitations can easily lead to lost audiences; dead-ends and navigational anomalies are some examples of the limitations. However, it is almost impossible to accurately predict and address all of the users' interaction expectations. Inferring a (behavior) model by predicting and analyzing users' navigational behaviors (trajectories) is necessary to understand users' interests. By analyzing the inferred behavior models, developers can identify interesting or problematic pages from applications; and therefore, improve the application's design. For example, consider an application that has an unidentified dead-end page. By inferring and analyzing behavior models of this application, developers can identify and remove the dead-end page and hence improve the application design. User navigation information is obtained by collecting the data in a log file.

Google Analytics[1] is a state-of-the-art approach to infer user behavior models and track users' actions. It infers user navigation behavior by instrumenting web pages. Google Analytics uses a page tagging approach to gather website traffic data. In this case, a snippet of JavaScript code needs to be manually added to every page of the application. It identifies popular (most visited) pages by instrumenting web pages. However, Google Analytics cannot automatically analyze the user navigational behavior of an application.

[1] https://www.google.ca/analytics/

Aside from Google Analytics, several studies have also proposed approaches to model user navigational behaviors. Applying data mining techniques on the data collected from the user side, or proxy servers, to extract usage patterns is one of the proposed approaches [1]. Additionally, client-side data can be collected using JavaScript or by modifying the source code of browsers; for example, [4, 5, 31-34] generate probabilistic user behavioral models from log files.

However, addressing many questions can remain problematic despite having an accurate user navigation behavioral model. Augmenting this model with appropriate metrics (reward values) is required to address those questions. For example, (1) How to infer and analyze users' interests in a behavioral model? (2) Which pages of a web application are more interesting from the users' perspective? and (3) Which pages of an application may have design limitations?

Later we show that the web page's reward (user interest) value [34], [35] can be mapped to the popularity of a web page.

In this research, we provide an automated approach [36] to generate reward augmented behavioral models to answer these questions. This research contributes to current research in behavioral models in the following ways:

- It provides a new method to infer probabilistic user behavioral models using Markov models and Reinforcement Learning (RL) to automatically and incrementally, learn reward values.
- It is applied to and evaluated on a mobile and desktop application (www.ualberta.ca). It is believed that this is the first time such behavioral models have been applied to a mission-critical, real-world large-scale application.

The chapter is organized as follows: In Section 3.1 we explain our research motivations while Section 3.2 reviews related work. The steps of the proposed approach are overviewed in Section 3.3. A detailed description of our approach is provided in Section 3.4. Section 3.5 discusses an empirical evaluation of the proposed approach while Section 3.6 explains the implications for practice and

Section 3.7 concludes the paper and provides some future work directions.

3.1 Problem Statement and Research Motivation

User behavior models, generated from navigational patterns, provide solutions to several software engineering problems. A navigational pattern is a record of where a user visits; the pattern is extracted from the start to the end of a user session. The derivation of such a model is non-trivial and has seen many previous attempts [34], [35]. This research seeks to improve on those previous attempts in several ways which are outlined below.

- Ideally, a user behavioral inference process should not be required to instrument the application's web pages to generate a user behavioral model. To instrument a web page, the source code of the program is modified with additional commands. The purpose of instrumenting web applications is data collection. Providing a non-instrumented inference approach is preferable if it can achieve appropriate results. In this work, Google Analytics is used as an example of a state-of-the-art instrumentation-based approach.
- Web application evolution can be done by upgrading an application already in service or by releasing a new version or derivative. It is required that any model generation approach supports the evolution of web applications. This can help sustain web applications.
- When users interact with a web application, the history of their requests is stored in server logs. In large-scale web applications, log files will have millions of entries per day. Any model generation approach must support the utilization of such large-scale data entries.

The following paragraphs provide further discussion on how each of these items is essential in a user behavior model inference process.

Behavioral models are normally generated by collecting data either from the client's system instrumentation or the server's log files. To instrument a web page, it is necessary to insert additional code fragments into the source code. However, this can be difficult when the source code is inaccessible. Many software systems still in use were developed using technologies that are now obsolete; these legacy systems may continue to be used for a variety of reasons. It is not always possible to comprehend and take over source code in case instrumenting is problematic. Legacy code can be very difficult to read since it is written without exploiting modern programming techniques. These outdated approaches lead to lengthy and complex code that is difficult to understand and instrument. Another likely scenario is when an application is being licensed to a company without providing the source code. In such a case, instrumenting the source code is not feasible. However, it is still possible to generate behavioral models for these systems from their log files.

Google Analytics is a state-of-the-art approach to infer user behavioral models. However, it should be noted that generating such models using Google Analytics needs instrumenting of the source code. Google Analytics is a web analytics service offered by Google that tracks and reports website traffic. It can also be used to identify poorly performing pages preventing a web application from reaching its defined goals. Goals are used to measure how well the web application is targeting the predefined objectives like sales, lead generation, viewing a specific page, or downloading a particular file. Since 2019, Google Analytics is the most widely used web analytics service on the web [37]. It is currently used by 29,134,826 live websites, for instance, Google, YouTube, and Twitter use Google Analytics [38]. Also, it is used by 85.56% of the top 10,000 most popular websites. In addition, among the top 100,000 and a million most popular websites, 85.18% and 62.28% use Google Analytics, respectively [38]. To evaluate the correctness of our proposed inference approach, we considered the compatibility of our approach with the results extracted by Google Analytics from an instrumented website. (Further details are provided in Section 3.5 which discusses the evaluation of the proposed approach on a large-scale case study and presents the empirical results.)

A model generation approach needs to support the evolution of web applications. A behavioral model should be generated incrementally during its evolution and should play a role in the application's evolution (upgrading) procedure. Consider an application in which, a specific link needs to be added to the main page. By adding the link, a new behavior model should be generated incrementally. This approach makes the model generation process quick, flexible, and possible in the early phases of the application life cycle. This may become clearer in the context of a simple example. As an example, consider a car dealership web application. By analyzing its users' behavioral model, pages that have more visitors will be determined. Adding some related ads which show information about car insurance or car appliances would be helpful for customers. Therefore, related advertisements will be added to the target pages. Subsequently, a new model will be created incrementally.

Analyzing the model also helps to detect design anomalies. For instance, there might be some pages, in which users are being prevented from leaving the page without closing it. Such pages are called dead-ends [39]. Detecting dead-ends can significantly help in addressing design anomalies and providing solutions to retain users. This is because the difficulties in navigating through the web application can result in losing the audience. Therefore, incrementally generating behavior models and analyzing them regularly play an essential role in the evolution of a web application.

The more the developers of the applications know about their users' needs and interests, the smarter choices they will make for their application's development. Large-scale web applications use behavior models to investigate how engaged users are with content and identify potential content issues. For instance, imagine a commercial company's web application that has a huge number of entries per day. By analyzing and tracking its behavioral model, it can be detected that users prefer to see "maintenance instructions" about a product on the same page that the product is placed on, or they prefer to click on the "maintenance instructions" link to see the information on a separate page.

Developers can easily address such issues and increase users’ satisfaction levels. Satisfied customers are more likely to stay with the company and contribute to its success. Google, Microsoft, and Facebook are examples of large-scale companies, which are using behavior model generation approaches to understand the behavior of their users [40-45].

This work automatically generates user behavioral models from the application’s log files; it does not require instrument the web pages. The proposed approach supports large-scale mobile and desktop applications. Besides, the evolution of the application is supported by our method. In this study, the user behavior model is a model that infers and analyzes users' interest from the users' interaction with an application.

is a model that generates and analyzes users’ navigational behaviors (trajectories) and is necessary to understand users’ interests. By analyzing the inferred behavior models, developers can identify interesting or problematic pages from applications; and therefore, improve the application’s design. It should be noted that our approach is not about the logfile analysis. It only uses the logfile’s data to generate behavior models.

3.2 Related Work

Mining of web application usage discovers user navigation behavior based on their interests. The goal of web usage mining is to capture, model, and analyze usage patterns and understand the behavior of users through the process of data mining of web access data [46]. This has been used for a variety of purposes, including personalized recommendation, user intention prediction, detecting problematic pages, web design enhancement, etc. Some papers utilize user navigational behavior in mobile applications. For example, [12], [13] exploit application usage records to characterize patterns and discover user behavior. Zhu et al. [47] propose a recommendation model by recording these usage patterns. The model considers the popularity of applications and the personal interests of users. Lu et al. [48]

propose a framework for behavior mining in an application. It discovers users' spatial and temporal usage patterns from users' trajectory histories.

Other papers model mobile and desktop application usage patterns. For example, Kawazu et al. [14] propose an analytical method to classify web user behavior based on user interest. Wang et al. [15] propose a clickstream model to characterize user behavior in online services. This helps service providers identify unexpected user behaviors and predicts future actions. It uses a Markov chain to analyze click transitions. Vassio et al. [49] propose an approach to model user navigational behavior using clickstream graphs. Usage patterns are characterized by taking into account both the temporal evolution and the impact of the device used to explore the web pages [49].

The following papers use web usage mining on desktop applications. For example, Samarasighe et al. [16] predict user intention using a KNN classifier method from the user's historical behavior. User behavior patterns are predicted and analyzed using a decision tree by Prakash et al. [17]. Gan et al. [50] propose a model to adaptively learn users' interests from their navigational behaviors. This emphasizes recent click-through behavior via a neural network architecture. Chu et al. [51] build a model that can predict user interest from browsing history. The method is inspired by the Word2Vec (word embedding) concept. Langhnoja et al. [52] study user behavior from access patterns captured in a weblog. Web usage mining includes three phases namely pre-processing, pattern discovery, and pattern analysis. Their method combines the technique of clustering and association rule mining to extract significant user behaviors. Jagli et al. [53] describe web usage mining to analyze the behavioral patterns and profiles of users interacting with a Website. The discovered patterns are represented as clusters that are frequently accessed by groups of visitors with common interests. Zhou et al. [54] introduced the concept of utility in the web path traversal mining model to express the significance of web pages in terms of browsing time spent by the user.

Furthermore, user interest is discovered using the browsing patterns by Lei et al. [55]. This approach demonstrates that the browsing time and the number of visits are two key behaviors revealing the user interest in web pages [55]. A user interest model is proposed using the browsing behavior collected by a self-written browser plug-in [56]. A decision tree that uses linear regression at leaf nodes [56] is used to analyze the users' browsing histories and calculate the user interest value. A user behavior model is generated focusing on how users navigate the web pages [57]. Additionally, Xuefeng et al. [58] propose a personalized recommendation model from weblogs using a fuzzy cluster algorithm. Association rules are mined to provide a recommendation model for users of a website.

Probabilistic Markov models of navigational behaviors are generated from the interaction history given in the form of a log file by Ghezzi et al. [33]. It manually "annotates" the states of the models with numerical values that represent rewards. Rewards indicate the impact of the state on some (undefined) metric of interest.

All of these approaches represent methods for web application usage mining to discover user navigation behavior. However, according to Table 3.1, these studies do not automatically calculate user interest value in mobile applications.

Our approach dynamically generates a set of probabilistic Markov models from the users' interactions with a large-scale application and automatically augments the state of the models with reward (user interest) values using an RL strategy. Rewards can be mapped to the popularity of a web page. These pages tend to focus on particular contexts (topics) that users are more interested in. Later we show that higher reward values are assigned to popular pages.

To show that our approach is assigning meaningful values to the model, we use Google Analytics as a standard. We consider it to be state-of-the-art amongst commercial applications deriving behavioral information. Additionally, our approach uses an automatic model analyzer to check and analyze the generated models. Our work stands out in three factors from previous work, represented in Table 3.1. Firstly, it is believed that this is the first time such behavioral models have been applied to a mission-critical, real-world large-scale mobile and desktop application. We have used 70,595 users' records to analyze our approach.

Secondly, it does not require any instrumentation of a web page. Finally, our method supports the evolution of an application. Although some of the related works represented in Table 3.1 consider some of these mentioned features, they do not include all of them together. Moreover, the previous works do not check and analyze their models using an automatic model analyzer. Our proposed approach considers all of the features represented in Table 3.1 which results in automatic user behavior models' generation from a real-life, mission-critical, large-scale mobile and desktop application that supports the evolution of the application.

Table 3.1 provides additional details and a comparison of the related work. It specifies both mobile and desktop applications. "Real-life" denotes whether an application is real or not. (N/D is used when it is not clearly defined in the paper.) A "mission-critical" application is an application that is essential and must function continuously for a business or segment of a business to be successful. The "number of users" indicates the number of users of the application.

Furthermore, it is shown whether or not a method uses an instrumenting approach. "User Interest" indicates whether an approach calculates the users' interest value of visiting a web page. It also specifies the methods automatically or manually calculated these values. Finally, the table includes whether a method supports the evolution (or maintenance) of an application.

Table 3.1: Comparison of Properties of Different Methods of User Behavior Modeling

Methods	**Application**		**Dataset**			**Instrumenting**	**User Interest**		**Evolution**
	Mobile	**Desktop**	**Real-life**	**Mission-critical**	**#Users**		**Yes/No**	**Automatic**	
Silva et al. [12]	Yes	No	Yes	N/D	5342	Yes	No	No	No
Lim et al. [13]	Yes	No	Yes	No	3500	No	No	No	No
Zhu et al. [47]	Yes	No	Yes	No	13000	Yes	Yes	No	No
Lu et al. [48]	Yes	No	Yes	N/D	64	Yes	No	No	No
Kawazu et al. [14]	Yes	Yes	Yes	No	N/D	Yes	No	No	No
Wang et al. [15]	Yes	Yes	Yes	N/D	N/D	N/D	N/D	N/D	N/D

Jagli et al. [53]	Langhnoja et al. [52]	Chu et al. [51]	Gan et al. [50]	Prakash et al. [17]	Samarasighe et al. [16]	**Methods**		Vassio et al. [49]
No	No	No	No	No	No	**Mobile**	**Application**	Yes
Yes	Yes	Yes	Yes	Yes	Yes	**Desktop**		Yes
Yes	Yes	Yes	Yes	Yes	Yes	**Real-life**	**Dataset**	Yes
N/D	N/D	N/D	N/D	N/D	N/D	**Mission-critical**		Yes
240	N/D	N/D	N/D	N/D	N/D	**#Users**		N/D
Yes	N/D	Yes	N/D	No	Yes	**Instrumenting**		Yes
No	No	Yes	Yes	Yes	No	**Yes/No**	**User Interest**	No
No	No	Yes	No	Yes	No	**Automatic**		No
No	No	N/D	N/D	No	N/D	**Evolution**		Yes

Current work [36]	Ghezzi et al. [33]	Xuefeng et al. [58]	Vassio et al. [57]	Luo et al. [56]	Lei et al. [55]	**Methods**		Zhou et al. [54]
Yes	No	No	No	No	No	**Mobile**	**Application**	No
Yes	Yes	Yes	Yes	Yes	Yes	**Mobile**		Yes
Yes	No	N/D	Yes	N/D	Yes	**Real-life**	**Dataset**	Yes
Yes	No	N/D	Yes	No	Yes	**Mission-critical**		No
70595	N/D	N/D	N/D	N/D	N/D	**#Users**		5446
No	No	N/D	Yes	Yes	Yes	**Instrumenting**		N/D
Yes	Yes	Yes	Yes	Yes	Yes	**Yes/No**	**User Interest**	Yes
Yes	No	Yes	Yes	Yes	Yes	**Automatic**		No
Yes	No	N/D	N/D	No	N/D	**Evolution**		No

3.3 Augmenting Behavioral Models by Reward Values

In this section, a short introduction about the current user behavioral model generation procedures is given to provide an overview of our inference approach.

3.3.1 User Behavioral Model

Users behave differently in their interactions with web applications. They browse web pages based on their interests. Therefore, providing a model representing possible user behavior and latent patterns with a graphical and traceable representation can be helpful. Several techniques are suggested to track users' actions and generate models containing the paths users have taken through a website. Some of these approach instrument web pages to collect user interaction history, while others mine server-side log files to extract interaction patterns.

Moreover, inferred models are also represented in various ways from tree-based data structures [59] to different types of probabilistic models [33], [4], [60]. To allow the universal application of this approach, we only assume the availability of server-side log files as the system input. This assumption also implies that the system requires no modification of the existing configuration, which is often a barrier to adoption. In terms of the output from the system, we are extracting probabilistic (state-oriented) models, analyzing them, and augmenting their states with reward values to accurately represent the user behavioral patterns.

3.3.2 Proposed Model Inference Approach

Our framework was designed and implemented to incrementally generate reward augmented user behavioral models for large-scale web applications and to overcome the limitations of former approaches. The main steps of the proposed framework are shown in Figure 3.1 and briefly discussed in the following paragraphs. It is worth noting that steps 2 and 3 are intertwined and do not occur sequentially, but for the sake of clarity they are explained separately. Each step will be elaborated upon in a separate section.

1) **Identifying the initial parameters**: each row of a log file includes a requested URL submitted by a user. At the first step, a set of Atomic Propositions (APs) is used to associate semantics to the URLs occurring in the data entry. APs can be defined by the system expert or automatically by considering the URL of the page as a proposition. Also, a system expert can define a set of user-classes to characterize different groups of users. For example, users' agents (internet browsers used to view the web pages) and locations could be considered as two user-classes. Classes categorize users based upon a set of common features. However to automatically infer a reward-augmented model, which is not limited to a specific scope, defining user-classes can be ignored. Also in this step, input data are processed and classified. Each of them is clustered into groups univocally identified by the sets of atomic propositions.
2) **Generating the behavioral model**: the model inference engine analyzes the processed entries and generates a Discrete-Time Markov Chain (DTMC) for each "user-class", defined in the previous step.
3) **Calculating and assigning reward values**: concurrently with generating each state of the DTMC, the corresponding reward value is also incrementally calculated using a reward function, and assigned to the state. An RL-based approach is applied to automatically estimate the reward values for each state (i.e. web page).
4) **Analyzing the model**: when the DTMCs are generated and annotated with reward values, the analysis engine evaluates the properties of the interaction patterns against the inferred models using probabilistic model checking. The probabilistic model checker not only evaluates the correctness or

incorrectness of a property but also provides insights into the users' behaviors and the impact of these behaviors on the reward values [39].

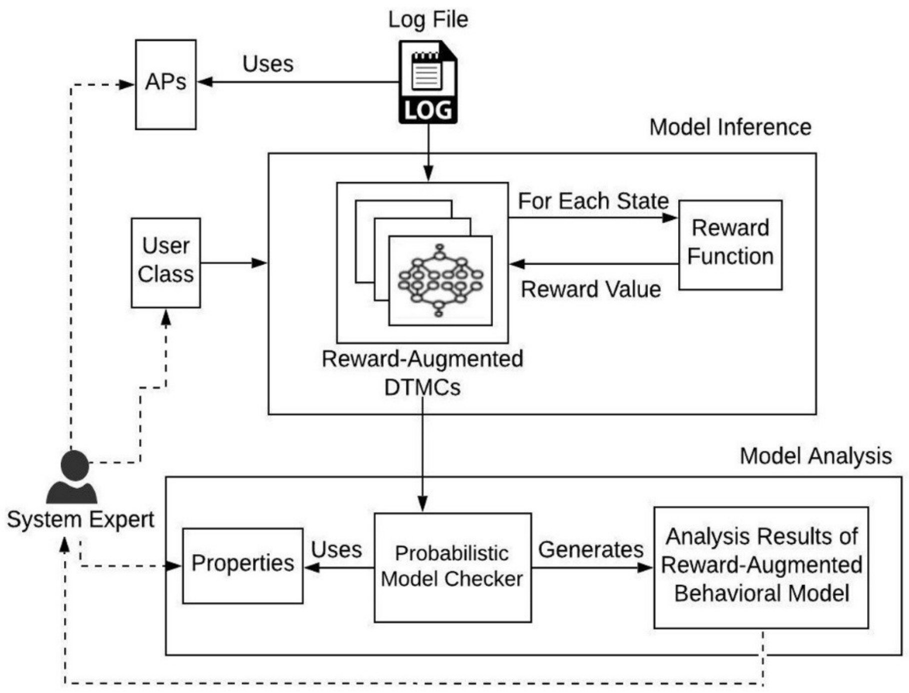


Figure 3.1: The Framework of the User Behavioral Model Inference Approach

3.3.3 Running Example

To define and demonstrate the proposed approach throughout the work, we introduce a real-world, large-scale mobile and desktop computing application called "MyUAlberta". This application includes several features for helping students and staff at the University of Alberta, enabling them to gain access to campus-related information through an easy-to-browse dashboard. The application has been active since September 2014 and more than 100000 app installs and more than 40000 monthly page views have been reported since then.

Users can view university news, events, and maps; and search for people who are registered as students or staff at the university. Students and academic staff can view course details including seat availability and check their timetable. These features along with several other features are represented in the MyUAlberta application.

This application has 58,498 iOS and 12,097 Android registered users with unique mobile devices. It is worth noting that these numbers do not include web users who only use the application through the web portal. Also, the application contains 18 main modules; each provides a link to an external resource (e.g. link to the university website) or represents an in-app feature. As an example of an in-app implemented feature: students can use this application to log in to the university authentication system and see their class schedule, and course lists (Registrar module). They can also review course content and take quizzes online using the eClass module. Library, Events, Student Services, Find a Person, ONEcard, News, Athletics, Social Media, and Photos and Videos are other modules that provide different services to university students and staff. Transit and Campus map modules are also two popular features in this application; they help users find out the departure time for several bus and LRT routes, as well as the campus-wide geographical map. Users can receive emergency push notifications or send feedback about their experiences in browsing the application. Therefore, according to the large scale of the application and its numerous users, analyzing the behavior of the users would provide information that could lead to the fruitful future of the application in terms of improvements.

To examine our approach we use the server log files of this application. We extracted the server log files of two different months (January and March 2018), each composed of more than 120,000 lines. It can be easily anticipated that the generated behavioral model using such log files would be massive and complex, so manually extracting the behavioral patterns and assigning the reward values to them is neither possible nor accurate.

3.4 Inference Details

In this section, we elaborate on the inference steps briefly discussed in the last section.

3.4.1 Identifying Initial Parameters

To infer user behavioral model, the inference approach needs a list of the interactions between the users and the webserver of the application in the Common Log Format (CLF), where each row represents a request submitted by a user to the webserver and contains the IP address, timestamp, requested URL and client's device information.

In large-scale web applications, log files will have millions of entries per day. Web usage mining offers system administrators and web designers the ability to analyze the massive volume of log files [61]. It targets minimal human intervention for log file processing [62]. Web usage mining has three main stages: data preprocessing, pattern discovery, and pattern analysis. Data preprocessing involves the removal of unnecessary data. Pattern discovery includes data mining techniques to extract usage patterns from log files [61, 63]. Pattern Analysis extracts the interesting patterns from the output of the pattern discovery process by eliminating the irrelative patterns [61, 64]. In web usage mining, users' behavior or interests are revealed by applying data mining techniques on log files [63]. A preprocessed log file improves the efficiency and effectiveness of the other two steps of web usage mining such as pattern discovery and pattern analysis [63, 64].

Due to a large number of irrelevant entries in a log file, the original log file cannot be directly used in the web usage mining process [63, 64]. Raw log files have some rows that belong to resources that are irrelevant to the users' interactions with web applications.

Our proposed inference approach clusters each row of the log file into the groups identified by a set of Atomic Propositions (APs). It uses several code fragments called filters to indicate the set of atomic propositions, which can be associated with the relevant requested URLs in the log files. Filters are parameterized with a regular expression to only identify the URLs matching the expression. For example, the proposition "home" is going to be used as a label for a row in the log file, which contains the requested URL that will lead the user to the homepage of the application. This procedure helps in: (1) Identifying the requests corresponding to the same URLs and clustering them into the same group. And, (2) detecting and filtering out rows that belong to CSS source code, JavaScript source code, or any resources that are irrelevant to the users' interactions with the web application.

Our inference framework also contains two default classifiers to classify users based upon (1) the user-agents (e.g. Firefox) and (2) the users' location extracted by geolocating the IP addresses [33]; more classifiers can be easily added. Classifiers help designers/analyzers to extract domain-specific information about the users by classifying users into several different customizable classes. For example, using this approach, we would be able to analyze specific users' behavioral patterns for clients who logged into the application using Chrome.

3.4.2 Generating the Behavioral Model

In the model inference step, the inference engine incrementally generates a set of Discrete-Time Markov Chains (DTMCs). DTMCs are probabilistic finite-state automata, which follow a Markovian property and represent the users' behavioral patterns. They are suitable options for representing user behavioral models, because:

- The transition from one state to another state in the model only depends on the current state. Therefore, in a user behavioral model, the probability distribution of the next page (the user might visit), only depends on the links and content provided in the page that the user is currently browsing. This perfectly matches the user behavior pattern definition, which illustrates the users' movement flow from one state (page) to another.

- The system evolves through discrete time steps. In user behavioral modeling, we are interested in analyzing user behaviors at discrete time intervals to predict the next movement of the user solely based on the current state. Therefore, changes to the system cannot occur at any time along with a continuous interval.

In our study, DTMCs are also annotated with a numerical value called a reward. Rewards indicate the quantitative value (benefit) of visiting a specific page on the website or being in a specific state of the model. In research conducted by Ghezzi et al. [33], rewards are manually determined and assigned by the system designer to the states of the models. (It is argued that such a manual approach is unrealistic given the huge number of decisions that would be required to be made continuously.) Accordingly, a DTMC which is augmented with rewards is a tuple (S, P, L, ρ) where:

- S is a set of states, and $s_0 \in S$ indicates the initial state;
- $P: S \times S \rightarrow [0,1]$ is the probabilistic matrix indicating the probability of the occurrence of a transition between two connected states.
- $L: S \rightarrow 2^{AP}$ is a function which maps a state to a set of atomic propositions.
- $\rho: S \rightarrow \mathbb{R}_{\geq 0}$ is a reward function that associates a non-negative number to each state.

In this study, we also infer a DTMC for each class defined by the classifiers.

To start the model inference process, an initial DTMC is generated. The initial DTMC consists of (1) two initial states: s_0 (start state) and s_e(end state); (2) a zero transition matrix P indicating the probability transitions between states; (3) a set of state labels indicating the start and end labels $L = \{start, end\}$ and (4) a reward function ρ which assigns 0 to both start and end states as an initial reward value. Assigning 0 as a reward value to the initial states illustrates that the value of states is not calculated yet. Then, the initial DTMC will be incrementally developed by processing each row (r) of the log file and adding more states and transitions to the model. The following paragraphs provide more details about this procedure:

- First, when a row r is processed, the algorithm assumes that the IP address in the row corresponds to a new user unless the IP address has been previously detected within a predefined time-window. A time window is the minimum time span between timestamps that is defined by the system expert to identify the requests that are issued by two different users but the same IP addresses. When the time-window for a certain IP address expires, the algorithm assumes that the user associated with that IP address left the system. In this study, we assumed that the time-window is equal to 1 minute, which is equal to the minimum session timeout in Google Analytics. This should not be confused with the default session duration in Google Analytics, which is equal to 30 minutes.
- If the previous step considers r as a request issued by a new user, the algorithm adds a new state to the model and labels it by the set of propositions associated with r. At this point, it considers the start state s_0 as its parent state. But if the request belongs to a known user, the algorithm still builds the new state with the same labels but considers the latest state associated with the previous request as its parent state.
- Then the transition probability p_{ij} is assigned to the transition between s_i and s_j. p_{ij} is equal to the ratio between the number of transitions between s_i and s_j, and the total number of transitions with the source state s_i [65].
- During the inference procedure, if the time-window expires for a certain IP address, the algorithm generates a transition from the latest discovered state to the end state s_e and updates the transition probability.
- The previous steps will be repeated until no new request is found in the log file.

Table 3.2: An Excerpt of the Myualberta Log File

```
1.1.1.1 - - [24/Jan/2018:06:50:56 +0700] "GET /home
1.1.1.1 - - [24/Jan/2018:06:51:00 +0700] "GET /trnst
1.1.1.1 - - [24/Jan/2018:06:51:04 +0700] "GET /social
```

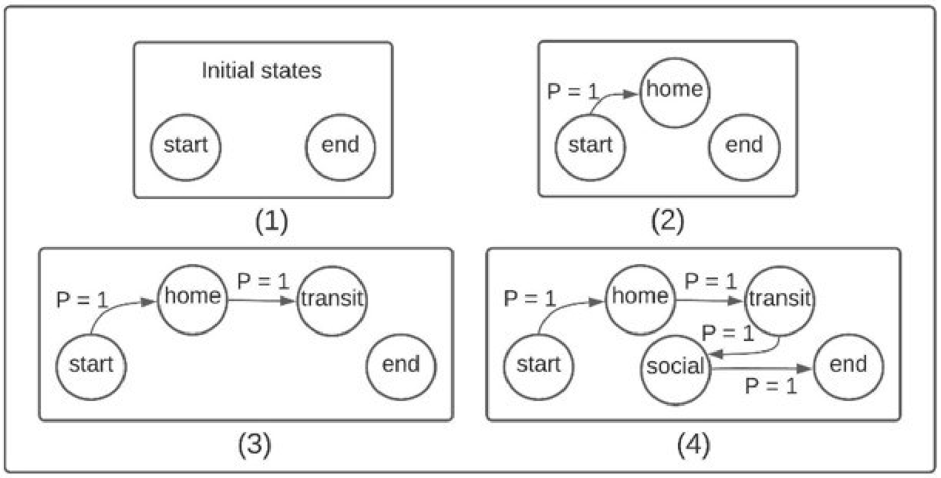


Figure 3.2: An Excerpt of the Model Inference Procedure for Myualberta Study Instance

Table 3.2 shows an excerpt of the MyUAlberta log file containing users' IP addresses, the requests' timestamps, and the requested URLs. This log file represents the interaction of a user with the application.

Figure 3.2 depicts implementing the above step on the MyUAlberta study instance. In this case, as the log file is processed, the initial DTMC is generated by building start and end states (See Figure 3.2 section (1)). When the first line of the log file is read, since this is the first time a request to the homepage is getting processed, a new state labeled home is generated. Since the IP address of this request has not been already encountered, the algorithm connects it to the start state (Figure 3.2 section (2)). At the same time, the inference engine assigns a probability of 1 to this transition. This is because there is no other state with the source of start state yet. As the algorithm processes the second row of the log file that contains the same IP address, it adds a new state labeled transit to the DTMC and considers the home state as the source state for it. The transition probability of the new connection is also 1 (see Figure 3.2 section (3)).

The third row of the log file again belongs to the same IP address but containing a request to load the social media page. Therefore, the algorithm adds the new state called social to the model in the same way as previous states and connects it to the transit state (see Figure 3.2 section (4)).

When the time-window for a specific IP address expires, the algorithm assumes that the user left the application. Therefore, it generates a new transition that connects the latest discovered state (which is associated with this user) to the end state. Figure 3.2 section (4) depicts this step.

During the inference process, the reward values are also calculated and assigned to the states of the model, but for the sake of clarity, we present the remaining steps in the next section.

3.4.3 Calculating and Assigning Reward Values

As mentioned previously, the necessity for manually producing the reward signal is a major problem with previous inference approaches, such as [33].

To illustrate the use of rewards in user behavioral models clearly, we first provide an example related to a sell-and-buy website. Assuming that the goal of the website is to increase the number of advertisements, the designer can assign reward values to states by considering the number of advertisements on each page. For example, if there are 10 advertisements on the homepage, the system expert associates the reward value 10 with the proposition homepage. Accordingly, other states of the model also get annotated by the number of advertisements their corresponding proposition contains. Depending on the website’s goals and missions, designers only define one technical or non-technical metric of interest and assign rewards based upon this metric only once during the setup phase of the inference procedure. Therefore, to recalculate reward values based on the new metrics: (1) system experts should recalculate reward values, and (2) models should be regenerated. In such situations, defining an approach that can represent the reward values of the states, from a general perspective, can be very useful. Therefore, automating the calculation process makes the approach more effective specifically for large-scale software systems.

The following paragraphs outline our proposed technique to solve this issue. We utilize an RL strategy to automate the estimation of the rewards signal. Therefore, we first provide a quick background about the applied techniques and definitions.

3.4.3.1 Reinforcement Learning

This section provides an overview and definition of Reinforcement Learning.

RL is located between supervised and unsupervised learning to learn what to do to maximize a numerical reward signal [66]. The learner does not know what actions to take to reach the goal of maximizing the reward signal and only can pick and try actions to detect those increasing the accumulative reward [67].

Reinforcement learning algorithms are defined iteratively not by characterizing learning methods, but by characterizing a learning problem. The agent and the environment are interacting continually: the agent selects actions and the environment responds to the actions, presents new states to the agent which gives rise to rewards. This cycle is repeated as part of a Markov Decision Process (MDP) [66], [68]. MDPs are used as stochastic extensions of finite automata or Markovian processes to model the decision-making process and solve optimizing problems. They are augmented by actions and rewards so that they consist of actions, transitions, labels, and states. In the following paragraphs, some definitions are introduced that help demonstrates our proposed approach in the next section.

Definition 1. Markov decision process (MDP)

MDPs provide a mathematical framework to model the decision-making process in situations where outcomes are partly random and partly under the control of a decision-maker. They are useful in addressing a wide range of optimization problems solved via dynamic programming and reinforcement learning [66]. MDPs are an extension of Markov chains; the difference is the addition of actions (allowing choice) and rewards (giving motivation). Conversely, if only one action exists for each state and all rewards are the same (e.g. "zero"), an MDP reduces to a Markov chain.

More precisely, an MDP is a discrete-time stochastic control process. In other words, an MDP contains:

1. A set of possible states S.
2. A set of possible actions A.
3. Transition function, $\mathrm{T}: \mathrm{S} \times \mathrm{A} \times \mathrm{S} \rightarrow [0,1]$ giving for each state and action. It computes the probability of reaching state s' by performing action a in state s and is denoted as $\mathrm{T}(\mathrm{s}, \mathrm{a}, \mathrm{s}'),\ \mathrm{s}, \mathrm{s}' \in \mathrm{S}, \mathrm{a} \in \mathrm{A}$.
4. Reward function, $\mathrm{R}(\mathrm{s}, \mathrm{a}, \mathrm{s}')$ specifies rewards for transitioning from state s to state s' due to action a.

Therefore, an MDP has a set of states. These states will play the role of outcomes in the decision-making approach as well as providing information, which is necessary for choosing actions.

An MDP also has a set of actions. At each time step, the process is in state s, and the decision-maker may choose any action a that is available in state s. The process responds at the next time step by randomly moving into a new state s', and giving the decision-maker a corresponding reward $\mathrm{R}(\mathrm{s}, \mathrm{a}, \mathrm{s}')$.

To indicate the order of different states and actions during agent and environment interaction, they should be denoted according to the time at which they occur. So, $\mathrm{s}_\mathrm{t} \in \mathrm{S}$ denotes the state at time t [69]; according to this definition of a Markovian process, we would have:

$$\mathrm{P}(\mathrm{s}_{\mathrm{t}+1} | \mathrm{s}_\mathrm{t}, \mathrm{s}_{\mathrm{t}-1}, \mathrm{s}_{\mathrm{t}-2}, \dots) = \mathrm{P}(\mathrm{s}_{\mathrm{t}+1} | \mathrm{s}_\mathrm{t}) = \mathrm{T}(\mathrm{s}_\mathrm{t}, \mathrm{a}_\mathrm{t}, \mathrm{s}_{\mathrm{t}+1}) \qquad (3.1)$$

So, it specifies the probability that the next state will be $\mathrm{s}_{\mathrm{t}+1}$ for each state $\mathrm{s}_\mathrm{t} \in \mathrm{S}$ and action $\mathrm{a}_\mathrm{t} \in \mathrm{A}$. The probability that the process moves into its new state is influenced by the chosen action.

This process is called Markov because it has what is known as the Markov property; that is, the next state s' depends on the current state s and the decision maker's action a. Therefore, the next state is independent of all the previous states and actions. The current state captures all that is relevant about the world to predict what the next state will be.

Finally, there is the Reward function. The agent's utility is defined by the reward function. The goal of the reward function is to specify the reward for each action that the agent performing. So, $\mathrm{R}: \mathrm{S} \times \mathrm{A} \times \mathrm{S} \rightarrow \mathbb{R}$ gives rewards for particular transitions between states. The reward function plays an important role in an MDP. At each discrete time, an agent observes state $\mathrm{s_t} \in \mathrm{S}$ and chooses action $\mathrm{a_t} \in \mathrm{A}$ and receives immediate reward $\mathrm{r_t} \in \mathrm{R}$. Then state changes to $\mathrm{s_{t+1}}$. The immediate reward indicates the immediate feedback value of reaching a certain state. An agent tries to maximize its total reward in the long term view. So an MDP can be denoted by the tuple $\langle \mathrm{S}, \mathrm{A}, \mathrm{T}, \mathrm{R} \rangle$ depicting it as a state transition graph [68].

Definition 2. Policy

The goal in an MDP is to find a function, called policy, which determines which action to take in each state, to maximize the reward function. Policy π gives the probability of taking action a when in state s:

$$\pi: S \times A \rightarrow [0,1]$$

$$\pi(s, a) = P(a_t = a | s_t = s) \tag{3.2}$$

Definition 3. Next-State Function

At each discrete time, an agent receives an immediate reward $\mathrm{r_t}$ by choosing action $\mathrm{a_t}$ from state $\mathrm{s_t}$ and then the state changes to $\mathrm{s_{t+1}}$. Markov assumes that $\mathrm{s_{t+1}} = \delta(\mathrm{s_t}, \mathrm{a_t})$ and $\mathrm{r_t} = \mathrm{R}(\mathrm{s_t}, \mathrm{a_t}, \mathrm{s_{t+1}})$. $\mathrm{r_t}$ and $\mathrm{s_{t+1}}$ depend only on the current state and action. The next-state function δ for state s maps the state s at time t to the state at time t+1 that follows the action. The value of reaching state s through action a is computed using the action-value function which is defined in the following paragraph. The next-state function δ and reward function R may be nondeterministic and not necessarily known to the agent.

If the next-state function is known, dynamic programming can be used to learn value functions like V(S), defined below, to obtain the policy. If the next-state function is unknown, Q-Learning can be used to learn action-value functions like Q(S, A). The value function is described in the following paragraphs.

Definition 4. State-Value Function

The state-value function returns the value, i.e. the expected return for selecting a certain state s. Return means the overall reward. We can define the value of a state under a policy π, formally $V^{\pi}(s)$, as [68]:

$$V^{\pi}(s) = E_{\pi}\{ r_t + r_{t+1} + r_{t+2} + \cdots | s_t = s\} \tag{3.3}$$

The sum in equation (3.3) is typically infinite. In a more realistic setting, rewards in the future get discounted, and equation (3.3) becomes [67]:

$$V^{\pi}(s) = E_{\pi}\{r_t + \gamma\, r_{t+1} + \gamma^2\, r_{t+2} + \cdots | s_t = s\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s\right\} , \gamma \in [0, 1] \tag{3.4}$$

Where E_{π} is the expected value by following policy π and discount factor γ, which determines the importance of future rewards. The state-value function $V^{\pi}(s)$ specifies the value of a state is equal to the total amount of rewards a learner can accumulate, starting from that state. $\gamma = 0$ indicates that it is only concerned about immediate rewards.

Definition 5. State-Action Value Function

The state-action value function returns the value, i.e. the expected return for using action a in a certain state s as:

$$Q^{\pi}(s, a) = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s, a_t = a\right\} , \gamma \in [0, 1] \tag{3.5}$$

Where E_{π} is the expected value by policy π.

The Bellman-Equation expresses the relationship between the value of a state and the value of a successor state.

$$V^{\pi}(s) = \sum_{a \in A} \pi(s,a) \sum_{s' \in S} T(s,a,s')(R(s,a,s') + \gamma V^{\pi}(s')) \quad (3.6)$$

Where R is the reward for passing from state s to successor state s'. $V^{\pi}(s')$ returns the state-value of the successor state s'. T is the probability to access state s' when choosing action a in state s. $\pi(s,a)$ is the probability of choosing action a in state s. The Bellman-Equation calculates the value of a state by considering all available options and assessing each by its likelihood of appearance.

An agent aims to find the optimal policy. There is at least always one policy that is better than or equal to all other policies. This policy is called the optimal policy π^*.

$$\pi^* = \arg\max_{\pi} V^{\pi}(s), \forall(s) \quad (3.7)$$

There is only one optimal-value function, which is the base for an optimal policy. The optimal value function chooses an action a in a state s not by following a policy but by choosing the maximum. We attempt to learn the value function of the optimal policy π^*, denoted by V^{π^*} (which we write as V^*). It can then do a look-ahead search to choose the best action from any state s. The optimal-value-function can be noted as follows:

$$V^*(s) = \max_{a \in A} \sum_{s' \in S} T(s,a,s')\,(R(s,a,s') + \gamma V^*(s')) \quad (3.8)$$

As aforementioned, this works well if the agent knows next-state function $\delta: S \times A \rightarrow S$, and $R: S \times A \times S \rightarrow \mathbb{R}$. When the agent does not know δ, the Q-function which is very similar to V^* is defined as:

$$Q^{\pi}(s,a) = R(s,a,s') + \gamma V^*(s') \quad (3.9)$$

If the agent learns Q, it can choose an optimal action even without knowing δ.

$$\pi^*(s) = \arg\max_{a}[Q(s,a)] \quad (3.10)$$

Where Q is the action-value function the agent will learn. Q and V^* are closely related as:

$$V^*(s) = \max_a Q(s, a) \tag{3.11}$$

This allows us to write Q recursively as:

$$Q^{\pi}(s_t, a_t) = R(s_t, a_t, s_{t+1}) + \gamma V^*(\delta(s_t, a_t)) = R(s_t, a_t, s_{t+1}) + \gamma \max_a Q(s_{t+1}, a) \tag{3.12}$$

All RL-based algorithms are based upon providing an approach for appropriately estimating state-action value functions. This has led to the exploration and production of several different estimating methods and techniques. One of the most popular of these is Q-Learning [70], which is an off-policy Temporal Difference control algorithm. In other words, Q-Learning can estimate Q-value functions (Q-learning-based estimations of the state-action value function) without requiring an initial model of the environment.

Additionally, Q-learning can handle problems with stochastic transitions and rewards without requiring any adaptations. It has been proven that for any finite MDP, Q-learning eventually finds an optimal policy. This means that the expected value of the total reward that has been returned over all successive steps is the maximum achievable.

In this situation, because of the lack of known transition and reward models, the algorithm handles problems with stochastic transitions and rewards and uses exploration and sampling approaches to learn the required model. Therefore, Q-learning finds an optimal policy for any finite MDP and estimates the agent's Q-value function based on the Q-value estimation of an action. In other words, using the above definitions equation (3.13) is inferred. This process is incrementally evaluated as follows [68]:

$$Q_{k+1}(s_t, a_t) = Q_k(s_t, a_t) + \alpha\left(r_t + \gamma \max_a Q_k(s_{t+1}, a) - Q_k(s_t, a_t)\right) \tag{3.13}$$

Where, α $(0 < \alpha \leq 1)$ is the learning rate, which determines the extent to which new information can override old information and how fast we modify our estimates [71]. In the next section, we illustrate how we use Q-value functions to incrementally calculate reward values.

3.4.3.2 Automated Reward Calculation Algorithm

Since the Q-value function is proven to converge to an optimal policy [69] and estimate the state-action value function in model-free problems [70], we have used Q-learning to estimate the reward values in this study. The problem of estimating the reward values for the user behavioral model needs to be addressed using an approach, which: (1) can incrementally learn reward values from the current state of the model; and (2) can easily get configured to generate meaningful reward values for web applications. Therefore, a Q-value function is an appropriate option to address this issue.

To present a Q-value function, which fits the behavioral model generation process, we have modified the Q-value function (equation (3.13)) similar to [65] as below:

Equation (3.14) illustrates how the Q-value function has been used in this approach to calculate the value of state s_i, which is called $\rho(s_i)$:

$$\rho(s_i) = 1 - Similarity\ (pageA, pageB) + \gamma \max \rho(s_{i+1}) \qquad (3.14)$$

A discount factor γ determines the importance of future rewards.

In this study, we suggest using a function (diff) to calculate reward and then Q-values: Given two-state labels (web application pages), s_A and $s_B \in$ S, we define the diff function that computes the degree of change between states. diff can be defined as $1 - similarity\ (pageA, pageB)$ where, $similarity\ (pageA, pageB)$ is computed based upon one of the following approaches. A detailed description of the algorithmic process is described in Algorithm 3.1.

ALGORITHM 3.1: Reward Calculation Algorithm

Input: Model states $(s_i, s_{i+1}) \in AP$; $i = 0$; $\rho = 0$
Output: Reward value ($Reward \in \mathbb{R} \geq 0$)

```
begin
For each state $s_0$ To $s_{N-1}$ do
    urlA⟵StateMatcher($s_i$)
    urlB⟵StateMatcher($s_{i+1}$)
    pageA⟵Extractor(urlA)
    pageB⟵Extractor(urlB)
        $\rho(s_i) \leftarrow 1 - \textbf{Similarity}\ (pageA, pageB)$
    $max \leftarrow \max \rho(\delta(s_{i+1}))$
    $\rho(s_i) \leftarrow (\rho(s_i) + 0.9 \max)/100$
    For each $(s_i, s_j)$ ; $s_j \in S_e$ *
        merge $(s_i, s_j)$ if
        $(s_i = s_j)$ AND (Reward $(s_i)$=Reward $(s_j)$) AND
         (Adjacent $(s_i)$= Adjacent $(s_j)$) AND
         (Reward(Adjacent $(s_i)$)= Reward(Adjacent $(s_j)$))
    Repeat Until no new $s_j \in S_e$ is found
 Repeat Until no new $s_i \in S$ is found
end
```

* S_e is the set of states which are already labeled with the reward values

As it is mentioned, the reward calculation algorithm has been synchronized with the model-inference process. It is executed every time a new state is generated. Accordingly, the rewards are calculated and updated incrementally during the model generation process. This is essential as it allows the algorithm to work in real-time. Therefore, each step of the automated reward calculation algorithm is elaborated in the following paragraphs.

1. While states are created, a function (StateMatcher) searches in the regular expression library to find the requested URLs matching these states. For example, if the state label is "eclass", the method detects the corresponding URL that has been requested by the users; in this case, that is "www.eclass.srv.ualberta.ca/".
2. When URLs are retrieved, another method (Extractor) is called. This function extracts the information of the detected URLs. In this research, we used the following approaches for information extraction of a detected URL which are described in the next section.

- Visual-based
- Text-based (character or word–level)

3. After storing the extraction results, the difference between the two URLs' content is calculated using similar methods based on the type of information (visual-based or text-based) extracted from the requested URLs.
4. To eliminate any redundancies in the model, the merging step of the gkTail inference algorithm is applied to merge the equivalent states [72]. According to this state-merging procedure, two states are considered equivalent if they have the same future of length k (in our study k=1). Therefore, two states can be merged if they share the same label, rewards, and immediate future, which means their adjacent states also have the same labels and reward values. This procedure prunes the model of redundant states with the same values.

3.4.3.3 Reward Calculation Algorithm - Information Extraction of a Web page

As mentioned in Section 3.4.3.2, the Extractor function collects the information of the retrieved URLs. In this study, we used the following approaches for information extraction of a detected URL which are described in the subsequent paragraphs.

i. Visual-based (Visual information extraction)
ii. Text-based (character or word–level)

i. Visual Information Extraction

To extract the visual information of a web page, we use the semantic blocks in the web pages using Gestalt laws of grouping technique; this technique is described in detail in [28]. The Gestalt laws of grouping explain the mechanism of how humans perceive and understand things. To construct each block for the block tree, these laws need to be translated into computer-compatible rules using Gestalt laws translator. The laws are described in Chapter 2. We provide a quick background about the semantic blocks in the following paragraphs.

Semantic Blocks

The semantic blocks, nodes of a semantic block tree, are achieved by merging semantically correlated rendered blocks with the Gestalt laws of grouping [28]. A rendered-block tree model represents retrieved visual information of a web page by taking a web page's DOM tree as the input instead of parsing the source code. This is because the DOM tree contains all the visual information of a web page [28].

Two major ways, explored by researchers, to represent a web page for visual similarity evaluation are screen shots (images) and DOM trees. The advantages of the two methods are combined [73], and it proposes a new web page representation method, a block tree. It only extracts visible DOM elements and merges these elements into separate groups according to their semantic meanings.

To construct blocks, separate rendered objects are merged into semantically related groups based on the Gestalt laws of grouping [73]. Given a web page, WP, the rendered object maps to a visible DOM element *e* of the DOM tree DT of WP. The object contains all visible CSS properties of *e* as the visual features and serves as the merging candidates to build the blocks of the Block-Tree (BT) [73].

The Gestalt laws of grouping explain a human's mechanism for the perception of related objects. To construct each block for the block tree, these laws need to be translated into computer-compatible rules [26, 28].

Among all the six Gestalt laws, the first two show us how to extract rendered objects from the DOM tree, and the remaining four laws regulate the way of merging the extracted rendered objects into groups (that is, the blocks in the block tree) by the visual features [73].

The block tree takes the previously merged blocks as tree nodes and follows the DOM tree's hierarchy to organize these nodes. In the beginning, the first visible

DOM element is the “BODY”, so the root node of the block tree will be a block that holds it. Next, it follows the bottom-up rule. From the root block onwards, all the direct child rendered objects of a block are evaluated by the Gestalt laws and split into one or more groups. Each of the laws is then applied to create a block. These blocks will maintain their hierarchy in the DOM tree [73].

The Tree Edit Distance (TED) is defined as the minimum cost of editing operations (“insert”, “delete”, and “relabel”) when shifting from a block tree to another different block tree [74]. This reflects the structural similarity between two different block trees by mapping node pairs. The pseudo-code [74] represented in Algorithm 3.2 details this step’s calculations and is described in the following paragraphs.

Assume $T^p_{i,j}$ represents a subtree (block-tree) of T^p rooted at T^p_i which is mapped to an identical subtree (block-tree) of T^q rooted at $T^q_{j'}$, namely $T^q_{j,i}$. Accordingly, computing the extended subtree similarity $S(T^p, T^q)$ has four following steps according to [74].

Step 1: Identify all the mappings: It finds all the possible mappings and stores two lists of nodes for each mapping, one for each subtree. T^p and T^q are the inputs to this step and V^p and V^q are the outputs. The GetMapping(i,j) function produces two lists of nodes ($V^p[i][j]$ and $V^q[j][i]$) for a mapping. Its objective is to detect the largest possible mapping. T^p_{ia} denotes the ath child of the T^p_i node, where $1 \leq a \leq deg(T^p_i)$, and ia represents the index of the ath child of the T^p_i node. $deg(\mathrm{T})$ represents the degree of T, which is the maximum number of children of any node in the tree, and E is a matrix that indicates how the children of T^p_i and T^q_j are matched.

Step 2: Identify each node’s largest mapping: To compute this step, first, assume two arrays, namely LS^p and LS^q, of size $|T^p|$ and $|T^q|$, respectively. $LS^p[i]$ indicates the largest subtree that T^p_i belongs to by the indexes of root nodes of the mapping, denoted by $LS^p[i]_{mi}$ and $LS^p[i]_{mj}$. As indicated in Algorithm 3.2, filling LS^p and LS^q with appropriate values is the objective of this step.

Step 3: Compute the weight of each subtree: This step calculates $W\left(T_{i,j}^{p}\right)$ and $W\left(T_{j,i}^{q}\right)$ for all the subtrees in the mappings. In Algorithm 3.2, they are denoted by $W^p[i][j]$ and $W^q[j][i]$. It goes through LS^p and increases the weight of a subtree when it is reported as the largest subtree of a node in LS^p. This procedure is repeated for LS^q as well.

Step 4: Calculate $S(T^p, T^q)$: It can calculate $S(T^p, T^q)$ according to $S(T^p, T^q) = \alpha\sqrt{\sum_{m_k \in M} \beta_k \times W(m_k)^{\alpha}}$, where $\alpha, \alpha \geq 1$, is a coefficient to adjust the relation among different sizes of mappings. β_k is a geometrical parameter. Further, m_k is the weight of subtree k in the mapping [74].

To extract the visual content of two web pages, the Extractor function collects visual information and stores them in two trees ($pageA$, $page$B). To compare web pages and evaluate their visual similarity, we use a block tree edit distance (B-TED) [28]. Since a web page can be ultimately represented by a semantic-block tree (or simply block-tree), and each block contains all the visual information, visual similarity between two web pages can be reflected by B-TED [28].

ALGORITHM 3.2: Subtree Similarity Function Algorithm

Input: Method calls in tree format (T^p, T^q)

Output: Similarity Score

```
Step 1
Begin
  for $i = 1$ to $|T^p|$ do
    for $j = 1$ to $|T^q|$ do
      if $label(t_i^p) == label(t_j^q)$ then
        $GetMapping(i, j)$
      end of if
    end of for
  end of for
Step 2
  for $i = 1$ to $|T^p|$ do
    for $j = 1$ to $|T^q|$ do
      for $k = 1$ to $|V^p[i][j]|$ do
        $i' \leftarrow V^p[i][j]_k,\ j' \leftarrow V^q[j][i]_k$
        if $|V^p[i][j]| > |V^p[LS^p[i']_{mi}][LS^p[i']_{mj}]|$ then
          $LS^p[i']_{mi} = i, LS^p[i']_{mj} = j$
        end of if
        if $|V^q[j][i]| > |V^q[LS^q[j']_{mj}][LS^q[j']_{mi}]|$ then
          $LS^q[j']_{mi} = i, LS^q[j']_{mj} = j$
        end of if
      end of for
    end of for
  end of for
Step 3
  for $i = 1$ to $|LS^p|$ do
    $W^p[LS^p[i]_{mi}][LS^p[i]_{mj}] + +$
  end of for
  for $j = 1$ to $|LS^q|$ do
    $W^q[LS^q[j]_{mj}][LS^q[i]_{mi}] + +$
  end of for
Step 4
  for $i = 1$ to $|T^p|$ do
    for $j = 1$ to $|T^q|$ do
      $temp = \left(\frac{W^p[i][j] + W^q[j][i]}{2}\right)^{\alpha}$
      if $depth(t_i^p) \neq depth(t_j^q)$ then
        $temp = temp \times \beta$
      end of if
      $S = S + temp$
    end of for
  end of for
  $S = \sqrt[\alpha]{S}$
End
Step 1, $GetMapping(i, j)$ function
Begin of $GetMapping(i, j)$
  $V^p[i][j] = \{t_i^p\}$
  $V^q[j][i] = \{t_j^q\}$
  for a = 1 to $\deg(t_i^p)$ do
    for b = 1 to $\deg(t_j^q)$ do
      $E[a][b] = Max \begin{cases} E[a-1][b] \\ E[a][b-1] \\ E[a-1][b-1] + |V^p[ia][jb]| \end{cases}$
    end of for
  end of for
  a= $\deg(t_i^p)$
  b= $\deg(t_j^q)$
  while $a > 0$ and $b > 0$ then
    if $E[a][b] == E[a-1][b-1] + |V^p[ia][jb]|$ then
      $V^p[i][j] = V^p[i][j] \cup V^p[ia][jb]$
      $V^q[j][i] = V^q[j][i] \cup V^q[jb][ia]$
      $a = a - 1$
      $b = b - 1$
    else if $E[a][b] == E[a][b-1]$ then
      $b = b - 1$
    else
      $a = a - 1$
    end of if
  end of while
End
```

By encoding the content of each block into its label, the mapping procedure in TED calculation compares the blocks by their visual information [28]. In this research, we used the same properties as [28] to compare the visual similarity between two web pages.

ii. Text-Based Information Extraction

The following two approaches are used to extract a web page's context-based information.

- Character-based extraction of a web page
- Word-based extraction of a web page

Character-Based Extraction of a Web page

We used Crawljax [75] to extract the content of a web page as a String. Crawljax has a method for extracting elements from the DOM tree of a web page. It can extract candidate elements from the current DOM tree in the browser, based on the tags defined by the user. In this research, Crawljax is not used for deep crawling. It is used to extract characters from a web page's context. The Extractor function collects all words and links on the pages as characters and stores them in two Strings, ($pageA$, $page$B).

Algorithm 3.3 shows the similarity function works by calculating the Levenshtein distance between these Strings. The reason for applying the Levenshtein distance is its proven capability in measuring the similarity between two strings. The similarity function plays the role of the Q-value function to initiate the Q-value ($\rho(s_i)$) for state s_i, so when the user sends another request URL from the current state (the web page the user is currently viewing), the algorithm calculates the Q-values of upcoming states (web pages) and chooses the maximum amount to learn the current states' reward.

ALGORITHM 3.3 : Similarity Function Algorithm for Character-Based Information Extraction

```
Input: Method calls in String format $(c_1, c_2)$
Output: Similarity Score

$Similarity\ (c_1, c_2)$ begin
if $(Length.c_1 < Length.c_2)$
then $Swap\ (c_1, c_2)$
    BigLength $\longleftarrow Length.c_1$
Return (bigLength – $ComputeEditDistance^*(c_1, c_2)$) / bigLength
```

** We have implemented the "Levenshtein distance" algorithm to compute the Edit distance*

Word-Based Extraction of a Web page

We used a word-tokenizer in NLTK [76] to split the text into words and store them in two arrays ($pageA$, $page$B). Also, we used three different methods to compare the similarity of web pages based on the words' extraction of each page:

1 Basic word's comparison
2 Doc2Vec
3 TF-IDF

As shown in Table 3.3, for simplicity, we name the Basic word comparison, Doc2Vec, and TD-IDF methods as Word-1, Word-2, and Word-3, respectively.

Table 3.3: Simple Terms for Text-Based Methods Used in This Study

WORD-1	Basic word's comparison
WORD-2	DOC2VEC
WORD-3	TD-IDF

1 Basic Word’s Comparison

In this method, we compare the words of the web pages with each other. It is a basic comparison of two pages without considering semantic relationships among words (See Algorithm 3.4 for the similarity function).

ALGORITHM 3.4: Similarity Function Algorithm for Word-1 Method

Input: Two arrays of words (s_1, s_2)

Output: Similarity Score

```
Similarity $(s_1, s_2)$ begin
counter = 0
for each word $w_1$ in $s_1$
   for each word $w_2$ in $s_2$
        if ($w_1 == w_2$)
         then counter = counter + 1
counter ⟵ normalize* (counter)
Return (counter)
```

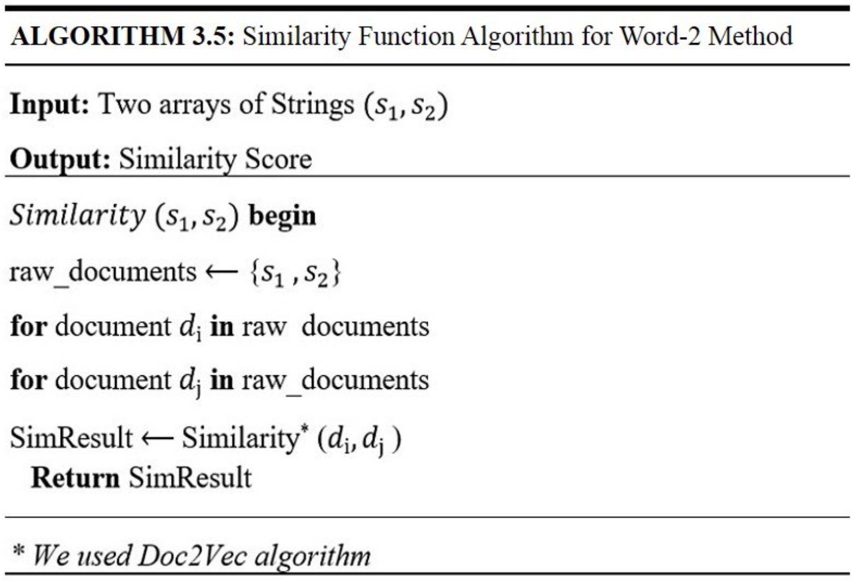
* *We normalized as: counter/(* $Length.s_1 \times Length.s_2$*)*

2 Doc2Vec Word Embedding

It is also known as word representation, plays an increasingly vital role in building continuous word vectors based on their context in a large corpus. Word embedding captures both semantic and syntactic information of words and can be used to measure word similarities, which are widely used in various NLP tasks [77].

The idea that words occurring in similar contexts have similar meanings [78] has provided the basis for many methods that use word co-occurrences to create vector representations of words (i.e. word embeddings), such that words with similar meanings have similar vectors [79, 80]. Le and Mikolov [81] proposed Doc2Vec as an extension to Word2Vec [82] to learn document-level embeddings. The goal of Doc2Vec is to create a numeric representation of a document, regardless of its length. After the model is trained, the word vectors are mapped into a vector space such that semantically similar words have similar vector representations (e.g., “strong” is close to “powerful”) [81].

So when training the word vectors W, the document vector D is trained as well, and at the end of the training, it holds a numeric representation of the document. While the word vectors represent the concept of a word, the document vector intends to represent the concept of a document.



ALGORITHM 3.5: Similarity Function Algorithm for Word-2 Method

Input: Two arrays of Strings (s_1, s_2)

Output: Similarity Score

```
Similarity $(s_1, s_2)$ begin
raw_documents ⟵ $\{s_1, s_2\}$
for document $d_i$ in raw  documents
for document $d_j$ in raw_documents
SimResult ⟵ Similarity* $(d_i, d_j)$
  Return SimResult
```

** We used Doc2Vec algorithm*

We used gensim's [83] implementation of Doc2Vec to determine the similarity of the web pages (See Algorithm 3.5).

3 TF-IDF

Another way to create a numeric representation of a word is by using TD-IDF (Term Frequency-Inverse Document Frequency) [84], which is designed to reflect the importance of a word to a document in a corpus.

Term Frequency is the number of times a word has occurred in the document. TF-IDF scheme is used for extracting features or important words which can be the best representative of the document. It lowers the weight of the words that occur too often in all the sentences such as 'a', 'the', 'as' and increases the weight of the words that can be important in a sentence.

TF-IDF is used to convert a document into a structured format. The TF-IDF value increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus, which helps to control for the fact that some words are generally more common than others [85].

We also used gensim's [83] implementation of TF-IDF to obtain the similarity of the web pages (See Algorithm 3.6 for the similarity function).

ALGORITHM 3.6: Similarity Function Algorithm for Word-3 Method

Input: Two arrays of Strings (s_1, s_2)

Output: Similarity Score

```
$Similarity\ (s_1, s_2)$ begin
raw_documents ⟵ $\{s_1 , s_2\}$
for word $w_i$ in raw  documents
for word $w_j$ in raw_documents
SimResult ⟵ Similarity* $(w_i, w_j)$
Return SimResult
```

** We used TF-IDF algorithm*

We applied several methods to compare the content of the web pages in our approach. The result (Table 3.4) shows that the models inferred using the Word-3 approach (TF-IDF) perform better when compared to the models implementing Word-1, Word-2, character-based and visual-based extraction of a web page, in terms of a measure of the strength of the association between two variables. Further details are provided in Section 3.5.

3.4.3.4 Automated Reward Calculation with an Example

We initialize the reward calculation process to give all the states the same reward value (zero) and the same chance of being observed (requested by users).

Using this procedure, all reward values are calculated incrementally during the model generation process. Eventually, the inference engine assigns the rewards of states as the sum of the reward values of the propositions associated with the states. Our proposed reward calculation process not only automatically incrementally computes the reward values, but also uses the server-side logs as the only source of the input. Moreover, the empirical evaluation (provided in the next section) shows that the reward values calculated by our proposed approach correctly represent the benefits or losses associated with the states.

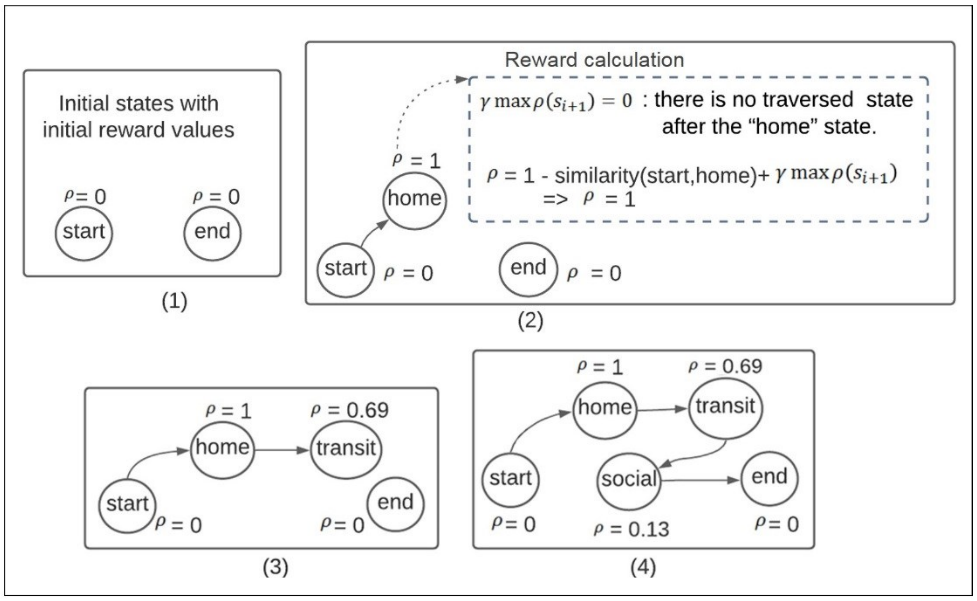


Figure 3.3: An Excerpt of the Reward Calculation Procedure for MyUAlberta Study Instance

Figure 3.3 shows the initial steps of the reward calculation process for the MyUAlberta study instance. As depicted, the model generation process initially goes through the same approach discussed in Section 3.4.2; and when a state is generated, the reward value is also calculated incrementally. Therefore, first, the content of the current state is extracted and compared with the new state's content using $similarity\,(start, home)$. Then, the maximum amount of previously initialized or calculated values for the next state is considered. Therefore, in this case, $\gamma \max \rho(s_{i+1}) = 0$, because there is no already traversed state after the "home" state. So, for instance, in a case that the user browses only one page, the model contains at least 3 states: start, browsed page, and end; and the reward value of the considered page would be 1. After browsing the "home" page, the user visits another page called "transit", so its state is generated in the model. Then, the similarity of these browsed pages is calculated using $similarity\ (home, transit)$ and since there is no traversed state after "transit", then $\gamma \max \rho(s_{i+1}) = 0$, and the reward value would be 0.69 which shows the similarity value of the pages. As shown in Figure 3.3 the user browses another page called "social" which is traversed from "transit". The reward is calculated by comparing the content of the current (social) and the last visited page (transit) and it would be 0.13. Because there is no traversed state after the "social" state, then $\gamma \max \rho(s_{i+1}) = 0$ and the reward value is calculated according to the similarity function ($similarity\ (transit,$ social)). Finally, there is no further state which means the user session has timed out; therefore, there would be a transition to the end state which has a reward value of 0.

3.4.4 Analyzing the Model

To analyze the behavioral model, it is necessary to identify one or more properties of the model, which can be evaluated by a probabilistic model checker (e.g. PRISM). In this step, the system expert defines properties of interest using a reward-augmented Probabilistic Computation Tree Logic (PCTL). This approach helps to identify the set of DTMCs, which are more relevant to the specified property [33, 86, 87].

In this study, we are more interested in properties that are specifying reward values of different states in the final model. Therefore, this approach analyzes properties, which are related to the expected values of the rewards. This is achieved using the $\mathcal{R}$ operator.

The $\mathcal{R}$ operator in a model checker can be used in a Boolean-valued query: $\boldsymbol{\mathcal{R}}\, bound\ [rewardprop]$ or a real-valued query: $\mathcal{R}\, query\ \left[\, rewardprop\, \right]$. Where bound takes the form $< r,\ <= r,\ > r$, or $>= r$ for an expression r and query is $=?,\ min\ =?$ or $max\ =?$.

rewardprop represents the reward property. There are various types of reward properties:

Reachability reward $\mathcal{F}$: Reward accumulated along a path until a certain point is reached;

Cumulative reward $C <= k$: Expected state reward cumulated after k steps;

Instantaneous reward $I = k$: Expected state reward to be gained in the state entered at step k;

Steady-state reward S.

For example, to consider the reward value of all the states up to the state labeled as “news”, the following property can be used:

$$\{\}\,\mathcal{R} =_{?} [\mathcal{F}\ news] \tag{3.15}$$

Inside the bracket {}, the web application development or operations team can also specify the scope of the property for a defined user class (e.g. a user agent) or simply leave it empty (no limit).

Given a property and a set of inferred DTMCs, the algorithm identifies DTMCs, which are relevant to the scope of the property. For instance, if the scope of the property is limited to users who browse the application with Chrome, the inference engine only selects the DTMC, which are associated with this specific user class. In situations where the algorithm selects more than one DTMC for the specified scope, the extracted DTMCs need to be merged to build a single DTMC.

To build a single DTMC by merging selected DTMCs (if there are multiple DTMCs) we used the merging approach suggested in [33] as follows. It is used in the property analysis step of this study.

- The set of states in the new DTMC consists of the union of the states of the DTMCs required to be merged.
- The transition probabilities in the new DTMC are calculated using the law of total probability:

$$P_T(s_i, s_j) = \sum_{1 \le k \le n} P_k(s_i, s_j) \times P_i(u_k) \tag{3.16}$$

  where, $P_i(u_k)$ is the probability of belonging to the user-class u_k when transitioning from s_i to s_j.

- Labels of the states in the new DTMC are the same as labels in their corresponding input DTMC.
- Reward values of the states in the new DTMC are the same as reward values in their corresponding input DTMC.

As it is previously mentioned, our approach evaluates the specified property for the final DTMC using PRISM. PRISM is not only able to evaluate the truth or falsity of a property but also can compute the reward functions using the reward properties. Additional details are provided in Section 3.5.7.

Therefore, our framework passes the property and the DTMC to PRISM and receives the results of the evaluation. In the following section, we empirically evaluate the performance of this approach in a real-life, enterprise-critical study instance.

3.5 Empirical Evaluation

3.5.1 Empirical Study

To evaluate the performance of the proposed approach, we apply it on two separate log files obtained from MyUAlberta, a large-scale mobile, and desktop application. Our approach generates user behavior models based on these log files.

3.5.2 Privacy

The data collection process has been discussed and approved by the University of Alberta, IST department. We undertook all possible actions to protect the leakages of private information and the identity of users. In particular, the IP addresses of users are anonymized, and only the data that is strictly needed for our study is retained.

3.5.3 Research Goal

The goal of this research is to generate a behavior model from users' navigation histories recorded in a log file to understand the users' interests in browsing pages of web applications. Augmenting behavior models with appropriate metrics (reward value) is required to address questions such as: (1) Which pages of an application may have design limitations? (2) How can a model identify and improve these limitations? And (3) Which are the most and the least interesting pages of the application? (less interested by users)?

3.5.4 Results

Regarding our proposed approach, the most interesting (popular) pages in terms of the amount of differentiation they provide, are labeled with the highest reward values. Therefore, it makes it possible to have a comparison with the results of users' behavioral-flow from Google Analytics. We use Google Analytics as the state-of-the-art approach for web analytics and tracking users' navigations.

We use log files of two months (January and March 2018) of the MyUAlberta application; each has 23 modules (states) as shown in the first column of Table 3.4 as "URL". The second to sixth columns of Table 3.4 (a) show the results of calculating reward values using the automated reward calculation algorithm, which is built based on the first log file (January). The second to sixth columns of Table 3.4 (b) show the results of the reward values based on the second log file (March). The rewards are calculated according to the character-based, visual-based, and three different ways of word-based extraction of a web page.

Table 3.4: Results of Running Reward Calculation Algorithm on MyUAlberta Case Study

URL	**(Jan) Character-Based**	**(Jan) Visual-Based**	**(Jan) Word-1**	**(Jan) Word-2**	**(Jan) Word-3**
	Reward Value	**Reward Value**	**Reward Value**	**Reward Value**	**Reward Value**
…/home/	0.8328	0.5886	0.8916	0.6322	0.8720
…/athletics/	0.1875	0.3423	0.3052	0.2376	0.2485
…/social/	0.2180	0.1962	0.3248	0.2354	0.2267
…/trnst/	0.5014	0.4098	0.4556	0.4142	0.4360
…/news/	0.1395	0.2180	0.2485	0.3575	0.2442
…/video/	0.1264	0.4033	0.2790	0.2180	0.2202
…/uaemergency/	0.1744	0.1984	0.1962	0.1962	0.2354
…/calendar/	0.1264	0.0218	0.2790	0.2420	0.2551
…/people/	0.1395	0.3423	0.2376	0.1591	0.2202
…/login/	0.4338	0.1962	0.4316	0.2420	0.3793
…/eclass.srv.ualberta.ca/portal/	0.4578	0.4098	0.4360	0.4360	0.4098
…/campusmap.ualberta.ca/	0.3161	0.2136	0.3030	0.2202	0.3314
…/myonecard.ualberta.ca/	0.2354	0.2245	0.3096	0.2158	0.2616

URL	**(Jan) Character-Based**	**(Jan) Visual-Based**	**(Jan) Word-1**	**(Jan) Word-2**	**(Jan) Word-3**
	Reward Value	**Reward Value**	**Reward Value**	**Reward Value**	**Reward Value**
…/capsconnections.ualberta.ca/	0.1461	0.2376	0.2376	0.1962	0.2180
…/stustrv/	0.3662	0.1962	0.4273	0.3379	0.3989
…/customize/	0.2049	0.2180	0.2376	0.1788	0.2180
…/MyUAlbertaFeedback/	0.0000	0.0000	0.0000	0.0000	0.0000
…/search/	0.1875	0.3946	0.3052	0.2245	0.2529
…/photos/	0.2049	0.4142	0.2376	0.1940	0.2333
…/ualberta.ca	0.2136	0.3662	0.2921	0.2289	0.2485
…/kurogoerror/	0.1657	0.2180	0.2878	0.0262	0.1635
…/library/	0.2354	0.4120	0.3357	0.2747	0.2790
…/registrar/	0.3880	0.1962	0.4251	0.3335	0.3706

(a)

URL	**(March) Character-Based**	**(March) Visual-Based**	**(March) Word-1**	**(March) Word- 2**	**(March) Word-3**
	Reward Value	**Reward Value**	**Reward Value**	**Reward Value**	**Reward Value**
…/home/	0.8393	0.5886	0.8916	0.6496	0.8764

URL	(March) Character-Based	(March) Visual-Based	(March) Word-1	(March) Word- 2	(March) Word-3
	Reward Value	Reward Value	Reward Value	Reward Value	Reward Value
…/athletics/	0.2267	0.3423	0.2267	0.2463	0.2333
…/social/	0.2180	0.1962	0.2398	0.2376	0.2071
…/trnst/	0.4360	0.4316	0.4404	0.4033	0.4142
…/news/	0.2267	0.2180	0.2289	0.3314	0.2180
…/video/	0.1482	0.4033	0.1570	0.2224	0.1918
…/uaemergency/	0.1482	0.1984	0.1526	0.2180	0.1918
…/calendar/	0.2289	0.0218	0.2289	0.2572	0.2202
…/people/	0.2420	0.3423	0.2594	0.2354	0.2442
…/login/	0.3880	0.1962	0.3750	0.2245	0.3270
…/eclass.srv.ualberta.ca/portal/	0.4316	0.4273	0.4316	0.3859	0.4120
…/campusmap.ualberta.ca/	0.2289	0.2136	0.2376	0.2311	0.2180
…/myonecard.ualberta.ca/	0.2572	0.2245	0.2616	0.2790	0.2463
…/capsconnections.ualberta.ca/	0.2071	0.2376	0.2115	0.2354	0.2027
…/stustrv/	0.3662	0.1962	0.3706	0.2594	0.2398
…/customize/	0.2115	0.2180	0.2158	0.2180	0.1962

URL	(March) Character-Based	(March) Visual-Based	(March) Word-1	(March) Word- 2	(March) Word-3
	Reward Value	Reward Value	Reward Value	Reward Value	Reward Value
…/MyUAlbertaFeedback/	0.0000	0.0000	0.0000	0.0000	0.0000
…/search/	0.2180	0.3946	0.2485	0.2507	0.2420
…/photos/	0.2202	0.4142	0.2180	0.2398	0.2180
…/ualberta.ca	0.0000	0.0000	0.0000	0.0000	0.0000
…/kurogoerror/	0.1657	0.2180	0.1134	0.2180	0.1766
…/library/	0.2354	0.4120	0.2485	0.2507	0.2289
…/registrar/	0.3880	0.1962	0.3771	0.3488	0.2834

(b)

Note: (a): Reward values using the automated reward calculation algorithm for the first log file. (b): Reward values using the automated reward calculation algorithm for the second log file.

These results illustrate that some pages (URLs) have higher reward values in each extraction category compared to others, which indicates that these pages provide more varied content. We implement several extractor functions in our approach. As shown in Table 3.4, the homepage of the application has the maximum recorded reward value; the "transit" and "eclass" modules are ranked second and third in terms of reward values. The reward value indicates a user's interest in visiting a web page; higher reward values represent popular pages.

The MyUAlberta application was launched for the first time in 2014 and it has been continuously attached to a Google Analytics account since then. Thus, historical data from actual interacting users is stored accurately. To indicate the interesting (popular) pages from users' perspective, Google Analytics' data along with users' behavioral flows is helpful. Google Analytics shows page views that represent the total number of pages that visitors looked at on the website.

Table 3.5 provides the number of page-views for each considered page (URL) in January and March. According to Table 3.5 (a), the maximum number of viewers belongs to the homepage which equals 35,385 page-views in January. The "transit" page has the second maximum number of viewers at 10,563 page-views comparing to other pages. Table 3.5 (b) shows that the homepage of the application has been viewed 35,267 times which had the maximum number of viewers amongst all other pages. Also, the "transit" page was the second more viewers page compare to other pages. It had been viewed 10,382 times in March.

A user flow is a series of web pages and includes the steps that a user performs to complete the task, during a website visit or other process. The Users' behavioral flow report extracted from the Google Analytics account illustrates the paths users take through an application's content. The web application developers can use the flow reports to see how users enter, engage, and exit the application. Developers can also use these reports to troubleshoot the application's content by finding any unexpected place users exit or loopback. Figure 3.4 shows the users' behavioral flows representing the most engaging content in the web application for the two mentioned months. Figure 3.4 (a) shows the behavioral flow in January; and, Figure 3.4 (b) illustrates the users' behavioral flow in March. Unlike a map, which displays possible and known routes, a flow visualization reveals the actual path as it was traveled step by step, including any detours or backtracking that happened along the way.

Table 3.5: Number of Page-Views for Each Considered Page (URL) on MyUAlberta Study Instance

Index	**URL**	**(a) January Page-views**	**(b) March Page-views**
1	…/home/	35385	35267
2	…/athletics/	419	339
3	…/social/	290	222
4	…/trnst/	10563	10382
5	…/news/	507	378
6	…/video/	241	66
7	…/uaemergency/	109	60
8	…/calendar/	640	546
9	…/people/	271	1369
10	…/login/	4881	4689
11	…/eclass.srv.ualberta.ca/portal/	7871	8171
12	…/campusmap.ualberta.ca/	2001	898
13	…/myonecard.ualberta.ca/	1356	1435
14	…/capsconnections.ualberta.ca/	123	137
15	…/stustrv/	1747	1878
16	…/customize/	140	181
17	…/MyUAlbertaFeedback/	15	0
18	…/search/	937	1264
19	…/photos/	267	333

Index	**URL**	**(a) January Page-views**	**(b) March Page-views**
20	…/ualberta.ca	1159	0
21	…/kurogoerror/	282	118
22	…/library/	1100	1166
23	…/registrar	4113	2324

The top path is the most common user flow. It is the series of pages seen by typical visitors. Figure 3.4 shows the homepage in the MyUAberta application is the point of entry or the starting page, where users mainly start browsing the application. Then, during the first engagement (interaction), “eclass” and “transit” pages receive the highest portion of the traffic going out from the homepage. Following the connections and nodes through different engagement levels demonstrates the overall traffic flow in the web application.

To choose the best extractor function in this study, we compare the reward values calculated using different extractor functions with the Google Analytics’ page views considering the correlation coefficient value between them.

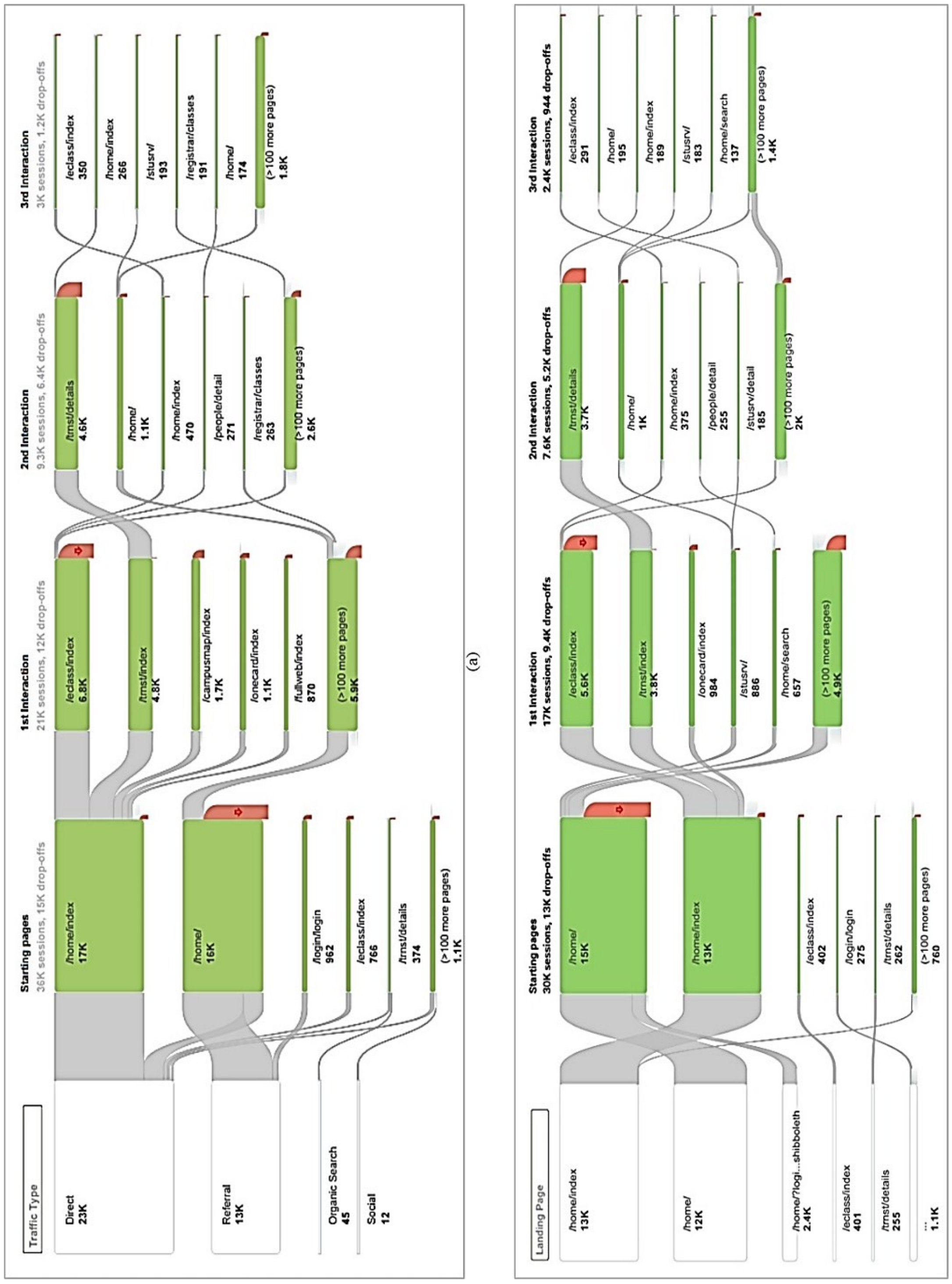


Figure 3.4: User Flow Extracted from Google Analytics, (a) in January; (b) in March

3.5.5 Correlation Coefficients

To determine which extractor function is performing better than the others, we use correlations between the reward values and the page view metric, extracted from the Google Analytics account. Since the variables are not normally distributed, we calculate the Spearman correlation between them. The correlation coefficient matrix of two random variables (X and Y) is the matrix of correlation coefficients for each pairwise variable combination:

$$R = \begin{pmatrix} r(X,X) & r(X,Y) \\ r(Y,X) & r(Y,Y) \end{pmatrix} \tag{3.17}$$

Where:

$$r(X,Y) =$$

$$(1/(N-1)) \sum_{i=1}^{N} ((X_i - \mu_X)/\sigma_X)\,((Y_i - \mu_Y)/\sigma_Y) \tag{3.18}$$

if each variable has N scalar observations. μ_X and μ_Y are the mean and σ_X and σ_Y are the standard deviations.

Table 3.6 shows the Spearman correlation coefficient matrix of different reward values of five extractor functions and page views for January and March; designated $\rho 1$ and $\rho 2$, respectively. $\rho 1$ and $\rho 2$ indicate the positive correlation between the two variables is statistically significant especially in the Word-3, Word-1, and character-based extractors which are 0.9345, 0.9088, and 0.8400 for January and 0.9040, 0.9010, and 0.8986 for March, respectively.

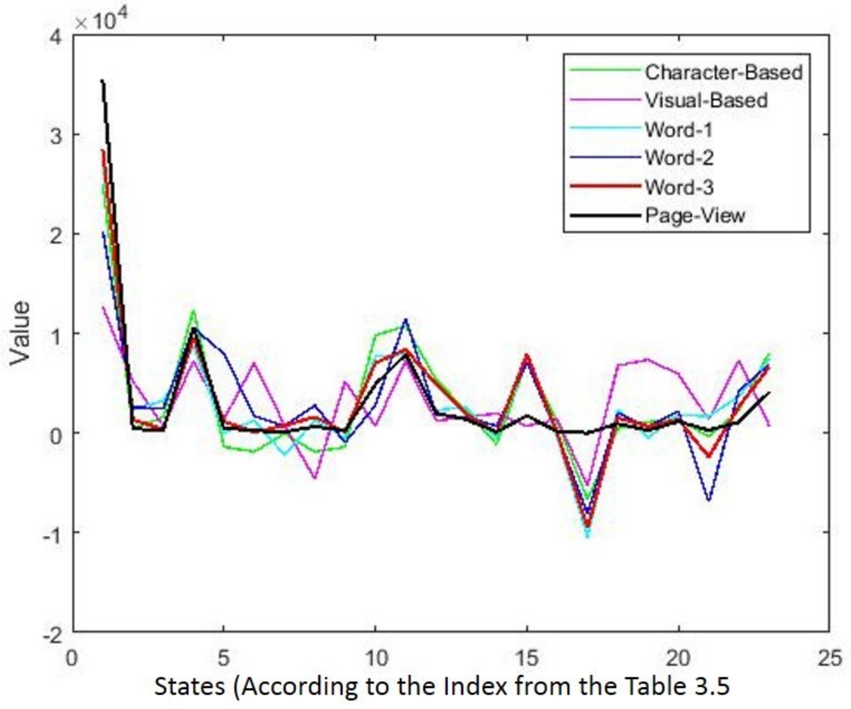
Table 3.6: Spearman Correlation Coefficient Matrix

Spearman Correlation Coefficient	$\boldsymbol{R1}$ $(\boldsymbol{January})$	$\boldsymbol{R2}$ $(\boldsymbol{March})$
Character-Based	$\begin{pmatrix} 1 & 0.8400 \\ 0.8400 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0.8986 \\ 0.8986 & 1 \end{pmatrix}$
Visual-Based	$\begin{pmatrix} 1 & 0.3273 \\ 0.3273 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0.3031 \\ 0.3031 & 1 \end{pmatrix}$
Word-1	$\begin{pmatrix} 1 & 0.9088 \\ 0.9088 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0.9010 \\ 0.9010 & 1 \end{pmatrix}$
Word- 2	$\begin{pmatrix} 1 & 0.8003 \\ 0.8003 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0.7256 \\ 0.7256 & 1 \end{pmatrix}$
Word-3	$\begin{pmatrix} 1 & 0.9345 \\ 0.9345 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0.9040 \\ 0.9040 & 1 \end{pmatrix}$

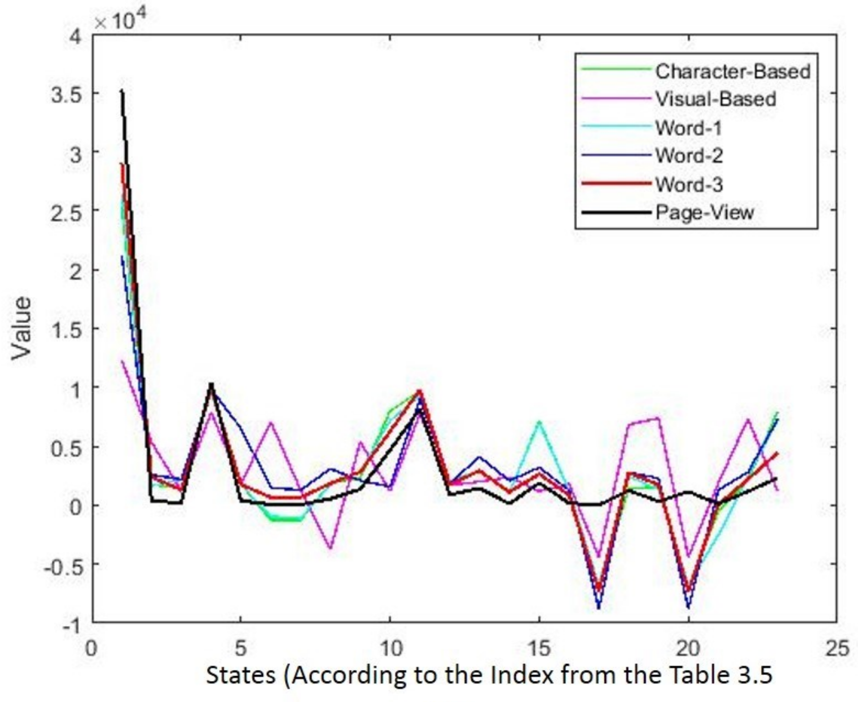
Figure 3.5 (a) depicts the Spearman correlations between Google Analytics' page views and character-based, visual-based, Word-1, Word-2, and Word-3 reward extractors in MyUalberta study instance during January. The order of the states (pages) represented in this figure is based on the index from Table 3.5. The peak in all plots belongs to the homepage, which has the highest reward value and page views. The "strong" correlation between the variables (reward values and Google Analytics' page views) is represented in the word-3 reward extractor shown with red color. As it is shown in this curve (word-3), there is a pretty strong positive

correlation as the slopes and peaks in the fitting polynomials can be visually compared.

Figure 3.5 (b) shows the Spearman correlations between Google Analytics' page views and character-based, visual-based, Word-1, Word-2, and Word-3 reward extractors in MyUalberta study instance during March. Again, the peak in all plots belongs to the homepage, which has the highest reward value and page views. It is illustrated that there is a strong correlation between the Word-3 extractor reward values and Google Analytics' page views.



(a)



(b)

Figure 3.5: Spearman Correlations among Google Analytics' Page Views and Different Reward's Extractors (a) for the First Log File (January) (b) for the Second Log File (March)

We also calculated the Pearson correlation coefficient to indicate the correlations between the reward values calculated using our approach, and the page view metric, extracted from the Google Analytics account.

Table 3.7 shows the Pearson correlation coefficient matrix of reward values of the extractor functions and page views for two months of January and March determined with R1 and R2, respectively.

R1 and R2 indicate the positive correlation between the two variables is statistically significant especially in Word-3, Word-1, and character-based methods which are 0.9115, 0.8850, and 0. 8606 for January and 0.9162, 0.8804, and 0.8672; for March, respectively. However, the correlation coefficient between the rewards based on Word-3 and the Google Analytics' page-views has the best result amongst the other extractors; it is illustrated that there is a strong correlation between the Word-3 extractor reward values and Google Analytics' page views. This result again shows the importance of calculating reward values in user behavioral models especially in the cases that Google Analytics data does not exist or is not available.

Table 3.7: Pearson Correlation Coefficient Matrix

Pearson Correlation Coefficient	$\boldsymbol{R1}\ (\boldsymbol{January})$	$\boldsymbol{R2}\ (\boldsymbol{March})$
Character-Based	$\begin{pmatrix} 1 & 0.8606 \\ 0.8606 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0.8672 \\ 0.8672 & 1 \end{pmatrix}$
Visual-Based	$\begin{pmatrix} 1 & 0.5543 \\ 0.5543 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0.5716 \\ 0.5716 & 1 \end{pmatrix}$
Word-1	$\begin{pmatrix} 1 & 0.8850 \\ 0.8850 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0.8804 \\ 0.8804 & 1 \end{pmatrix}$
Word- 2	$\begin{pmatrix} 1 & 0.7801 \\ 0.7801 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0.7751 \\ 0.7751 & 1 \end{pmatrix}$
Word-3	$\begin{pmatrix} 1 & 0.9115 \\ 0.9115 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0.9162 \\ 0.9162 & 1 \end{pmatrix}$

As shown in Figure 3.5, Table 3.6, and Table 3.7 there is a strong correlation between the Word-3 reward values and Google Analytics' page views. Therefore, we use the Word-3 extractor for calculating the reward values in our approach. All future discussion in this study is based on the Word-3 extractor. For clarification, Table 3.8 (a) and (b) show the reward values using the Word-3 extractor for two log files.

Table 3.8: Reward Values using Word-3 Extractor for Two Log Files

URL	**(a) January- Reward Value**	**(b) March- Reward Value**	**(c) January- (Error-rate)**	**(d) March- (Error-rate)**
…/home/	0.8720	0.8764	0.0491	0.0258
…/athletics/	0.2485	0.2333	0.9887	0.9906
…/social/	0.2267	0.2071	0.9922	0.9939
…/trnst/	0.4360	0.4142	0.7162	0.7132
…/news/	0.2442	0.2180	0.9864	0.9896
…/video/	0.2202	0.1918	0.9935	0.9982
…/uaemergency/	0.2354	0.1918	0.9971	0.9983
…/calendar/	0.2551	0.2202	0.9828	0.9849
…/people/	0.2202	0.2442	0.9927	0.9622
…/login/	0.3793	0.3270	0.8688	0.8705
…/eclass.srv.ualberta.ca/portal/	0.4098	0.4120	0.7885	0.7743

URL	(a) January- Reward Value	(b) March- Reward Value	(c) January- (Error-rate)	(d) March- (Error-rate)
…/campusmap.ualberta.ca/	0.3314	0.2180	0.9462	0.9752
…/myonecard.ualberta.ca/	0.2616	0.2463	0.9636	0.9604
…/capsconnections.ualberta.ca/	0.2180	0.2027	0.9967	0.9962
…/stustrv/	0.3989	0.2398	0.9531	0.9481
…/customize/	0.2180	0.1962	0.9962	0.9950
…/MyUAlbertaFeedback/	0.0000	0.0000	0.9996	0.9996
…/search/	0.2529	0.2420	0.9748	0.9651
…/photos/	0.2333	0.2180	0.9928	0.9908
…/ualberta.ca	0.2485	0.0000	0.9689	0.9680
…/kurogoerror/	0.1635	0.1766	0.9924	0.9967
…/library/	0.2790	0.2289	0.9704	0.9678
…/registrar/	0.3706	0.2834	0.8895	0.9358

Regarding Table 3.8 (a) and (b), the homepage of the application has the maximum amount of reward values compared to all the other pages which are equal to 0.8720 and 0.8764 for the first and the second log files, respectively. It means that users are mostly going to the homepage of the application. Therefore, it is the most popular page and users are more interested to visit the page. The second and the third ranks of maximum reward values include “transit” and “eclass” modules which are equal to 0.4360, 0.4142, and 0.4098, 0.4120 for the first and the second log file, respectively.

As it is shown with red color (word-3) in Figure 3.5 (a) and (b), there is a “strong” correlation between the rewards of our approach and the page-views of Google Analytics. It indicates that there is an association between pages with higher page views and pages with higher reward values. Pages with higher page views have higher corresponding reward values. It shows that users are more interested to visit these pages.

Pages with higher reward values are pages with more varied content compared to the previously viewed page. The results also indicate that such high reward values’ pages are offering different links and text rather than the previously viewed page. So, the users are more interested and, subsequently, apt to explore the pages containing varied contents.

According to the results, the homepage has the maximum reward value. Users may be brought to the homepage indirectly, for instance, a user can be in /library/ page and go to the homepage by clicking on “UAlberta Home”. However, many users directly visit the homepage without referrals from other pages. Therefore, in many cases, there is no actual URL exists to be compared to the homepage. As a result, the homepage receives more accumulated rewards (Equation 3.14). This can easily explain the reason for the high reward value of the home state. Our proposed approach not only automates the reward calculation process but also produces results, which are technically explainable and compatible with the data extracted from Google Analytics (as the proof of the concept). The difference between Google Analytics’ page views and reward values as “Error-rate” is normalized and shown (per page) in Table 3.8 (c) and (d), for two log files. Regarding this table, the homepage of the application has the minimum amount of error rate compared to all the other pages which is equal to 0.0491 and 0.0258 for the first and the second log files, respectively. The second and the third ranks of minimum error rate include “transit” and “eclass” modules which are equal to 0.7162, 0.7132, and 0.7885, 0.7743 for the first and the second log file, respectively.

The results (Table 3.8 (a) and (b)) also indicate that such high reward values’ pages are offering different links and text rather than the previously viewed page. So, the users are more interested and, subsequently, apt to explore the pages containing varied contents.

As it is shown in Table 3.8, the reward value of the “/ualberta.ca” page is 0.2485 and 0, respectively for the first and the second log file. It indicates that during March it was not possible to reach the main page of the University of Alberta web application from other pages of the application. It was found that there was a design anomaly in the UAlberta application and users were not able to go to the homepage of the application by clicking on the “UAlberta Home” during March. (For instance, a user can be on the “eclass” page and go to the homepage by clicking on “UAlberta.ca”.) Also according to Table 3.5, we can see that the page view of the “ualberta.ca” page is 1159 and 0 for January and March, respectively. It indicates that users could not be brought to the homepage from the other pages of the application during March. This also shows that our proposed approach for calculating the reward values is compatible with the data extracted from Google Analytics.

It is possible to analyze users’ behaviors of a web application from its inferred model. By analyzing the behavior of the inferred model from the MyUalberta application, it is represented there is not any transition from the “registrar” module to the other pages and see courses users have registered. It indicates that there could be an anomaly in the “registrar” module where there are not any transitions that come out from this state. It was found that a dead-end occurred in the “registrar” module of the application. Users are being prevented from leaving the “registrar” page without closing it. Therefore, there is no transition from this module to others. This shows that our proposed approach can detect design anomalies and dead-ends.

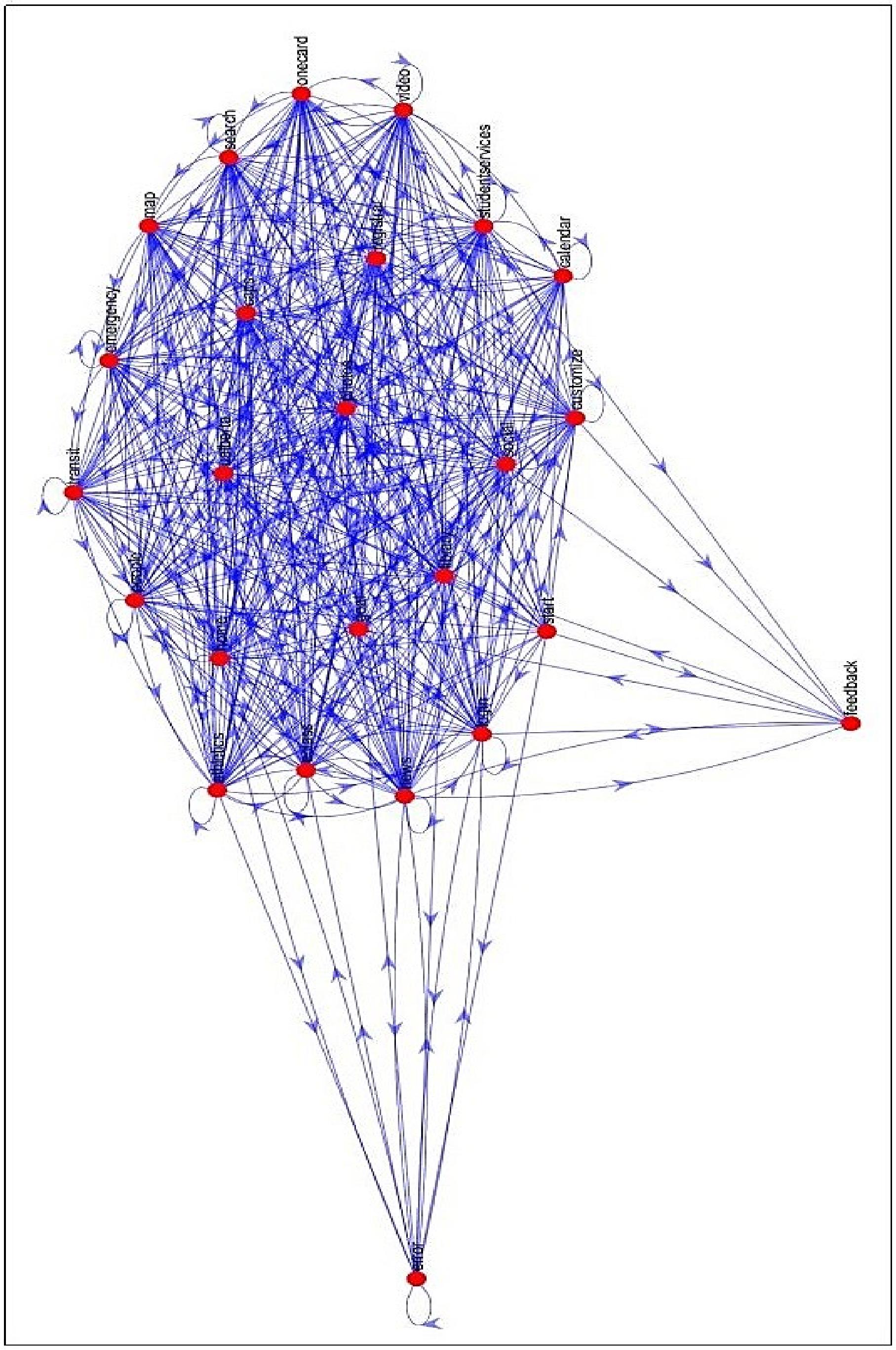


Figure 3.6: The Entire User Behavior Model of “MyUAlberta” Study Instance with 23 States (Also Two States as Start and End) and 486 Transitions

3.5.6 Real-Time Simulation

To demonstrate the flexibility of our approach, we mimic a new dataset. We merge these two log files and create a single log file containing the information of both log files. The entire model is shown in Figure 3.6 which has 486 transitions, 23 states, and two states for starting and ending the model. In the following paragraphs, we empirically evaluate the performance of our approach on the merged log file. The source code was implemented on Intel(R) Core(TM) i7-3770 CPU, 16.0 GB RAM.

The Spearman correlation coefficient matrix of reward values and page views for the merged log file is presented as follow:

Spearman Correlation Coefficient = $\begin{pmatrix} 1 & 0.8861 \\ 0.8861 & 1 \end{pmatrix}$

It states that there is a “strong” correlation between the reward values and Google Analytics’ page views. Again, these results are consistent with the previous analysis.

As it is aforementioned in this work, to eliminate any redundancies in the model, the merging step of the gkTail inference algorithm is applied to merge the equivalent states. According to this state-merging procedure, two states are considered equivalent if they have the same future of length k (in our study k=1). This procedure prunes the model of redundant states with the same values. The result shows that 971 merges are found during these experiments. Hence, the approach is certainly impacting model production and is helping to eliminate redundancies.

3.5.7 Model Analysis Using Probabilistic Model Checking

Probabilistic model checking refers to a range of techniques for calculating the likelihood of occurrence of certain events during the execution of systems that exhibit such behavior. PRISM, a probabilistic model checker, first parses a model description and constructs an internal representation of the probabilistic model, computing the reachable state space of the model and discarding any unreachable states. This represents the set of all feasible configurations which can arise in the modeled system. Next, the specification is parsed and appropriate model checking algorithms are performed on the model by induction over syntax. In some cases, such as for properties that include a probability bound, PRISM will simply report a true/false outcome, indicating whether or not each property is satisfied by the current model. More often, however, properties return quantitative results and PRISM reports, for example, the actual probability of a certain event occurring in the model.

One of the most important operators in the PRISM property specification language is the P operator, which is used to reason about the probability of an event's occurrence. PRISM also includes support for the specification and analysis of properties based on reward structures. It can analyze properties that relate to the expected values of these rewards. This is achieved using the R operator, which works similarly to the P operator.

Our proposed approach infers behavior models of a system and incrementally calculates reward values for each state of the model. By using a model checker (e.g. PRISM), questions such as "What is the probability of reaching the "eclass" web page in the next state?", "What is the probability of eventually visiting the "full web" state not followed immediately by "goal" state?", "Is there a path between the "registrar" and "social" web pages?", or "What is the probability of reaching the "transit" web page in the next second state?" can be answered. Also, model checking not only deals with correctness but also with incorrectness, often providing a counterexample in case the program does not meet the specification.

To analyze the behavioral model that has been inferred, we use PRISM to evaluate the inferred model's efficiency using the merged log file. Also, we express some queries which can be evaluated by PRISM as follows:

- To consider the reward values of all the states up to the state labeled as "goal" (end), the following property can be used: $\{\,\}\,\mathcal{R} =_{?} [\mathcal{F}\ goal]$. It calculates the reward values of all the states. The evaluation results are the same as the reward values calculated using our approach.

- According to the reward values and Google Analytics' page views of the application, it is illustrated that "home", "eclass" and "transit" pages of the application have the maximum amount of reward values. Therefore, users mostly prefer to visit these pages. By using PRISM, we calculate the probability of going to these pages in the next state using $\{\}\, P =_? [X\ home]$, $\{\}\, P =_? [X\ eclass]$ and $\{\}\, P =_? [X\ transit]$ queries. So, the probability values are 0.5434, 0.2168, and 0.2142 for "home", "eclass" and "transit", respectively which indicate a high probability of visiting these pages.
- We used a query to show the probability of eventually visiting the "registrar" state followed immediately by "goal" (end) which is $\{\}\, P =_? [F(\ registrar\ \&\ (X\ goal\ ))]$. It shows the probability of not visiting the other states from the "registrar" which is 0.1493.
- As it is mentioned in Section 3.5.5, users could not be brought to the homepage from the other pages of the application in March. $\{\}\, P =_? [F(\ fullweb\ \&\ (X\ !\, goal\ ))]$ shows the probability of eventually visiting the "full web" state not followed immediately by "goal" which is 0.1896.
- To realize the probability of reaching "home" in the next state and "news" in the next second state, we can use $\{\}\, P =_? [\ (X\ home\ )\&\ (XX\ news\ )]$ which results in 0.042.

So, our approach can evaluate specified properties for the final DTMC using probabilistic model checkers such as PRISM. As previously mentioned, PRISM can compute the reward functions using considered reward properties. We used some queries to check the model and the evaluation results indicate its compatibility with the inferred model. Also, we used PRISM to realize the probability of reaching from a specific starting state to a particular finishing state. Although we used PRISM, other probabilistic model checkers such as Storm [88] could be used instead.

3.5.8 Threats to Validity

In empirical studies like ours that involve model inference analysis, a set of threats exists that can raise questions about the validity of the research outcome. In this section, we present a set of such threats to the validity of our research that were addressed during this work. The first of such possible threats can be content validity. In the scope of this research, content validity refers to a subjective assessment that checks whether all reasonable evaluation metrics were considered during our study or not. To mitigate this threat, we composed them from an extensive review of the related literature.

To maintain the internal validity of the experiments, we not only formally defined the evaluation metrics, but we also carefully controlled the experimental environment and simulate them to fulfill all their desired assumptions. Furthermore, our method is a non-instrumented-based approach that minimizes the threats to internal validity. Also to evaluate the accuracy of our results, we not only compare them against the state-of-the-art approach but also an analyst from the University of Alberta, IST department, verified the correctness of our methods; and thus we believe in the adequacy of our accuracy measures.

Also, we plan to mitigate the threats to the external validity of the experiments that is to safeguard that our experimental results could be generalized outside the scope of this study. However, in this study, we have chosen a web application with multi-user processes.

3.6 Implications for Practice

We propose an approach to fully automate the behavioral model generation for web applications. The main learning points of this approach are explained in the following paragraphs.

At the first step, a set of Atomic Propositions (APs) is used to associate semantics to the URLs occurring in the data entry. APs can be defined by the system expert or automatically by considering the URL of the web page as a proposition. Our proposed inference framework uses several code fragments to indicate the set of atomic propositions for the MyUAlberta application, which can be associated with the relevant requested URLs in the log files.

Also, a system expert can define a set of user-classes to characterize different groups of users. Our inference framework contains two default classifiers to classify users based upon the user-browsers (e.g. Firefox) and the users' location extracted by geolocating the IP addresses. However to automatically infer a reward-augmented model, which is not limited to a specific scope, defining user-classes can be ignored.

As mentioned in Section 3.3, the reward calculation algorithm has been synchronized with the model-inference process. Accordingly, the rewards are calculated and updated incrementally during the model generation process. When the models are generated and annotated with reward values, the analysis engine evaluates the properties of the interaction patterns against the inferred models using probabilistic model checking (PRISM).

To construct probabilistic models (such as DTMCs) with PRISM, it must be specified in the PRISM language, a state-based language, based on the Reactive Modules formalism. To analyze these probabilistic models, it is necessary to identify one or more properties of the models; a property is the formal specification of a requirement to be checked. Properties of these models are written in the PRISM property specification language which is expressed in one of the probabilistic logics, a probabilistic extension of temporal logics. For instance, Probabilistic Computation Tree Logic (PCTL) is used for specifying properties of discrete-time models such as DTMCs.

Our approach integrates PRISM with our code, therefore, the model can be generated and analyzed in one single framework. However, the model and the properties can be separately written in the PRISM tool. The analysis engine of our approach processes PCTL properties relying on the PRISM probabilistic model checker; we present the PTCL properties in the PRISM syntax. The model checker then automatically checks if the model satisfies the given specification.

More precisely, a property is composed of a statement in curly brackets and a PCTL expression. The scope of the property is defined by the statement that can be ignored (void), while the PCTL expression specifies the formula to be verified with the model checker. The scope of the property identifies the user-classes the PCTL formula refers to.

As an example, as shown in Table 3.4, the reward value of "UAlberta.ca" is 0 in March. This issue can be identified by analyzing the model using the following property:

$$\{\,\}\, P =_{?} [F(\text{ UAlberta.ca} \,\&\, (X\, !\, goal\,))]$$

This property shows the probability of eventually visiting the "UAlberta.ca" state (UAlberta Home) not followed immediately by "goal" is 0. This means that users were not able to go to the homepage of the application by clicking on the "UAlberta Home" during March that is compatible with the page views from Google Analytics.

Although probabilistic model checking tools have been used to verify various systems, this usually has been done by software engineers who have a good understanding of DTMCs and are familiar with the syntax of both modeling and property specification languages. Indeed, the engineers can set properties in the analysis engine of our framework to describe the expected behaviors of either all or specific classes of users, as well as the impact of changes in the navigation attitude of users. They can set different properties to analyze the model. For example, they can use properties to check whether or not the model has dead-end or any other problematic pages and therefore, improve the application's design. Hence, a limitation of this modeling framework is that software engineers utilizing this system need to have a strong background in formal methods.

3.7 Conclusions and Future Research

In this research, we present a novel stochastic approach to (1) generate user behavioral models for mobile and desktop web applications, (2) automatically calculate the states' rewards, (3) annotate and analyze the models to verify the quantitative properties, and (4) address some limitations found in existing approaches. Our proposed approach not only builds a fully automated inference framework but also provides the following advantages as compared to other behavioral model generation methods:

- Our proposed approach is not only applicable to any new web application of any size and scale, but also legacy applications since it is not dependent on the specific input data. In other words, a server log file would be sufficient to start the modeling procedure.
- This approach provides the capability to evaluate and verify the property of the inferred models.
- Reward values can easily add semantics to inferred behavioral models. They help in interpreting model behaviors and detecting anomalies. Calculating reward values and assigning them to the states of the model during the inference procedure would be a more accurate and time-saving approach as compared to manually assigning them by systems experts. The proposed technique uses RL to incrementally calculate the value of the reward measure using the information extracted from browsed web pages in different states. The proposed approach adds meaningful reward values to the model, explaining the real users' interest or willingness in browsing web pages.
- It is easy to apply this procedure to calculate domain-specific reward values and identify different measures.
- It makes the dead-end or anomaly detection procedure faster and more meaningful by limiting the search space to the states with low reward values.

Besides, the empirical results approved that the proposed inference approach applies to large-scale mobile and desktop applications with many web pages and large volumes of interactions (data entries) per day, and can generate meaningful and compatible reward values in the considered study instance. We are extending the approach to be applicable on any probabilistic timed automata to calculate the reward values.

Chapter 4

Two New Semantic Approaches (Fusion-Block and Integrated-Block methods) to Improve Web page Segmentation

The World Wide Web has become a massive repository of information. The content and layout of web pages are getting more complex. Thus, identifying, and categorizing distinct informational elements from web pages has become increasingly difficult. Web page segmentation provides a solution to this problem. Web page segmentation is the process of partitioning a web page into blocks (visually and semantically coherent segments of a web page), in a manner, where each block contains distinctive content. Also, humans tend to segment a web page based on their understanding, thus, it is important to generate a segmentation model to segment a page by simulating human perception. Segmenting a web page into meaningful components (blocks) is one of the substantial pre-processing steps of web page analysis and contributes to several applications' domains such as information retrieval, web archiving, and mobile applications [6]. For instance, in the context of information retrieval, segmentation can be used to extract specific parts of a web page.

There are two major factors in segmenting a web page into different blocks, (1) how the content of a web page is extracted and (2) how the extracted content is processed to retrieve distinct information. Most research extracts and organizes content relying on the DOM (Document Object Model) structure of an HTML page using heuristic or machine learning-based approaches [7-10]. The DOM structure considers a web page document as a tree structure, where each fragment of the document is related to a particular node of the tree. Therefore, DOM represents the structure of web page design. However, the information available in the DOM is very limited. Additionally, full-text content is not extracted from the complex DOM structure which results in lacking the impressive feature extraction process [18]. To overcome these difficulties, some researchers prefer to segment web pages using visual information in a web page.

This vision-based segmentation method focuses on the analysis of visual features of the document content as they are perceived by a human reader. It exploits visual clues such as font size, font color, background color, spaces between paragraphs, etc. [19]. Considering the visual information allows these techniques to achieve a higher segmentation accuracy in comparison to DOM-based approaches [6], web pages are structured more flexibly now.

Some research presents segmentation methods based on both DOM-based and text-based approaches [89-91]. They focus on properties of the text content, such as density (number of characters in a text), to detect the content segments. However, current methods do not consider semantic analysis to categorize pages. Semantic analysis includes extracting text from segmented blocks, computing textual similarity, and regrouping blocks. A fusion approach that combines different analyses (DOM, vision, and text-based segmentation) is required to obtain higher segmentation accuracy.

Human tends to classify ambiguous objects based on their understanding. So, they group visually similar elements in a category. As an example, assume a page of a newspaper. Our mind automatically categorizes this page into separate groups without reading the text according to different features of the page such as the size of each column, font size, aligned lines, images, etc. This idea is proposed by Koffka et al. [25] and is known as Gestalt laws. According to Gestalt laws, humans group visually similar objects together based on several rules known as Gestalt laws of grouping [23-25, 92].

We propose two new combination models of web page segmentation by dividing the content of a web page into blocks by initially considering human perception (inspired by Gestalt laws of grouping) and subsequentially re-segmenting initial similar blocks using semantic text similarity. This research contributes to current research in web page analysis in the following ways:

- To improve the segmentation accuracy, this study provides two new semantic methods of web page segmentation by merging the DOM structure, vision-based similarity features, and text-based similarity metrics of web pages.
- Specifically, can a low-level visual-based segmentation be augmented with a high-level segmentation process that provides a semantic analysis of textual features?
- Further, we demonstrate the utility of transformer technology as a vehicle for this text-based process.
- By evaluating the system on three datasets and by comparing it with state-of-the-art studies, the results represent that our proposed approaches (Fusion-Block and Integrated-Block) outperform the other existing web page segmentation methods, in terms of higher accuracy.

The chapter is organized as follows: In Section 4.1 we explain the challenges and our research motivations; while Section 4.2 reviews and presents the related work. A detailed description of our approach is provided in Section 4.3, whereas Section 4.4 presents an evaluation of the proposed approach. In Section 4.5 we discuss some limitations and future work opportunities for our study and conclude the chapter.

4.1 Problem Statement and Research Motivation

It is quite easy for humans to recognize related web page content fast and correctly from complex pages. However, with the huge number of web pages, it is impossible to identify and segment related information manually. Web page segmentation models generated from different page features provide solutions to several web page analysis problems. To segment web pages based on human perception, it is essential to employ laws that simulate human understanding. Also, it is required to utilize semantic text similarity to segment web pages. This study segments web pages by simulating human perception and utilizing structure, vision, and text-based methods. Web page segmentation is often carried out for various purposes such as to retrieve or cluster information [6]. This research seeks to improve on these previous attempts in several ways which are described below.

The DOM structure of a web page is used in most of the existing research on web page segmentation. This segmentation model can only gain limited information from web pages. To improve the segmentation method, some research has been carried out using visual features of web pages. To simulate human perception in the segmentation process, a series of laws are presented, the Gestalt laws of grouping. According to these laws, humans group visually similar objects together. However, Gestalt laws do not consider the text similarity of web pages. To consider the text similarity, we use semantic text similarity metrics in addition to these laws to segment web pages into blocks. Thus, each block has related content in terms of both visual and textual features. In this research, semantic segmentation includes dividing the content of a web page into blocks by initially considering human perception (inspired by the Gestalt laws of grouping), and subsequentially re-segmenting these initial similar blocks using semantic text similarity. Further details of Gestalt laws and text similarity metrics are provided in Section 2 and Section 4.3, respectively.

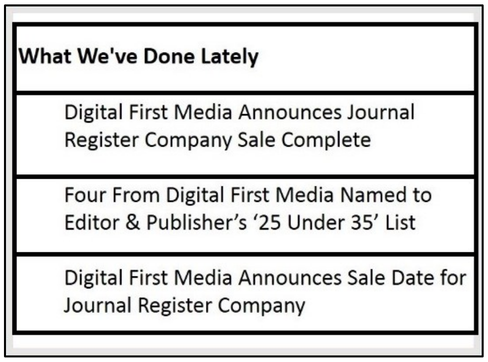
The shortcomings of the DOM structure lead to performance limitations of the structural-based segmentation methods; typically, the (long) text of a web page results in several short or scatter text segments [18]. However, these scattered text sections have related content; and hence, need to be grouped into a single block. It is hypothesized that to regroup small blocks and obtain longer text segments in larger merged blocks, a semantic analysis that uses Natural Language Processing (NLP) techniques is required.

Some segmentation methods have been carried out using NLP techniques [89-91]. These methods consider text density metrics such as text formats and words' frequency of a document but do not consider the semantic text similarity of blocks.

Two sentences can have the same meaning regardless of the choice of word and hence should be grouped in a single block; for example, consider the following two sentences that are grouped in two different blocks using a segmentation method.

“I read more books than Sarah”, and “She reads fewer books than me”.

Although these sentences have different words they have the same meaning. The segmentation method that segments these two blocks does not consider the semantic similarity of the sentences. Most of the segmentation methods focus on the structural and visual features of a web page. The semantic similarity of documents determines the probability of the relatedness of the documents [93]. Generally, documents are semantically similar if they convey the same meaning.


What We've Done Lately

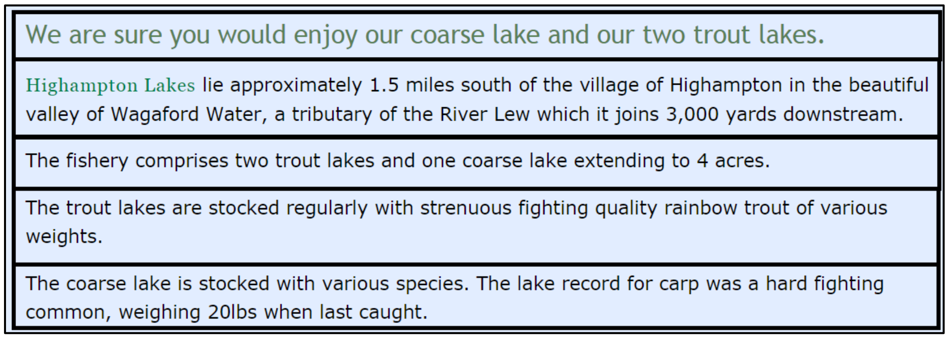
Digital First Media Announces Journal Register Company Sale Complete

Four From Digital First Media Named to Editor & Publisher’s ‘25 Under 35’ List

Digital First Media Announces Sale Date for Journal Register Company

Figure 4.1: A Part of the Page “www.journalregister.com”

As another example, consider some parts of the web pages “www.journalregister.com” and “www.fishdevon.co.uk” as shown in Figure 4.1 and Figure 4.2, respectively. The existing segmentation methods that we used in this research to compare with our methods, segment the paragraphs shown in Figure 4.1 and Figure 4.2 into four and five separated blocks, respectively, while these blocks are semantically related and need to be grouped in a single block. These methods use DOM structure and visual properties of web pages to segment pages. Most of the segmentation methods do not use semantic text similarity of blocks. Thus, some paragraphs with similar subjects are segmented into different blocks based on their text formats, instead of employing the semantic text similarity of blocks.


We are sure you would enjoy our coarse lake and our two trout lakes.

Highampton Lakes lie approximately 1.5 miles south of the village of Highampton in the beautiful valley of Wagaford Water, a tributary of the River Lew which it joins 3,000 yards downstream.

The fishery comprises two trout lakes and one coarse lake extending to 4 acres.

The trout lakes are stocked regularly with strenuous fighting quality rainbow trout of various weights.

The coarse lake is stocked with various species. The lake record for carp was a hard fighting common, weighing 20lbs when last caught.

Figure 4.2: A Part of the Page "www.fishdevon.co.uk"

Some research use textual features of web pages, for instance, Jiang et.al [18] propose a segmentation method that uses logical and visual features of content. Also, this method uses text density (the number of words in a block of text) to segment web pages. As mentioned in [10], this method does not consider semantic text similarity metrics of blocks. In this research, we propose two segmentation methods (Fusion-Block and Integrated-Block) that use structural, visual, and textual features of web pages. The first method uses the Doc2Vec technique and the second method uses the transformer to compare the text similarity of web pages.

Doc2Vec is an NLP technique for representing documents as a vector and is a generalization of the Word2Vec method. The Word2Vec technique assigns a single word embedding vector to each word in a text corpus based on the frequency of words. For example, the word "goal" in "last-minute goal" and "life goal" has the same word embedding representation vector. However, Doc2Vec does not use the relation of the word to the other words in a document by considering the meaning of sentences. Thus, we propose the Integrated-Block method to deeply compare the text similarity of web pages.

Google has proposed the Bidirectional Encoder Representations from Transformers (BERT) method for NLP techniques such as text classification, summarization, generation, and similarity [94]. BERT is a neural network-based technique to represent text. It uses the transformer encoder to learn a language model [95]. The transformer uses a sequence of tokens as an input and returns a sequence of vectors as an output, where each vector corresponds to an input token. A token is an instance of a sequence of characters in a document that is grouped as a semantic unit for processing [96]. First, the transformer randomly masks some of the input tokens to train a deep bidirectional representation and then predicts the masked tokens [94]. For example, consider this sentence "Sarah reads more books than me". BERT represents "reads" using both its right and left words by masking the word in the input as "Sarah xxx more books than me". Then, it runs the entire sequence through a bidirectional transformer encoder and predicts the masked word. More information can be found in [94]. (This research investigates the application of BERT in the Integrated-Block method as a mechanism to improve semantic web-based segmentation). To overcome the shortcomings of the structural-based and the NLP-based segmentation methods, this research uses BERT to deeply compute the semantic textual similarity between the basic-blocks and regroup the related blocks in merged blocks. This similarity regrouping model leads to more stable semantic features. Further details are provided in section 4.4; Figure 4.11 (d), (e) and Figure 4.12 (d), (e) show the results of web page segmentation using our methods on some parts of the web pages represented in Figure 4.1 and Figure 4.2.

To achieve a high-performance segmentation model, it is required to merge the visual and textual content of a web page into a single model. By using the latest semantic NLP techniques, such as BERT, we can produce a superior model. We demonstrate the validity of this conjecture by an empirical comparison against the current state-of-the-art techniques.

4.2 Related Work

In this section, we briefly review the relevant literature on web page segmentation, which can be divided into three categories: DOM-based, vision-based, and fusion approaches. The fusion approaches combine the DOM-based, vision-based, and/or text-based approaches to obtain higher segmentation accuracy or for specific applications. The related work of each category is explained in the following paragraphs.

Most segmentation methods follow a DOM-based approach that divides a page into blocks. For example, the DOM tree representation of a web page is created by using an HTML parser to analyze and extract content, as described by Gupta et al. [97]. The content extractor uses filtering techniques to navigate the DOM tree and modify specific nodes, eliminating non-content nodes in the process. This method, it should be noted, does not perform well on rich format web pages in comparison with textual pages.

Chen et al. [98] propose a method that identifies content blocks using a partitioning algorithm that divides a single content block into several smaller ones. This approach considers the whole page as a single block that it then partitions into constituent subcomponents such as a left sidebar, a right sidebar, a header, a footer, and the main content. Fan et al. [99], meanwhile, propose a Site Style Tree (SST) to capture web page content. In this method, information-rich content is extracted from each node of the SST using entropy thresholding.

Some methods segment web pages using vision-based properties. For example, Kong et al. [100] propose a segmentation approach by using the Spatial Graph Grammar (SGG) without relying on DOM structures. This method directly interprets a web page from its image, instead of DOM structures. Image-processing techniques are used to divide an image into different regions and recognize and classify objects, such as texts, buttons, etc., in each region.

Other methods segment web pages using fusion approaches. In this study, we divide them into two categories of (1) DOM-based and vision-based, and (2) a combination of DOM-based, vision-based, and text-based approaches.

The first category of fusion methods segments web pages using structural features and visual properties. For example, Sanoja et al. [101] propose the Block-O-Matic strategy, inspired by visual-based content segmentation techniques and automated document processing methods. This method includes three phases—analysis, understanding, and reconstruction of the web page. It combines the logical, visual, and structural features of web pages to understand and analyze the content. Manabe et al. [102], meanwhile, propose a method—called HEPS (HEading-based Page Segmentation)—to extract logical structures of web pages. This method uses the HTML tags, computed style calculated by Web browsers based on several factors, and the image height to determine the visual style of the DOM tree nodes of a web page.

A Vision-based Page Segmentation (VIPS) algorithm is proposed by Cai et al. [11, 103]. This algorithm divides a page into fragments based on the visual properties and logical structure (i.e., DOM) of the page. Once again, although this method performs well on traditional pages, it does not perform well on modern web pages. The box clustering segmentation model, introduced by Zeleny et al. [6], uses visual properties of web pages, the distance between elements, and their visual similarity, and follows a three-step procedure: box extraction, computation of distances between boxes, and clustering of boxes. Cormer et al. [104] propose a hierarchal segmentation method that similarly uses the visual properties of the web page to achieve segmentation. The method presented by Mehta et al. [105] uses the VIPS algorithm to divide pages into small fragments based on visual properties, it also uses a pre-trained Naive Bayes classifier to create bigger blocks.

Liu et al. [106] propose the ViDE approach that primarily utilizes the visual features human users can capture on the web pages to perform deep web data extraction, including data record extraction and data item extraction. By using visual features for data extraction, ViDE avoids the limitations of those solutions that need to analyze complex web page source files.

Kumar et al. [107] propose a web page segmentation algorithm that re-DOMs the input page to produce clean and consistent segmentations. The algorithm includes four stages. First, each inline element is identified and wrapped inside a <span> tag. Next, the hierarchy is reshuffled. Third, redundant nodes that do not contribute to the visual layout of the page are removed. Finally, the hierarchy is supplemented to introduce missing structure which is accomplished by computing a set of VIPS-style separators across each page region and inserting enclosing DOM nodes accordingly.

To segment blocks accurately in a manner that simulates human perception in identifying related content, Xu and Miller [22, 27, 28, 73] propose the "Gestalt Layer Merging" (GLM) model, premised on the Gestalt laws of grouping. This method can be used to segment blocks in complex modern web pages. In the research presented in this study, we use this method to generate the basic-blocks.

The second category of fusion methods combines the structural, visual properties, and textual content of a web page to achieve segmentation. For example, Jiang et al. [18] propose a web page segmentation method that uses both visual and logical features of content. Their method uses text density (the number of words in a block of text) as its segmentation algorithm. The densitometric approach proposed by Kohlschütter et al. [10] uses the text density metric to identify blocks of a web page. These approaches, it should be noted, do not consider semantic text similarity metrics, but instead, focus on the structural and visual features of the web page. Table 4.1 represents a comparison of the related work. It specifies the categories of each method. Deep semantic specifies whether or not a method uses the relation of the word to the other words in a document by considering the meaning of sentences.

There is a lack of standard procedures to compare the accuracy of web page segmentation methods [6]. However, to compare the accuracy of segmentation techniques, some research has been carried out. For example, Blustein et al. [108] design experiments to compare web page segmenters by proposing some questions that a segmentation method should answer. The questions are about the purpose of web page segmentation and the dataset used in an experiment. Some studies utilize precision, recall, and F-measure metrics to evaluate the performance of their segmentation approach. For example, Kovacevic et al. [109] and Xu and Miller [22], evaluate the performance of their segmentation approach using these metrics.

Table 4.1: Comparison of Properties of Different Segmentation Methods

Method	DOM Based	Vision Based	Fusion Approaches			
			DOM & Vision Based	DOM & Vision & Text-based		
				No Semantic Text Similarity	Semantic Text Similarity	
					Non-Deep Semantic	Deep Semantic
Gupta et al. [97]	✓					
Chen et al. [98]	✓					
Fan et al. [99]	✓					
Kong et al. [100]		✓				
Sanoja et al. [101]			✓			
Manabe et al. [102]			✓			
Cai et al. [11, 103]			✓			
Zeleny et al. [6]			✓			
Cormer et al. [104]			✓			
Mehta et al. [105]			✓			
Liu et al. [106]			✓			
Kumar et al. [107]			✓			
Xu and Miller [22, 27, 28, 73]			✓			
Jiang et al. [18]				✓		
Kohlschütter et al. [10]				✓		
Current Approach (Fusion-Block) [110]					✓	
Current Approach (Integrated-Block)						✓

Our proposed approaches (Fusion-Block [110] and Integrated-Block) combine the DOM-based structure, vision, and text-based segmentation techniques. As mentioned above, they generate the basic-blocks using the Gestalt laws of grouping and then employ a semantic text similarity method to regroup these related basic-blocks into larger fusion or integrated blocks. In other words, each fusion block or integrated block is composed of a group of basic-blocks with similar text content. It is expected that these enhanced semantically-based, visually-initiated blocks will deliver superior performance across a wide array of tasks on modern, multi-media web pages.

4.3 The Proposed Fusion Web page Segmentation Approach

Web page segmentation keeps related content together as blocks, where each block contains distinctive content. To facilitate the description of our approach, an overview of the segmentation procedure is given in the following paragraph.

4.3.1 Overview

The DOM elements represent the structure of web pages [101]. So far, most of the studies have utilized the DOM elements to segment web page content. Relevant content groups together as blocks; each block contains distinctive content. The goal of web page segmentation is to construct a content structure from web page features that groups the elements of a web page using metrics such as distances, locations, and semantic context. A fusion model which includes DOM, vision, and text-based segmentation approach is required to achieve a superior segmentation result.

4.3.2 Proposed Approaches

Our frameworks are designed and implemented to generate web page segmentation models for different web pages and to overcome the limitations of the former approaches. The main steps of our proposed frameworks are shown in Figure 4.3 and Figure 4.4.

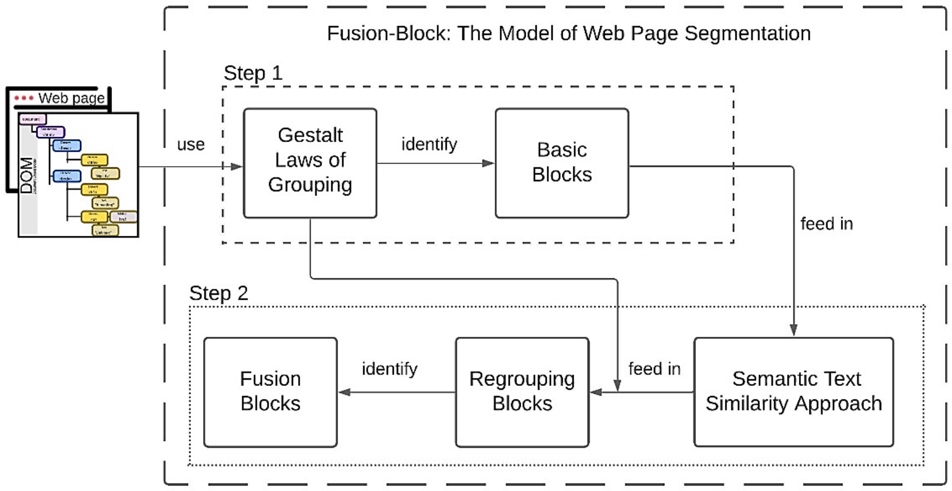


Figure 4.3: The Framework of the Fusion-Block Approach

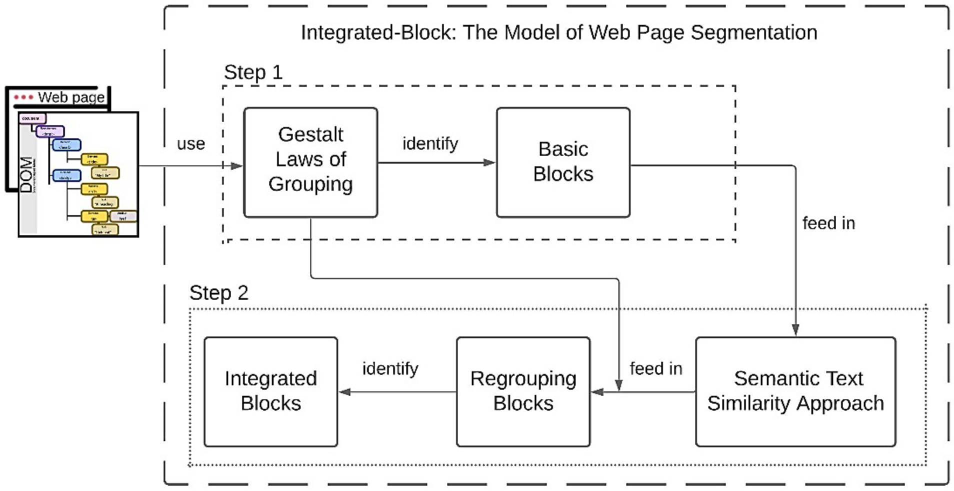


Figure 4.4. The Framework of the Integrated-Block Approach

As shown in Figure 4.3 and Figure 4.4, our models are mainly categorized in two steps; (1) they identify basic-blocks in web pages to segment a web page using the Gestalt laws of grouping technique, and (2) both of the models compare text similarity of basic-blocks identified in step 1 and regroup the semantically related blocks as fusion or integrated blocks using a semantic analysis approach. These steps are explained in the following paragraphs.

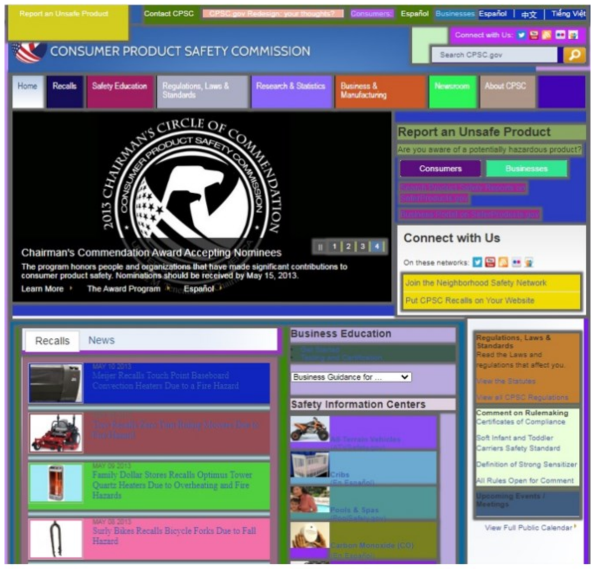
4.3.2.1 Step 1: Basic-Block

For the first step, our models segment web page content into basic-blocks inspired by human understanding. This study uses Gestalt laws to simulate human understanding and perception. To present a web page, a “layer tree” is designed [22]. Nodes of a semantic block tree (layer tree) indicated as basic-blocks are constructed by merging correlated blocks with the Gestalt laws of grouping. The Gestalt Layer Merging (GLM) model includes three components: (1) the layer tree constructor, (2) the Gestalt laws translator, and (3) the web page block identifier [22]. The DOM tree of a web page is taken as a prototype by the layer tree constructor to build up its layer tree. Constructing layer tree nodes is done simultaneously with building up the layer tree and starts from adding the root node to the layer tree, and then executes recursively until all visible DOM elements are extracted and added to the layer tree. Further details of constructing a layer tree can be found in [22].

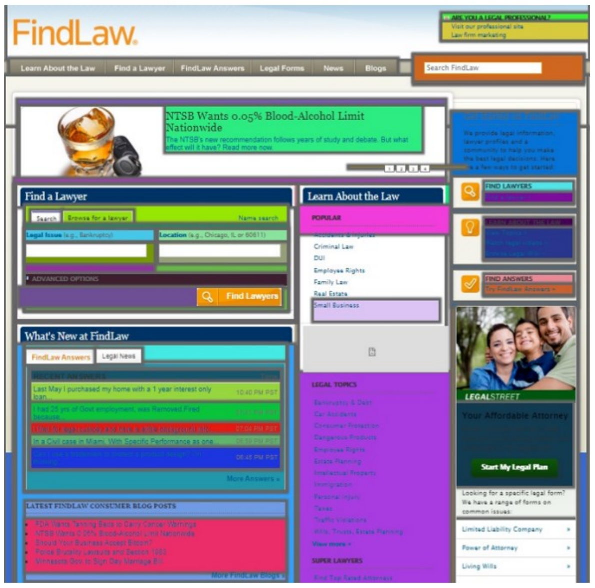
The layer tree is built by removing hierarchical inconsistencies between the DOM tree representation and the visual layout of the web page. In the DOM tree, child elements are located inside their parent elements by default; however, some CSS rules can manipulate locations, such as “position”, “float”, etc. These rules sometimes cause the DOM hierarchy to be misaligned against the visual hierarchy. Therefore, such an inconsistency must be eliminated in the layer tree construction. Also, invisible elements existing in the DOM tree are removed when constructing the layer tree. An invisible DOM element is either an element with an area of 0 (including the borders and shadows), an element without any actual content (text, image, background, etc.), or an element that is completely covered by its visible child elements. A modification is necessary to be done on a layer tree according to [22]. Therefore, a layer tree only extracts visible DOM elements and merges these elements into separate groups according to their semantic meanings. The details of this modification can be found in [22].

The Gestalt laws explain the mechanisms of how humans perceive and understand things. To construct each block of the layer tree, the Gestalt laws translator interprets the Gestalt laws of grouping into six machine compatible rules which are expressed in Chapter 2. Further details can be found in [22].

According to these six laws (Gestalt laws of simplicity, closure, proximity, similarity, continuity, and common fate), a model can allow elements to be categorized whether in a group or not. This group of similar elements includes the results of six laws merged. Our models use a naïve Bayes classifier same as [30] to merge these laws. Figure 4.5 shows the basic-blocks of two web pages (the homepage of the United States Consumer Product Safety Commission "https://www.cpsc.gov" and FindLaw "www.findlaw.com"), where for each basic block, a different background color is assigned. Step 2 of our models is presented to complete the segmentation process by considering semantic textual analysis explained in the following paragraphs.



(a)



(b)

Figure 4.5: An Example of the Basic-Blocks of the Two Web pages, (a) “www.cpsc.gov”, (b) “www.findlaw.com”

4.3.2.2 Step 2: Fusion-Block and Integrated-Block

Human understandings are simulated using Gestalt laws of grouping to identify basic-blocks. According to the Gestalt law of similarity, the similarity between blocks includes three features, (1) background, (2) foreground, and (3) size similarity. Block features such as color, textual styles, width, and height are compared regardless of considering the semantic analysis. Hence, objects may be segmented in different blocks, even though they have semantically related text. It is required to utilize semantic analysis to address this problem. This study utilizes semantic text similarity to identify semantically related blocks and regroup them as a single block. A description of semantic analysis is explained in the following paragraphs.

Natural language processing (NLP) is a series of techniques simulated human techniques of processing and analyzing a language. NLP system inventions are associated with the semantic analysis of linguistic structures [111]. Semantic analysis within the framework of NLP evaluates and represents human language and analyzes texts with an interpretation similar to those of human beings. Recently, text analysis is of one the popular topic of research. It plays an important role in NLP and aims to numerically represent unstructured text documents into a structured form that can be processed and analyzed by computers [112]. Many methods have been proposed to transform raw data (a series of symbols and words) into the form of a vector at character, word, sentence, or document level to express the similarity and dissimilarity between textual elements [113]. Text representation methods are mainly divided into two types: context-free and contextual representation models.

The main step of non-contextual text representation is to map discrete language symbols into a distributed embedding space [114]. Each word of the document is mapped into a vector. Vector representations of text can be constructed in many ways. For example, Mikolov et al. [115, 82] propose Word2Vec, an effective tool for learning word representations from a corpus, which implements two models: Continuous Bag-Of-Words (CBOW) and Skip-gram [116]. The CBOW model scans the text with a context window and learns to predict the target word [117]. The Skip-gram model predicts the words in the context of the target word [93]. Word2Vec uses local neighboring words as context [117]. It can learn word vectors in a short time from a large-scale document and has been applied in different aspects of text processing such as text representation [118].

Another text representation is TD-IDF (Term Frequency-Inverse Document Frequency) [119], based on the bag-of-words philosophy, which involves the assumption that a document is simply a collection of words, and thus, the document can be vectorized by computing the relative importance of each word, i.e., by considering the word's frequency in the document and its popularity in the corpus. TF-IDF and bag-of-words do not consider the semantics of the words and also the ordering of the words is lost, which are the major weaknesses of these methods. For example, assume that the three words of "Country", "City", and "Flower" are equally distant in a document even though the first two words are semantically more related than "Flower".

Pennington et al. [120] propose Global Vectors for word representation (GloVe) that directly captures global corpus statistics. Comparing to GloVe, the word embeddings trained from Word2Vec can better capture the semantics of words and exploit the relatedness of words. FastText is an extension of Word2Vec proposed by Facebook [121]. Instead of using individual words, FastText breaks words into several n-grams (sub-words) [113].

Some research utilizes word embedding [122-124] to understand the semantic logic of the text. Doc2Vec was proposed by Mikolov et al. [81], inspired by Word2Vec. The neural-network-based document embedding known as Doc2Vec extends Word2Vec from the word level to the document level [81]. Each document has its vector values in the same space as words. Thus, the distributed representation for both words and documents is learned simultaneously. Some of the major weaknesses of the bag-of-words models are addressed by Doc2vec. Doc2vec method can be applied to variable-length pieces of text regardless of their length. Furthermore, it does not rely on the parse trees and does not require task-specific tuning of the word weighting function [81].

Doc2Vec starts with training the model, subsequently, a vector space is built based on the word vectors [81], therefore, semantically similar words show similar vector representations (e.g., “City” is more related to “Country” than to “Flower”). By training the word vectors, the document vector is trained, and the document’s numeric representation is held at the end of the training. The document vector represents the concept of a document as the concept of a word is represented by the word vector. Some studies have represented that Doc2Vec results in better classification accuracy than other representation methods in different domains [81, 125-127].

These non-contextual methods have two major limitations: (1) However, the ordering of words in a text is meaningful, these representation models are insensitive to word order and only capture the relations between words [117], and (2) they only obtain a single global representation for each word and ignore their context [95]. Thus, they fail to capture higher-level concepts in context. To address these issues, contextual representation is proposed.

Contextual representation models assign each word a representation based on its context [127]. These models are divided into two categories of unidirectional and bidirectional representations. Unidirectional representation of a word is generated based on the left or right surrounding words in a document [94]. For example, the unidirectional representation of the word “goals” in the sentence “I have three goals in my life”, is based on “I have three” or “in my life”, not both of them. Bidirectional representation of a word considers the left and right surrounding words in a text corpus [94]. For example, a bidirectional representation of the word “goals” in the sentence is generated based on “I have three” and “in my life”. The bidirectional representation model has a deeper sense of context than unidirectional models since it considers the context of a word based on all of its surroundings [117].

Different from non-contextual word representations, contextual representations move beyond word-level semantics where each token is associated with a representation that is a function of the entire input sequence [95]. These representations can capture many syntactic and semantic properties of words under diverse linguistic contexts [117]. Some studies have shown that contextual embeddings, pre-trained on a large-scale unlabeled corpus, can achieve high performance on a wide range of NLP tasks and can avoid training a new model from scratch [95]. Some other studies express those contextual embeddings can learn useful and transferable representations across languages [95]. The contextual representations are better suited to capture the semantics of text [117]. They have some model architectures such as transformer [21]; a neural network architecture based exclusively on attention mechanisms [117]. The neural attention mechanism aims to capture long-range dependencies and is inspired by how humans read and understand longer texts. The neural representation learning can be treated as a pre-training step or language modeling step for NLP downstream tasks [117]. Many methods focus on learning contextual word embeddings such as ELMo [128], GPT [129], and BERT [94]. ELMo and GPT do not consider the left and right surrounding words in a text corpus [95]. Google proposes an improved method, BERT (Bidirectional Encoder Representations from Transformers) [94], which can effectively exploit the deep semantic information of a sentence. Deep bidirectional means that it is conditioned on every word in the left and right contexts at the same time [117]. It works by masking some percentage of the input tokens at random and then predicting those masked tokens. Several studies reported that contextualized embeddings such as BERT better encode semantic information of a text [96]. Also, according to [94], BERT obtains state-of-the-art results in numerous benchmarks. Figure 4.6 shows the explained representation models in a timeline since 2013.

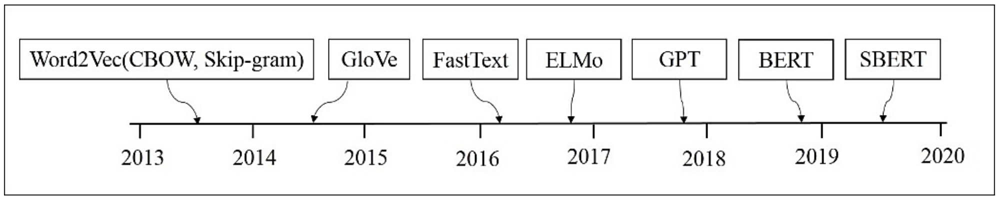


Figure 4.6: A Timeline of the Recent Text Representation Models Since 2013

BERT has set an advanced performance on sentence-pair regression tasks such as semantic textual similarity [130]. A shortcoming of the BERT network structure is that it maps sentences to a vector space that is rather unsuitable to be used with common similarity measures like cosine-similarity. To address this limitation, Reimers et al. [130] modify the pre-trained BERT model and propose Sentence-BERT (SBERT), which uses Siamese and triplet network structures to derive semantically meaningful sentence embeddings [130]. It uses cosine-similarity to compare the similarity between two sentence embeddings. SBERT is trained on the SNLI [131], Multi-Genre NLI [132], and STS benchmark dataset. It is fine-tuned with a 3-way softmax-classifier objective function for one epoch. More details can be found in [130].

In this research, we compare the semantic similarity of nearby blocks using two methods: a non-contextual and a contextual method. In the first approach, we use the Doc2Vec technique to captures both semantic and syntactic information of words and can be used to measure text similarity. We used gensim's [83] implementation of Doc2Vec to determine the semantic similarity of the blocks' text content. Therefore, related blocks are regrouped into a fusion block, which not only forms based on visual features by simulating human perception but also utilizes semantic analysis of blocks to improve web page segmentation.

In the second approach, we use SBERT technique in the same way as [130] to identify semantically related blocks and regroup them as an integrated block. Therefore, integrated blocks form based on visual features by simulating human perception and also utilize semantic analysis.

As it is mentioned earlier, the performance of structural-based segmentation methods can be restricted by the shortcomings of the DOM structure itself. The complex DOM structure leads to shortening or scattering of the long text of a web page content that is hard to extract useful features. Therefore, it is difficult to extract useful features from short text content, which challenges semantic analysis.

Also, paragraphs with similar subjects are separated into different blocks because they contain text with different formats such as different font sizes, font colors, etc. Our proposed approaches (Fusion-Block and Integrated-Block) address these problems using semantic analysis and regrouping the related scattered blocks into a fusion block or an integrated block that contains the longer text. Our semantic regrouping methods are presented in Algorithm 4.1 and Algorithm 4.2 for Fusion-Block and Integrated-Block methods, respectively. In these algorithms, there are two inputs, neighboring (adjacent) basic-blocks, and text difference limit t. (Text difference limit threshold can be set as $t = 0.5$ and $t = 0.4$ for the Fusion-Block and Integrated-Block methods, respectively, which are based on the empirical results presented in section 4.4) These algorithms semantically regroup basic-blocks based on the text similarity and the Gestalt Laws of grouping (proximity and continuity). We used Doc2Vec and SBERT to evaluate the text similarity of the basic-blocks. Our model employs Gestalt Laws of grouping (proximity and continuity) same as [30] to regroup the basic-blocks. The following highlights represent the reasons for using these two laws in the second step of our approaches.

- The Gestalt laws of simplicity, closure, and common fate are already translated and employed in the first step. They are not changed in the second step.
- The Gestalt law of similarity considers the visual features such as background similarity, foreground similarity, and size similarity of blocks. Since our algorithms regroup the related basic-blocks regardless of the text format, the Gestalt law of similarity is not used in this step.
- To group related neighboring blocks, the distance of these blocks needs to be considered. Thus, the Gestalt law of proximity needs to be used in the second step.
- To group related neighboring blocks in terms of considering human perception in segmenting the content of web pages, the Gestalt law of continuity is used in this step.

ALGORITHM 4.1: Semantic Regrouping Algorithm

Input: Two Neighboring Basic Blocks (B_1, B_2), Text Difference Limit *t*

Output: Fusion Block

```
Begin
If $Text\_Similarity^{*}(B_1, B_2) > t$
    $Fusion\_Block \leftarrow Regroup^{**}(B_1, B_2)$;
else
    Break;
End
Return Fusion_Block;
End
```

* We used Doc2Vec algorithm
** The blocks are regrouped according to Gestalt laws

ALGORITHM 4.2: Semantic Regrouping Algorithm

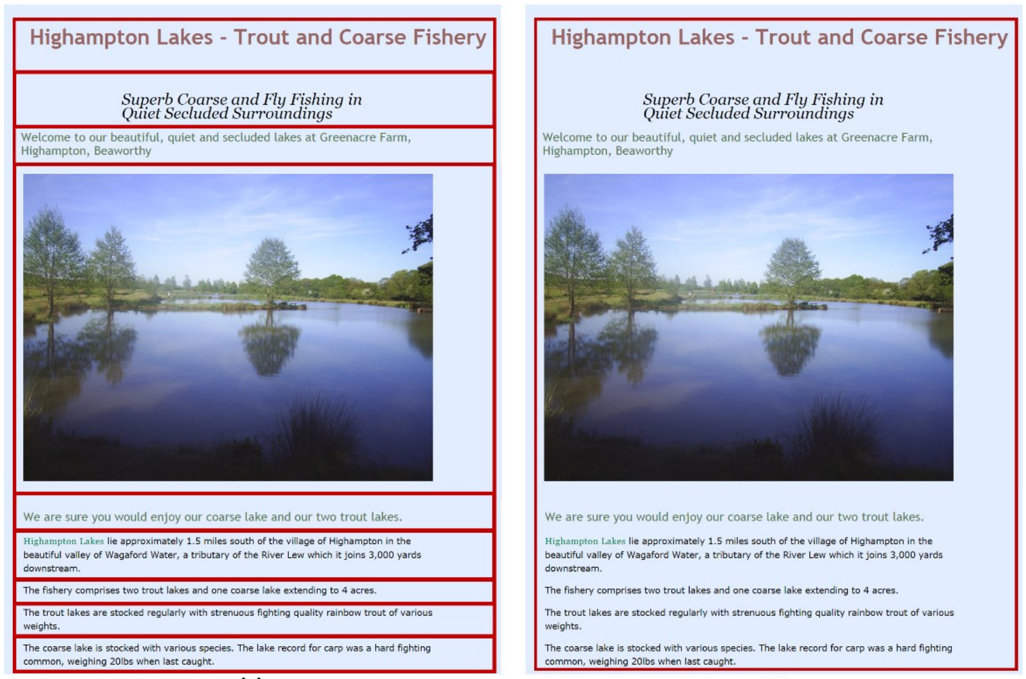
Input: Two Neighboring Basic Blocks (B_1, B_2), Text Difference Limit *t*

Output: Integrated Block

```
Begin
If $Text\_Similarity^{*}(B_1, B_2) > t$
    $Integrated\_Block \leftarrow Regroup^{**}(B_1, B_2)$;
else
    Break;
End
Return Integrated_Block;
End
```

* We used SBERT algorithm
** The blocks are regrouped according to Gestalt laws

We will continue with an example to explain our proposed approach. Figure 4.7 shows a part of the homepage of Highampton Lakes-Trout and Coarse Fishery (www.fishdevon.co.uk). According to part (a) of this figure, the paragraphs with similar subjects are separated into different blocks by current segmentation methods since they contain the text in different formats. Using the semantic analysis method, our models consider these similar paragraphs as a single group regardless of the different text formats as shown in Figure 12 (b).



Highampton Lakes - Trout and Coarse Fishery

Superb Coarse and Fly Fishing in Quiet Secluded Surroundings

Welcome to our beautiful, quiet and secluded lakes at Greenacre Farm, Highampton, Beaworthy

We are sure you would enjoy our coarse lake and our two trout lakes.

Highampton Lakes lie approximately 1.5 miles south of the village of Highampton in the beautiful valley of Wagaford Water, a tributary of the River Lew which it joins 3,000 yards downstream.

The fishery comprises two trout lakes and one coarse lake extending to 4 acres.

The trout lakes are stocked regularly with strenuous fighting quality rainbow trout of various weights.

The coarse lake is stocked with various species. The lake record for carp was a hard fighting common, weighing 20lbs when last caught.

Highampton Lakes - Trout and Coarse Fishery

Superb Coarse and Fly Fishing in Quiet Secluded Surroundings

Welcome to our beautiful, quiet and secluded lakes at Greenacre Farm, Highampton, Beaworthy

We are sure you would enjoy our coarse lake and our two trout lakes.

Highampton Lakes lie approximately 1.5 miles south of the village of Highampton in the beautiful valley of Wagaford Water, a tributary of the River Lew which it joins 3,000 yards downstream.

The fishery comprises two trout lakes and one coarse lake extending to 4 acres.

The trout lakes are stocked regularly with strenuous fighting quality rainbow trout of various weights.

The coarse lake is stocked with various species. The lake record for carp was a hard fighting common, weighing 20lbs when last caught.

(a) **(b)**

Figure 4.7: A Part of the Homepage of "www.fishdevon.co.uk", (a) Segmented Blocks using Current Segmentation Methods (b) Segmented Block using Our Approach

To semantically segment the part of the web page (www.fishdevon.co.uk), firstly, our model identifies basic-blocks according to the Gestalt laws of grouping. These paragraphs have related content; however, they are separated into different basic-blocks. In the second step of our models, the semantic similarity of the extracted basic-bocks are calculated and the semantically textual related blocks can be grouped according to Gestalt laws of proximity and continuity as shown in Algorithm 4.1 and 4.2. Our models regroup these basic-blocks as a single block since the text difference limit of these basic-blocks is greater than t. We utilized the Gestalt laws in addition to the text similarity of blocks to segment a web page according to human perception. After regrouping, the blocks are transformed into bigger fusion or integrated blocks that contain much more stable semantic features than before. These features can be extracted more accurately due to the bigger blocks and longer text sentences and can thus be used to achieve better performance on web page segmentation. Thus, our approaches (Fusion-Block and Integrated-Block) combine DOM structure, visual and textual features of web pages to improve the segmentation accuracy.

4.4 Evaluation

This section presents experiments we have performed to verify the effectiveness of our proposed approaches (Fusion-Block and Integrated-Block). To verify the effectiveness of our approaches, we apply them to open-source datasets. Also, we compare our methods with four existing well-designed algorithms. We use four evaluation metrics for evaluating the performance of our methods: precision, recall, F-1 score, and the Adjusted Rand Index (ARI). Our approaches (the Fusion-Block and Integrated-Block methods) segment a web page into fusion and integrated blocks. The experiments and the results are presented and discussed below.

4.4.1 Research Goal

The goal of this research is to propose a new model of web page segmentation by combining the DOM structure, visual features, and semantic text similarity metrics to achieve better segmentation performance. Our methods (Fusion-Block and Integrated-Block methods) merge web page content into basic-blocks by simulating human perception using the Gestalt laws. Our methods subsequentially identify similar blocks using semantic text similarity and regroup these basic-blocks as fusion and integrated blocks, respectively.

4.4.2 Dataset

The effectiveness of our proposed approach is evaluated against the following three datasets. These datasets are utilized to segment the content of web pages according to human judges (by using the semantic analysis approach).

1. DSpopular, a public dataset of 70 homepages of popular Websites such as "www.foxnews.com", "www.gnu.org", "www.google.com", etc., with manually labeled ground truths for segmentation collected in 2014 [133]. This dataset contains three versions of each page including (1) the basic HTML, (2) a serialized version of the DOM after all external resources are loaded, and (3) a DOM page with manually labeled semantic blocks.
2. DSrandom, a public dataset of 82 homepages of random Websites such as "www.honda.dk", "www.aiact.org", etc. with manually labeled ground truths for segmentation collected in 2014 [134]. Each page includes three versions the same as DSpopular.
3. DSnew, a dataset of 50 homepages of Websites from Alexa Topsites (the first 50 pages) [135] collected in 2017 such as "www.wikipedia.org", "www.facebook.com", etc. These pages are viewed and labeled according to human judges.

The number of web pages, the average, and the median number of blocks in each dataset are shown in Table 4.2. It is crucial to have a ground truth, validated by human assessors to check algorithm correctness. Thus, the accuracy of our proposed methods is evaluated using the manually labeled ground truths provided for each dataset. These datasets are collected from a real-world environment and include type-rich content; therefore, they are suitable for evaluating our method of web page segmentation.

Table 4.2: The Statistics of the Datasets

Dataset	**Number of Web pages**	**Average Number of Blocks**	**Median Number of Blocks**
$DS_{popular}$	70	12.59	16
DS_{random}	82	8.46	13
DS_{new}	50	18.33	12.5

4.4.3 Comparison Methods

We compare our proposed methods (Fusion-Block and Integrated-Block) with the following four well-known existing web page segmentation algorithms. The results show that our methods (Integrated-Block) are superior to all these algorithms in terms of semantic web page segmentation based on human judgment (ground truth).

1. VIPS [11], a well-known approach to segmenting a web page content structure based on its visual representation. This study is used the open-source implementation of VIPS [136] which was utilized in other papers [6, 137]. As the default setting of the tool permitted Degree of Coherence parameter in VIPS is set to 8 the same as [18].
2. BoM [101], a hybrid web page segmentation method that combines structural, visual, and logical features of web pages. This method consists of three phases; analysis, understanding, and reconstruction of a web page.
3. A web page segmentation method [18] combines visual, logic, and features of the content on a web page. For simplicity, we name this segmentation method as SegBlock in this research.
4. Semantic-Block [22], a web page block identification algorithm utilizing the Gestalt laws of grouping to simulate human perception.

4.4.4 Segmentation Accuracy

To quantify the accuracy of an algorithm, it will be necessary to define the metrics that measure the correctness of the result. Blustein et al. [108] propose research questions that segmentation methods should answer. The first question is about the part of web pages a method is used; our approach segments a web page into blocks as part of web pages. The second question is about the purpose of web page segmentation. This study focuses on a new fusion technique of web page segmentation that can be used in various fields such as recreating a web page in ways that can better fit the needs of users, improving the usability of web pages on mobile devices, etc. The other questions mentioned in [108] are about quantifying the accuracy of the algorithm and the dataset that a segmentation method is used. There is a lack of a standard procedure to compare the accuracy of web page segmentation methods [108]. However, precision, recall, and F-score are common metrics of accuracy evaluation in statistical analysis [6]. This study uses datasets provided by ground truths to segment the content of web pages. Thus, we focus on evaluation approaches based on a ground truth such as precision, recall, and F-score. The segmentation result generated by our approach groups the elements of a web page into cohesive regions both visually and semantically. Similar to the previous works [9, 10], we regard each generated segment (block) as a cluster and employ cluster correlation metrics to conduct the evaluation. In data clustering, the Adjusted Rand Index (ARI) is being used to measure the similarity between two clusters [6]. In this study, to verify the accuracy of the segmentation methods computed by our approach and the other four comparison algorithms, we employ the following four evaluation metrics (precision, recall, F-1 score and Adjusted Rand Index (ARI)).

1. Precision represents the ratio of correctly segmented blocks over the blocks segmented by the algorithm as equation 4.1.

$$\text{Precision} = \frac{TP}{TP+FP} \tag{4.1}$$

TP denotes two similar blocks are identified as similar, correctly; while FP indicates that two different blocks are identified as similar, incorrectly.

2. Recall represents the ratio of correctly segmented blocks over the ideal blocks that are manually obtained by humans (ground truth) as equation 4.2.

$$\text{Recall} = \frac{TP}{TP+FN} \tag{4.2}$$

FN indicates that two similar blocks are identified as different, incorrectly.

3. F-1 score which combines precision and recall computed as follows.

$$\text{F} - 1 \text{ score} = 2.\frac{\text{Precision.Recall}}{\text{Precision+Recall}} \tag{4.3}$$

4. Adjusted Rand Index (ARI) [138], which identifies the agreement between two clusters (segmented blocks and ground truth clustering) on a particular

dataset shown in equation 4.4. Value of the Rand Index is between 0 and 1; clusters' agreement on any pair of elements leads to value 1 shows these clusters are the same, and 0 states that the clusters do not agree on any elements. A version of the Rand Index is called ARI which has a value between 0 and 1; 1 shows that two blocks are identical and for random blocks, the value is 0 on average. ARI can be calculated as follow.

Table 4.3: Contingency Table for Comparing Partitions X and Y

Partition		Y				
	Group	y_1	y_2	…	y_s	Sums
X	x_1	n_{11}	n_{12}	…	n_{1s}	a_1
	x_2	n_{21}	n_{22}	…	n_{2s}	a_2
	…	…	…	…	…	…
	x_r	n_{r1}	n_{r2}	…	n_{rs}	a_r
	Sums	b_1	b_2	…	b_s	

Consider a set of n objects $S = \{O_1, O_2, \dots, O_n\}$, and suppose that $X = \{x_1, x_2, \dots, x_r\}$ and $Y = \{y_1, y_2, \dots, y_s\}$ represent two different partitions (blocks) of the objects in S. Given two partitions, X and Y, with r and s subsets, respectively, the contingency Table 4.3 can be formed to indicate group overlap between X and Y as n_{ij}, where $n_{ij} = |x_i \cap y_j|$. In Table 4.3, a generic entry,n_{rs}, represents the number of objects that were partitioned in the rth subset of partition r and the sth subset of partition s [139]. Thus, the ARI can be expressed as:

$$ARI = \frac{\sum_{ij}\binom{n_{ij}}{2} - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] / \binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] / \binom{n}{2}} \qquad (4.4)$$

4.4.5 Evaluation Results

Our experiments were performed on Windows 7 platform (64-bit, Intel Core i7-3770 CPU 3.40 GHz, 16.0 GB RAM). The evaluation of the segmentation algorithm is an important challenge. To compare one algorithm with another, it is crucial to have a ground truth, validated by human assessors to check algorithm correctness. Thus, in this section, we evaluate the performance of our algorithms (Fusion-Block and Integrated-Block) against a ground truth. We also apply the same method for all the four comparison methods (BoM, VIPS, SegBlock, and Semantic-Block) using the four evaluation metrics (precision, recall, F-1 score, and ARI) and explain the results as follows.

All the pages in the datasets have been segmented with our proposed approaches (Fusion-Block and Integrated-Block) and the other four methods. Table 4.4 represents the results of the evaluation metrics in each dataset. "Total" includes the results obtained by using the three datasets (DSpopular, DSrandom, and DSnew) together. The average number of correctly segmented blocks of web pages is represented in the "correct" column for each dataset; a block is said correctly segmented if its geometry and location are equal to a labeled block in the ground truth. According to this table, BoM, VIPS, SegBlock, Semantic-Block, Fusion-Block, and Integrated-Block methods achieve 25.74%, 24.14%, 38.16%, 41.67%, 48.06%, and 53.55% of the average number of correctly labeled blocks in the Total dataset, respectively. Thus, it indicates that our methods (Fusion-Block and Integrated-Block) obtain an improvement in terms of the average number of correctly labeled blocks.

Table 4.4: Evaluation Results

$DS_{popular}$					
	Correct	**Precision**	**Recall**	**F-1 Score**	**ARI**
BoM	2.78	30.5%	26.1%	28.1%	0.452
VIPS	2.78	23.7%	26.2%	24.9%	0.420
SegBlock	5.62	38.1%	40.2%	39.1%	0.530
Semantic-Block	6.75	40.3%	43.4%	41.8%	0.532
Fusion-Block	8.70	44.7%	54.1%	48.9%	0.598

Integrated-Block	9.50	51.7%	61.7%	52.6%	0.632
			DS_{random}		
	Correct	**Precision**	**Recall**	**F-1 Score**	**ARI**
BoM	2.55	30.8%	33.0%	31.8%	0.473
VIPS	1.97	27.8%	26.4%	27.1%	0.371
SegBlock	3.74	41.9%	44.8%	43.3%	0.531
Semantic-Block	3.82	43.3%	53.6%	48.0%	0.549
Fusion-Block	4.54	49.3%	61.2%	54.6%	0.610
Integrated-Block	5.55	56.6%	67.5%	61.5%	0.718
			DS_{new}		
	Correct	**Precision**	**Recall**	**F-1 Score**	**ARI**
BoM	5.54	30.5%	21.7%	25.3%	0.426
VIPS	6.38	20.7%	24.4%	22.4%	0.415
SegBlock	5.59	39.2%	38.9%	39.0%	0.464
Semantic-Block	5.89	40.6 %	41.7%	41.1%	0.483
Fusion-Block	6.02	45.8%	48.2%	47.0%	0.533
Integrated-Block	6.75	52.9%	58.7%	55.4%	0.619
			Total		
	Correct	**Precision**	**Recall**	**F-1 Score**	**ARI**
BoM	3.38	31.4%	27.9%	29.5%	0.450
VIPS	3.17	24.7%	25.8%	25.2%	0.405
SegBlock	5.01	39.6%	42.4%	40.9%	0.514
Semantic-Block	5.47	41.2 %	46.7%	43.8%	0.526
Fusion-Block	6.31	46.3%	54.9%	50.2%	0.583
Integrated-Block	7.03	53.4%	63.2%	57.8%	0.660

According to the table, the Integrated-Block method reaches the highest value of the average number of correctly segmented blocks. The Fusion-Block method [110] reaches the second-highest value of the average number of correctly segmented blocks. This method segments web pages by simulating human perception using the Gestalt laws of grouping. It also compares the textual similarity of blocks using the Doc2Vec algorithm. The third-highest value of the average number of correctly segmented blocks belongs to the Semantic-Block algorithm. This method segments web pages by simulating human perception regardless of textual analysis. We believe that simulating human perception allows this method to achieve the third-highest average number of correctly labeled blocks. The next highest amount of the average number of correctly labeled blocks belongs to the SegBlock method that segments web pages using visual and logical features of content. The number of words within a particular document is used by this method. We believe that SegBlock uses more features compared to BoM and VIPS, and thus, it reaches the fourth maximum average number of blocks after Integrated-Block, Fusion-Block, and Semantic-Block. The VIPS method identifies blocks using visual separators of web pages. The BoM algorithm relies on visual and logical features to segment a web page. Since the layouts of modern web pages are more complicated than before and the visual separators are much less obvious, we believe that BoM achieves a slightly better performance than VIPS in the amount of correctly segmented blocks over the whole dataset.

As shown in Table 4.4, our approach (Integrated-Block) outperforms all five comparison methods in terms of precision, recall, F-1 score, and ARI. It obtains 53.4%, 63.2%, 57.8%, and 0.660 in precision, recall, F-1 score, and ARI, respectively which shows a noticeable improvement in the segmentation's quality. According to this table, Integrated-Block achieves 15.3%, 15.1%, 15.1%, and 13.2% improvements against Fusion-Block (the second-highest amount of the evaluation metrics in the Total dataset) on the precision, recall, F-1 score, and ARI, respectively. The following highlights represent the results of comparing our approach (Integrated-Block) with the other comparison methods.

- On precision, Integrated-Block reaches 70.1%, 116.2%, 34.8%, 29.6%, and 15.3% improvements against BoM, VIPS, SegBlock, Semantic-Block, and Fusion-Block, respectively.
- On recall, Integrated-Block reaches 126.5%, 144.9%, 49.0%, 35.3%, and 15.1% improvements against BoM, VIPS, SegBlock, Semantic-Block, and Fusion-Block, respectively.
- On F-1 score, Integrated-Block reaches 95.9%, 129.3%, 41.3%, 31.9%, and 15.1% improvements against BoM, VIPS, SegBlock, Semantic-Block, and Fusion-Block, respectively.
- On ARI, Integrated-Block reaches 46.6%, 62.9%, 28.4%, 25.5%, and 13.2% improvements against BoM, VIPS, SegBlock, Semantic-Block, and Fusion-Block, respectively.

As shown in Table 4.4, for all the methods, DSrandom dataset has the maximum value of precision, recall, and F-1 score compared to the DSpopular, and the DSnew datasets. It shows that web pages in DSrandom tend to be less complicated (with less content) rather than DSpopular and DSnew.

Our model (Integrated-Block) segments web pages by combining the logic, visual and textual content of a web page using Gestalt laws of grouping to simulate human understandings. It merges web page content into blocks and compares the text similarity of the blocks to regroup these similar blocks as integrated blocks. Fusion-Block uses the text similarity method (Doc2Vec) to compare the text similarity of blocks while the other methods do not use the textual-similarity method to segment web pages. They only focus on the page structure and the visual features without considering the semantic text similarity metrics of blocks. As shown in Table 4.4, the minimum amount of precision, recall, F-1 score, and ARI belong to VIPS since we believe that it uses only the visual features of web pages, which makes it perform less accurately on the evaluation metrics. This table represents that Integrated-Block, Fusion-Block, Semantic-Block, and SegBloak were achieved F-1 score values of more than 40% in the Total dataset, which are 57.8%, 50.2%, 43.8%, and 40.9%, respectively. Additionally, these four methods achieved ARI values greater than 0.5 in the Total dataset that shows their segmentation is not close to randomness. The ARI values of Integrated-Block, Fusion-Block, Semantic-Block, and SegBlock are 0.660, 0.583, 0.526, and 0.514, respectively. Our approach groups semantically similar blocks using the semantic text similarity methods (we used the Doc2Vec and SBERT algorithms shown in Algorithm 4.1 and 4.2 for the Fusion-Block and Integrated-Block methods, respectively).

Table 4.5 and Table 4.6 show different text difference limit thresholds (t in Algorithm 4.1 and 4.2) from 0 to 1. These tables represent that $t = 0.5$ and $t = 0.4$ result in the highest amounts of ARI (comparing to the ground truth) for the Fusion-Block and Integrated-Block methods in the Total dataset which are 0.601 and 0.660, respectively. The ARI distribution over the different text difference limits in the Fusion-Block and Integrated-Block methods are shown in Figure 4.8 and Figure 4.9, respectively. Thus, according to this result, we set $t = 0.5$ and $t = 0.4$ for the Fusion-Block and Integrated-Block methods, respectively.

Table 4.5: ARI Values of Different Threshold for Fusion-Block

Threshold	**0**	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**1**
ARI (Total Dataset)	0	0.085	0.092	0.210	0.584	**0.601**	0.562	0.525	0.528	0.523	0.514

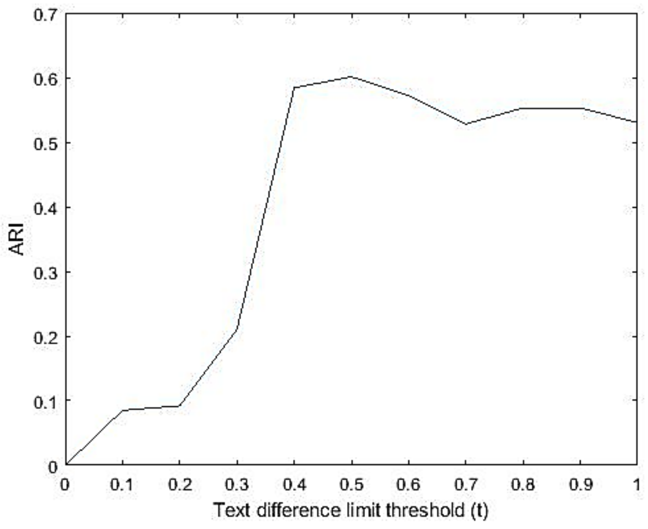


Figure 4.8: ARI Distribution of Different t for Fusion-Block

Table 4.6: ARI Values of Different Threshold for Integrated-Block

Threshold	**0**	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**1**
ARI (Total Dataset)	0	0.056	0.102	0.482	**0.660**	0.575	0.546	0.541	0.530	0.521	0.514

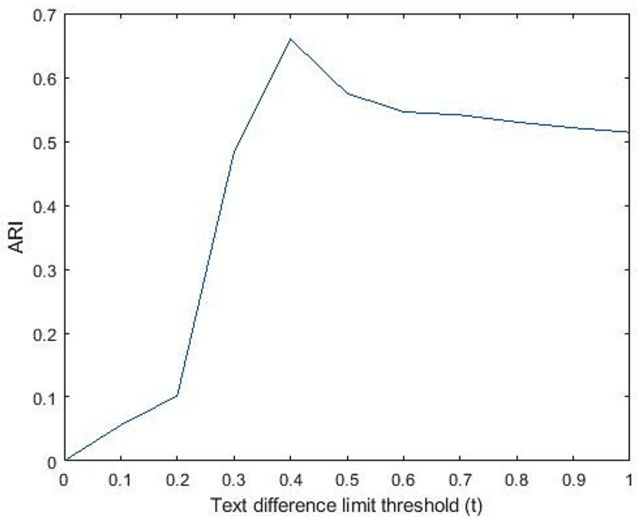
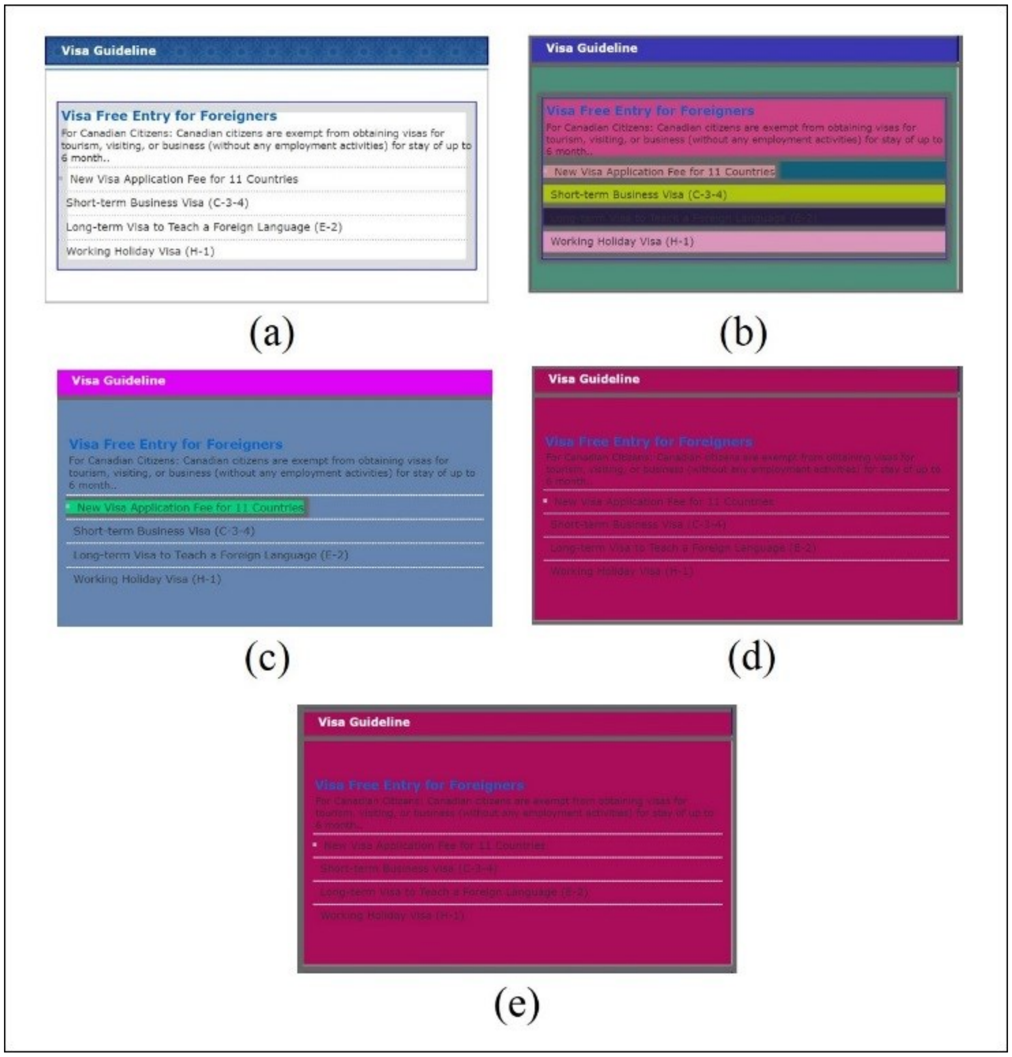


Figure 4.9: ARI Distribution of Different Threshold for Integrated-Block

Figure 4.10, Figure 4.11, and Figure 4.12 represent the result of web page segmentation using four different methods. The segmentation methods are applied on a part of the homepage of “www.koreanconsulate.on.ca”, “www.fishdevon.co.uk”, and “www.journalregister.com” from DSrandom, DSrandom, and DSpopular dataset, respectively. Subfigures (a) of these figures represent the manually labeled ground truth of these web pages. Subfigures (b) and (c) of these figures show the segmented blocks caused by the SegBlock and Semantic-Block, respectively. SegBlock and Semantic-Block methods do not segment blocks using semantic text similarity of blocks; this limitation is indicated and can be found in [18].

The result of segmentation using the SegBlock and the Semantic-Block methods are identical as shown in Figure 4.11 (b), Figure 4.11 (c), Figure 4.12 (b), and Figure 4.12 (c), while they are different in Figure 4.10 (b), and Figure 4.10 (c). As represented in subfigures (b) and (c) of Figures 4.10, 4.11, 4.12, the SegBlock and Semantic-Block methods did not group the whole paragraph; we believe that this is because the different font styles were used in the paragraph. Subfigures (d) and (e) of these figures represent the result of web page segmentation using the Fusion-Block and Integrated-Block methods. As shown in subfigures (d) and (e) of Figure 4.10 and Figure 4.11, the segmented blocks using the Fusion-Block and Integrated-Block methods are identical. This shows that the result of the text similarity using Doc2Vec (Fusion-Block) and SBERT (Integrated-Block) are identical; the paragraphs are related and grouped into a single block. The Fusion-Block method segments the paragraph shown in Figure 4.12 (d) into four separated blocks while these blocks are semantically related and need to be grouped in a single block. However, as represented in Figure 4.12 (e), Integrated-Block groups these related blocks as a single block.



Visa Guideline

Visa Free Entry for Foreigners

For Canadian Citizens: Canadian citizens are exempt from obtaining visas for tourism, visiting, or business (without any employment activities) for stay of up to 6 month..

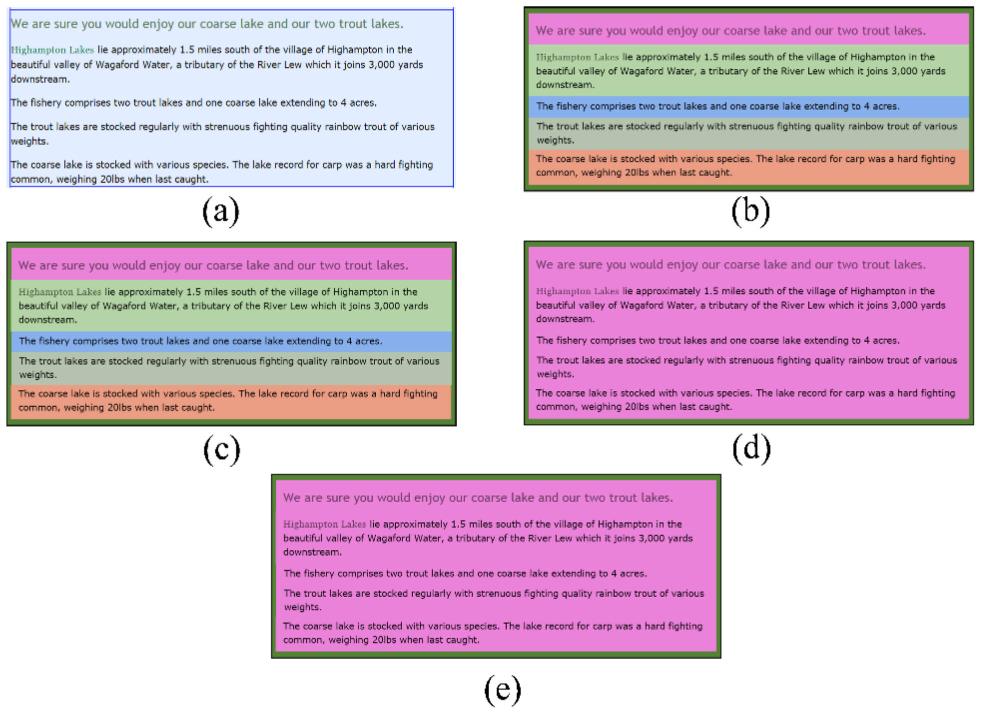
New Visa Application Fee for 11 Countries

Short-term Business Visa (C-3-4)

Long-term Visa to Teach a Foreign Language (E-2)

Working Holiday Visa (H-1)

Figure 4.10: A Part of the Web page of “www.koreanconsulate.on.ca” from DSrandom, (a) Manually Labeled Ground Truth, (b) Segmented Blocks using Segblock, (c) Segmented Blocks using Semantic-Block, (d) Segmented Blocks using Fusion-Block, (e) Segmented Blocks using Integrated-Block


(a)

We are sure you would enjoy our coarse lake and our two trout lakes.

Highampton Lakes lie approximately 1.5 miles south of the village of Highampton in the beautiful valley of Wagaford Water, a tributary of the River Lew which it joins 3,000 yards downstream.

The fishery comprises two trout lakes and one coarse lake extending to 4 acres.

The trout lakes are stocked regularly with strenuous fighting quality rainbow trout of various weights.

The coarse lake is stocked with various species. The lake record for carp was a hard fighting common, weighing 20lbs when last caught.

(b)

We are sure you would enjoy our coarse lake and our two trout lakes.

Highampton Lakes lie approximately 1.5 miles south of the village of Highampton in the beautiful valley of Wagaford Water, a tributary of the River Lew which it joins 3,000 yards downstream.

The fishery comprises two trout lakes and one coarse lake extending to 4 acres.

The trout lakes are stocked regularly with strenuous fighting quality rainbow trout of various weights.

The coarse lake is stocked with various species. The lake record for carp was a hard fighting common, weighing 20lbs when last caught.

(c)

We are sure you would enjoy our coarse lake and our two trout lakes.

Highampton Lakes lie approximately 1.5 miles south of the village of Highampton in the beautiful valley of Wagaford Water, a tributary of the River Lew which it joins 3,000 yards downstream.

The fishery comprises two trout lakes and one coarse lake extending to 4 acres.

The trout lakes are stocked regularly with strenuous fighting quality rainbow trout of various weights.

The coarse lake is stocked with various species. The lake record for carp was a hard fighting common, weighing 20lbs when last caught.

(d)

We are sure you would enjoy our coarse lake and our two trout lakes.

Highampton Lakes lie approximately 1.5 miles south of the village of Highampton in the beautiful valley of Wagaford Water, a tributary of the River Lew which it joins 3,000 yards downstream.

The fishery comprises two trout lakes and one coarse lake extending to 4 acres.

The trout lakes are stocked regularly with strenuous fighting quality rainbow trout of various weights.

The coarse lake is stocked with various species. The lake record for carp was a hard fighting common, weighing 20lbs when last caught.

(e)

We are sure you would enjoy our coarse lake and our two trout lakes.

Highampton Lakes lie approximately 1.5 miles south of the village of Highampton in the beautiful valley of Wagaford Water, a tributary of the River Lew which it joins 3,000 yards downstream.

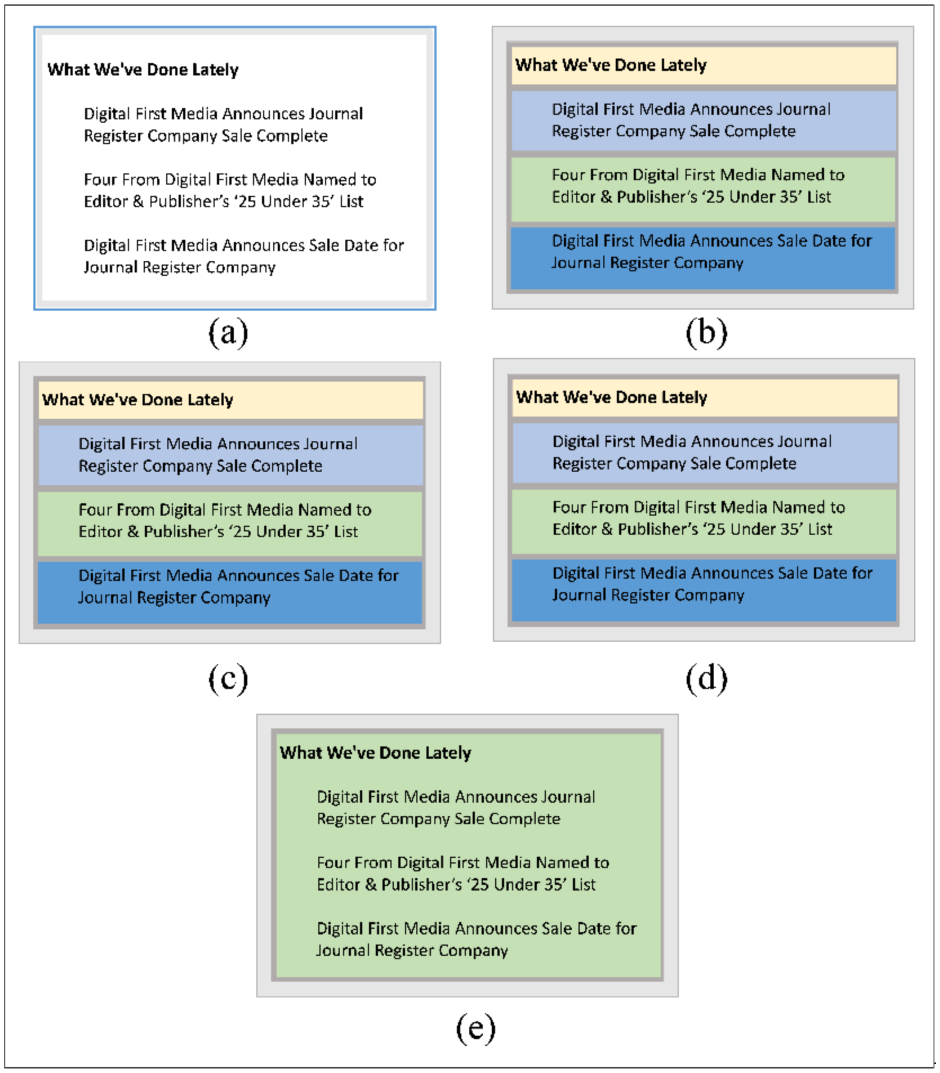
The fishery comprises two trout lakes and one coarse lake extending to 4 acres.

The trout lakes are stocked regularly with strenuous fighting quality rainbow trout of various weights.

The coarse lake is stocked with various species. The lake record for carp was a hard fighting common, weighing 20lbs when last caught.

Figure 4.11: A Part of the Web page of "www.fishdevon.co.uk" from DSrandom, (a) Manually Labeled Ground Truth, (b) Segmented Blocks using Segblock, (c) Segmented Blocks using Semantic-Block, (d) Segmented Blocks using Fusion-Block, (e) Segmented Blocks using Integrated-Block

As shown in Figure 4.10 (b) and Figure 4.10 (c), the Semantic-Block method segmented the blocks better than SegBlock, we believe that it is because Semantic-Block simulates human perception using Gestalt laws of grouping. Also, as shown in subfigures (d) of Figure 4.10 and Figure 4.11, the Fusion-Block method segmented the blocks better than SegBlock and Semantic-Block methods; it uses text similarity of blocks using Doc2Vec technique and regroups similar blocks as a single block. The Integrated-Block method uses the deep semantic analysis method and grouped the whole paragraph as a single block regardless of the different font styles. Thus, Integrated-Block method groups similar semantic text contents as an integrated block and overcomes the scattering or shortening of the long text of web page content mentioned in Sections 4.1 and 4.2.



What We've Done Lately

Digital First Media Announces Journal Register Company Sale Complete

Four From Digital First Media Named to Editor & Publisher's '25 Under 35' List

Digital First Media Announces Sale Date for Journal Register Company

(a)

What We've Done Lately

Digital First Media Announces Journal Register Company Sale Complete

Four From Digital First Media Named to Editor & Publisher's '25 Under 35' List

Digital First Media Announces Sale Date for Journal Register Company

(b)

What We've Done Lately

Digital First Media Announces Journal Register Company Sale Complete

Four From Digital First Media Named to Editor & Publisher's '25 Under 35' List

Digital First Media Announces Sale Date for Journal Register Company

(c)

What We've Done Lately

Digital First Media Announces Journal Register Company Sale Complete

Four From Digital First Media Named to Editor & Publisher's '25 Under 35' List

Digital First Media Announces Sale Date for Journal Register Company

(d)

What We've Done Lately

Digital First Media Announces Journal Register Company Sale Complete

Four From Digital First Media Named to Editor & Publisher's '25 Under 35' List

Digital First Media Announces Sale Date for Journal Register Company

(e)

Figure 4.12: A Part of the Web page of "www.journalregister.com" from DSpopular, (a) Manually Labeled Ground Truth, (b) Segmented Blocks using Segblock, (c) Segmented Blocks using Semantic-Block, (d) Segmented Blocks using Fusion-Block, (e) Segmented Blocks using Integrated-Block

4.5 Conclusions and Future Work

4.5.1 Conclusions

In this research, we present two new segmentation models (Fusion-Block and Integrated-Block) to semantically segment web pages into fusion and integrated blocks. Our models merge web page content into basic-blocks by simulating human perception using Gestalt laws of grouping. Additionally, they utilize semantic text similarity to identify similar blocks and regroup these similar blocks as fusion and integrated blocks, respectively. To verify the effectiveness of our approach, we (1) applied it to the open-source public datasets, (2) compared it with the four existing state-of-the-art algorithms. The results show that both of our approaches outperform all the comparison methods in terms of precision, recall, F-1 score, and ARI. Comparing our methods together, Integrated-Block outperforms the Fusion-Block in terms of all the mentioned metrics.

4.5.2 Limitations and Future Work

We demonstrate that the segmentation models proposed in this research (Fusion-Block and Integrated-Block) outperform the existing methods. However, there are also limitations in this segmentation technique which we plan to address in future work. In the Fusion-Block method, we used Doc2Vec to compare the textual similarity of basic-blocks to regroup them in a single block using Gestalt laws of grouping. In the Integrated-Block method, we used SBERT to compare the textual similarity of basic-blocks to regroup them as integrated blocks by simulating human perception using Gestalt laws of grouping. Figure 4.13 represents several segmented blocks using our approach (Integrated-Block) on a part of the web page of “www.irs.gov” from the DSpopular dataset. These blocks are considered as a single group in the ground truth which means that they have a similar concept. However, our proposed approach segments this part of the web page into five different blocks. It shows that the text representation model that we used (SBERT) does not group these blocks as a single block. Using fusion semantic analysis methods may yield better results when paragraphs have the same concept during the regrouping stage.

The representativeness of the material used for evaluation questions the external validity of the study. The presented evaluation results on the non-representative datasets may not be generalized to web pages in general. We intend to test our models on additional datasets and utilize fusion semantic analysis methods to extend our model in future work.

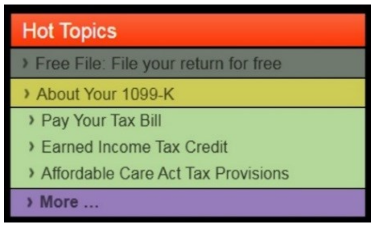

Hot Topics
› Free File: File your return for free
› About Your 1099-K
› Pay Your Tax Bill
› Earned Income Tax Credit
› Affordable Care Act Tax Provisions
› More ...

Figure 4.13: A part of the Web page of “www.irs.gov” from DSpopular

Chapter 5

Conclusions and Future Work

The primary objectives of this thesis are to examine the research gaps and improve the techniques that are commonly used during the analysis of large-scale web applications for inferring user behavior models. With this goal in mind, this thesis presents a solution to generate behavior models using Markov models and the Reinforcement Learning (RL) technique. The proposed method is a fully automated non-instrumented approach that incrementally generates the models.

To model user behaviors it is ideal to analyze web pages of an application. Web page analysis methods utilize web page segmentation to categorize the structure and content of a web page. One of the main factors in segmenting a web page into different blocks is how the extracted content is processed to retrieve distinct information [6]. In this research, we studied different web page segmentation algorithms and proposed two new models of web page segmentation named Fusion-Block and Integrated-Block.

In this section, at first, we present a detailed summary of our contributions in this thesis followed by a discussion on opportunities for future work.

5.1 Overall Contributions

In par with the structural configuration of this thesis, we categorize the primary contributions of this thesis in the following three sub-sections:

5.1.1 Model Inference

Today's users are spending more time on web applications [29, 30]. Many users browse applications and navigate through different web pages. They may have different and evolving interests, especially when it comes to large-scale applications. Knowing and predicting the different user behaviors are crucial factors that may directly affect the success of the application. Underestimating the importance of this factor may lead to software limitations which can easily lead to lost audiences.

However, it is almost impossible to accurately predict and address all of the users' interaction expectations. Inferring a model by predicting and analyzing users' navigational behaviors is necessary to understand users' interests. Developers can identify popular or problematic pages of applications and therefore improve the application design.

The mainstream approach towards capturing the user behaviors consists of monitoring the usage of the system and subsequently mining possible interaction patterns. Some existing solutions instrument web pages to track users' navigation actions, for example, Google Analytics [2], while others analyze log files such as [3]. However, current solutions suffer from several limitations. Some approaches lack generality, for example, they need to infer users' navigational behaviors to support specific tasks [4, 5]. On the other hand, the general frameworks simply return a set of statistics or patterns that are useful to understand the preferences of the system's users but cannot be directly used to evaluate and analyze web page limitations. In this research, we proposed a behavior model using Markov models and Reinforcement Learning to automatically and incrementally, learn users' interests [36]. We evaluated the utility of our approach by using it on a large-scale mobile and desktop application. Then, we analyzed the models using a model checker. By analyzing the inferred model, the application limitations are found. Next, we compared our method with the 19 most popular behavior model generation algorithms in terms of 9 factors such as whether an approach supports real-life application or not. It is believed that this is the first time such behavioral models have been applied to a mission-critical, real-world large-scale application.

5.1.2 Fusion-Block: A New Semantic Approach to Improve Web page Segmentation

The World Wide Web has become a massive repository of information. Thus, identifying, and categorizing distinct informational elements from web pages has become increasingly difficult. Web page segmentation provides a solution to this problem. Web page segmentation is the process of partitioning a web page into blocks, in a manner, where each block contains distinctive content. Also, humans tend to segment a web page based on their understanding, thus, it is important to generate a segmentation model to segment a page by simulating human perception. Most research extracts and organizes content relying on the DOM structure of an HTML page [8, 140]. Some researchers prefer to segment web pages using visual information in a web page. This vision-based segmentation method focuses on the analysis of visual features of the document content as they are perceived by a human reader. It exploits visual clues such as font size, font color, background color, spaces between paragraphs, etc. [19]. Some segmentation methods have been carried out using Natural Language Processing (NLP) techniques [10, 20]. However, these methods do not consider semantic analysis to categorize web pages. Semantic analysis includes extracting text from segmented blocks, computing textual similarity, and regrouping blocks.

This research presented a new method of web page segmentation by combining the DOM structure, visual features, and text similarity metrics to improve the segmentation performance. Our approach [110] generated a segmentation model by utilizing human perception and semantic analysis of a web page. To achieve this, our model merges the content of a web page into basic-blocks and identifies similar blocks using text similarity, and regroups these similar blocks as fusion blocks. Thus, a fusion block is composed of related blocks in terms of similar text contents using NLP algorithms. To evaluate the accuracy of our approach, we applied it to three public datasets and compared the technique with 4 state-of-the-art methods. The results show that our proposed approach outperforms the existing web page segmentation methods, in terms of higher accuracy.

5.1.3 Integrated-Block: A New Combination Model to Improve Web page Segmentation

In this research, we presented the Integrated-Block approach to improve the Fusion-Block segmentation method. This approach merges the DOM structure, vision-based, and text-based similarity metrics of web pages. We demonstrated the utility of transformer technology [21] as a vehicle for the text-based process. Also, we represented the validity of this method by an empirical comparison against the 5 current state-of-the-art techniques. The results represented that our proposed approach outperforms the five other existing web page segmentation methods, in terms of higher accuracy.

5.2 Opportunities for Future Work

In this thesis, several avenues have been explored that aims at generating user behavior models and web page segmentation methods. Our research has been presented in detail in Chapters 3 and 4 of this thesis. However, like any experimental work, all the solutions presented in this thesis can be further pursued and improved in different ways. In this section, we present a set of recommendations for future work for each of the presented approaches in this thesis.

- Chapter 3 of this thesis presents a novel approach to (1) generate user behavioral models for mobile and desktop web applications, (2) automatically calculate the states' rewards, (3) annotate and analyze the models to verify the quantitative properties, and (4) address some limitations found in existing approaches. Just like any other study, this research offers some opportunities for future work. To characterize different groups of users, system experts can define a different set of user-classes. Classes categorize users based upon a set of common features. Also, the proposed behavior model can be applied to other large-scale web applications. In addition, this approach can be extended to apply to any probabilistic timed automata to capture user behaviors.

- Chapter 4 of this thesis presents two novel web page segmentation methods named Fusion-Block and Integrated-Block, respectively. These methods merge the DOM structure, vision-based, and text-based similarity metrics of web pages. To improve the segmentation accuracy, we demonstrated the utility of NLP as a technique for the text-based process. For future work, these methods can be applied to additional datasets.
- According to Section 3.4, the Extractor function collects the information of the retrieved URLs. In this research, we used visual and text-based approaches for information extraction of a detected URL. Also, we used three different methods to compare the similarity of web pages based on the words' extraction of each page. This study can be further enhanced by using Fusion-Block and Integrated-Block methods to compare web pages.
- Also, a recommender system can be generated using the experiments of the behavior model inference approach on the applications to recommend popular web pages to the users.

References

[1] J. Srivastava, R. Cooley, M. Deshpande, and P.-N. Tan, "Web usage mining: discovery and applications of usage patterns from Web data," SIGKDD Explor. Newsl., vol. 1, no. 2, pp. 12-23, 2000.

[2] Google Analytics. Available: http://www.google.com/intl/en/analytics/

[3] F. M. Facca and P. L. Lanzi, "Mining interesting knowledge from weblogs: a survey," Data & Knowledge Engineering, vol. 53, no. 3, pp. 225-241, 2005.

[4] R. R. Sarukkai, "Link prediction and path analysis using Markov chains," Computer Networks, vol. 33, no. 1-6, pp. 377-386, 2000.

[5] Q. Yang and H. H. Zhang, "Web-log mining for predictive web caching," IEEE Transactions on Knowledge and Data Engineering, vol. 15, no. 4, pp. 1050-1053, 2003.

[6] J. Zeleny, R. Burget, and J. Zendulka, "Box clustering segmentation: A new method for vision-based web page preprocessing," Information Processing & Management, vol. 53, no. 3, pp. 735-750, 2017.

[7] S. Baluja, "Browsing on small screens: recasting web-page segmentation into an efficient machine learning framework," in Proceedings of the 15th international conference on World Wide Web, 2006, pp. 33-42.

[8] L. Bing, R. Guo, W. Lam, Z.-Y. Niu, and H. Wang, "Web page segmentation with structured prediction and its application in web page classification," in Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval, 2014, pp. 767-776.

[9] D. Chakrabarti, R. Kumar, and K. Punera, "A graph-theoretic approach to webpage segmentation," in Proceedings of the 17th international conference on World Wide Web, 2008, pp. 377-386.

[10] C. Kohlschütter and W. Nejdl, "A densitometric approach to web page segmentation," in Proceedings of the 17th ACM conference on Information and knowledge management, 2008, pp. 1173-1182.

[11] D. Cai, S. Yu, J.-R. Wen, and W.-Y. Ma, "Vips: a vision-based page segmentation algorithm," Microsoft technical report, MSR-TR-2003-79.

[12] F. A. Silva, A. C. Domingues, and T. R. B. Silva, "Discovering mobile application usage patterns from a large-scale dataset," ACM Transactions on Knowledge Discovery from Data (TKDD), vol. 12, no. 5, pp. 1-36, 2018.

[13] K.-W. Lim, S. Secci, L. Tabourier, and B. Tebbani, "Characterizing and predicting mobile application usage," Computer Communications, vol. 95, pp. 82-94, 2016.

[14] H. Kawazu, F. Toriumi, M. Takano, K. Wada, and I. Fukuda, "Analytical method of web user behavior using Hidden Markov Model," in 2016 IEEE International Conference on Big Data (Big Data), 2016, pp. 2518-2524: IEEE.

[15] G. Wang, X. Zhang, S. Tang, C. Wilson, H. Zheng, and B. Y. Zhao, "Clickstream user behavior models," ACM Transactions on the Web (TWEB), vol. 11, no. 4, pp. 1-37, 2017.

[16] K. Samarasinghe and S. Jayalal, "Prediction of User Intentions Using Web History," in 2019 International Research Conference on Smart Computing and Systems Engineering (SCSE), 2019, pp. 17-23: IEEE.

[17] P. Prakash and A. Jaya, "Analyzing and predicting user behavior pattern from weblogs," International Journal of Applied Engineering Research, vol. 11, no. 9, pp. 6278-6283, 2016.

[18] Z. Jiang, H. Yin, Y. Wu, Y. Lyu, G. Min, and X. Zhang, "Constructing Novel Block Layouts for Webpage Analysis," ACM Transactions on Internet Technology (TOIT), vol. 19, no. 3, pp. 1-18, 2019.

[19] J. Kang, J. Yang, and J. Choi, "Repetition-based web page segmentation by detecting tag patterns for small-screen devices," IEEE Transactions on Consumer Electronics, vol. 56, no. 2, pp. 980-986, 2010.

[20] H. F. Eldirdiery and A. Ahmed, "Web document segmentation for better extraction of information: a review," 2015.

[21] A. Vaswani et al., "Attention is all you need," arXiv preprint arXiv:1706.03762, 2017.

[22] Z. Xu and J. Miller, "Identifying semantic blocks in Web pages using Gestalt laws of grouping," World Wide Web, vol. 19, no. 5, pp. 957-978, 2016.

[23] S. E. Palmer, "Modern theories of Gestalt perception," Mind & Language vol. 5, no. 4, pp. 289-323, 1992.

[24] R. J. Sternberg and K. Sternberg, Cognitive psychology. Nelson Education, 2016.

[25] K. Koffka, Principles of Gestalt psychology. vol. 44, 2013. Routledge.

[26] H. Stevenson, "Emergence: The Gestalt Approach to Change," Unleashing Executive and Orzanizational Potential. Retrieved 7.

[27] Z. Xu and J. Miller, "Cross-browser differences detection based on an empirical metric for web page visual similarity," ACM Transactions on Internet Technology (TOIT), vol. 18, no. 3, pp. 1-23, 2018.

[28] Z. Xu and J. Miller, "A new webpage classification model based on visual information using gestalt laws of grouping," in International Conference on Web Information Systems Engineering, 2015, pp. 225-232: Springer.

[29] Q. Xu, J. Erman, A. Gerber, Z. Mao, J. Pang, and S. Venkataraman, "Identifying diverse usage behaviors of smartphone apps," in Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference, 2011, pp. 329-344.

[30] Z. Tu et al., "Your apps give you away: distinguishing mobile users by their app usage fingerprints," Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, vol. 2, no. 3, pp. 1-23, 2018.

[31] E. Manavoglu, D. Pavlov, and C. L. Giles, "Probabilistic user behavior models," in Third IEEE International Conference on Data Mining, 2003, pp. 203-210: IEEE.

[32] D. Lorenzoli, L. Mariani, and M. Pezz, "Automatic generation of software behavioral models," presented at the Proceedings of the 30th international conference on Software engineering, Leipzig, Germany, 2008.

[33] C. Ghezzi, M. Pezz, M. Sama, and G. Tamburrelli, "Mining behavior models from user-intensive web applications," presented at the Proceedings of the 36th International Conference on Software Engineering, Hyderabad, India, 2014.

[34] G. Greco, A. Guzzo, L. Pontieri, and D. Sacca, "Discovering expressive process models by clustering log traces," IEEE Transactions on Knowledge and Data Engineering, vol. 18, no. 8, pp. 1010-1027, 2006.

[35] T. Zhu, R. Greiner, and G. Häubl, "Learning a Model of a Web User's Interests," Berlin, Heidelberg, 2003, pp. 65-75: Springer Berlin Heidelberg.

[36] S. S. Sajjadi-Ghaemmaghami, S. S. Emam, and J. Miller, "Automatically Inferring User Behavior Models in Large-Scale Web Applications," Information and Software Technology, p. 106704, 2021.

[37] https://w3techs.com/technologies/details/ta-googleanalytics.

[38] https://trends.builtwith.com/analytics/Google-Analytics.

[39] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM: probabilistic model checking for performance and reliability analysis," SIGMETRICS Perform. Eval. Rev., vol. 36, no. 4, pp. 40-45, 2009.

[40] J. A. Dean, C. Anderson, and A. Battle, "Ranking documents based on user behavior and/or feature data," U.S. Patent 7716225 B1, 2016. Available: https://patents.google.com/patent/US7716225B1

[41] Y. Yue, R. Patel, and H. Roehrig, "Beyond position bias: examining result attractiveness as a source of presentation bias in clickthrough data," presented at the Proceedings of the 19th international conference on World wide web, Raleigh, North Carolina, USA, 2010.

[42] J. Huang, R. W. White, G. Buscher, and K. Wang, "Improving searcher models using mouse cursor activity," presented at the Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval, Portland, Oregon, USA, 2012.

[43] K. Wang, N. Gloy, and X. Li, "Inferring search behaviors using partially observable Markov (POM) model," presented at the Proceedings of the third ACM international conference on Web search and data mining, New York, New York, USA, 2010.

[44] H.-J. Kim and A. D. Corduneanu, "Modifying search result ranking based on implicit user feedback and a model of presentation bias," U.S. Patent 8938463 B1, 2015. Available: https://patents.google.com/patent/US8938463B1

[45] T. Paul, D. Puscher, and T. Strufe, "The User Behavior in Facebook and its Development from 2009 until 2014," CoRR, vol. abs/1505.04943, 2015.

[46] R. Omar, A. O. M. Tap, and Z. S. Abdullah, "Web usage mining: A review of recent works," in The 5th International Conference on Information and Communication Technology for The Muslim World (ICT4M), 2014, pp. 1-5: IEEE.

[47] K. Zhu, Z. Liu, L. Zhang, and X. Gu, "A mobile application recommendation framework by exploiting personal preference with constraints," Mobile Information Systems, vol. 2017, 2017.

[48] E. H.-C. Lu and Y.-W. Yang, "Mining mobile application usage pattern for demand prediction by considering spatial and temporal relations," Geoinformatica, vol. 22, no. 4, pp. 693-721, 2018.

[49] L. Vassio, I. Drago, M. Mellia, Z. B. Houidi, and M. L. Lamali, "You, the web, and your device: Longitudinal characterization of browsing habits," ACM Transactions on the Web (TWEB), vol. 12, no. 4, pp. 1-30, 2018.

[50] M. Gan and K. Xiao, "R-RNN: Extracting user recent behavior sequence for click-through rate prediction," IEEE Access, vol. 7, pp. 111767-111777, 2019.

[51] Y. Chu, H.-K. Yang, and W.-C. Peng, "Predicting Online User Purchase Behavior Based on Browsing History," in 2019 IEEE 35th International Conference on Data Engineering Workshops (ICDEW), 2019, pp. 185-192: IEEE.

[52] S. G. Langhnoja, M. P. Barot, and D. B. Mehta, "Web usage mining using association rule mining on clustered data for pattern discovery," International Journal of Data Mining Techniques and Applications, vol. 2, no. 1, pp. 141-150, 2013.

[53] D. Jagli and S. Oswal, "Web Usage Mining: Pattern Discovery and Forecasting," arXiv preprint arXiv:1310.2375, 2013.

[54] L. Zhou, Y. Liu, J. Wang, and Y. Shi, "Utility-based web path traversal pattern mining," in Seventh IEEE International Conference on Data Mining Workshops (ICDMW 2007), 2007, pp. 373-380: IEEE.

[55] X. Lei, "Modeling and Intelligent Analysis of Web User Behavior of WEB User Behavior," in 2018 International Conference on Engineering Simulation and Intelligent Control (ESAIC), 2018, pp. 192-195: IEEE.

[56] X. Luo, J. Wang, Q. Shen, J. Wang, and Q. Qi, "User behavior analysis based on user interest by web log mining," in 2017 27th International Telecommunication Networks and Applications Conference (ITNAC), 2017, pp. 1-5: IEEE.

[57] L. Vassio, M. Mellia, F. Figueiredo, A. P. C. da Silva, and J. M. Almeida, "Mining and modeling web trajectories from passive traces," in 2017 IEEE International Conference on Big Data (Big Data), 2017, pp. 4016-4021: IEEE.

[58] Y. Xuefeng, "User Personalized Recommendation Model Based on Web Log Mining," in 2017 International Conference on Smart Grid and Electrical Automation (ICSGEA), 2017, pp. 580-584: IEEE.

[59] S. Schechter, M. Krishnan, and M. D. Smith, "Using path profiles to predict HTTP requests," Computer Networks and ISDN Systems, vol. 30, no. 1–7, pp. 457-467, 1998. Elsevier BV

[60] F. Chierichetti, R. Kumar, P. Raghavan, and T. Sarlos, "Are web users really Markovian?," presented at the Proceedings of the 21st international conference on World Wide Web, Lyon, France, 2012.

[61] K. Suneetha and R. Krishnamoorthi, "Identifying user behavior by analyzing web server access log file," IJCSNS International Journal of Computer Science and Network Security, vol. 9, no. 4, pp. 327-332, 2009.

[62] C. Bertero, M. Roy, C. Sauvanaud, and G. Trédan, "Experience report: Log mining using natural language processing and application to anomaly detection," in 2017 IEEE 28th International Symposium on Software Reliability Engineering (ISSRE), 2017, pp. 351-360: IEEE.

[63] T. Hussain, S. Asghar, and N. Masood, "Web usage mining: A survey on preprocessing of web log file," in 2010 International Conference on Information and Emerging Technologies, 2010, pp. 1-6: IEEE.

[64] M.-T. Nguyen, T.-D. Diep, T. H. Vinh, T. Nakajima, and N. Thoai, "Analyzing and visualizing web server access log file," in International Conference on Future Data and Security Engineering, 2018, pp. 349-367: Springer.

[65] S. S. Emam and J. Miller, "Test Case Prioritization Using Extended Digraphs," ACM Trans. Softw. Eng. Methodol., vol. 25, no. 1, pp. 1-41, 2015.

[66] C. Szepesvári, "Algorithms for Reinforcement Learning," Synthesis Lectures on Artificial Intelligence and Machine Learning, vol. 4, no. 1, 2010. Morgan & Claypool Publishers.

[67] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, vol. 9, no. 5, Cambridge, Massachusetts: The MIT Press, 1998.

[68] M. Wiering and M. v. Otterlo, Reinforcement Learning (State of the Art), Springer, 2012.

[69] L. Long-Ji, "Self-improving reactive agents based on reinforcement learning, planning and teaching," Machine Learning, journal article vol. 8, no. 3, pp. 293-321, May 01 1992.

[70] C. J. Watkin, "Learning From Delayed Rewards," Ph.D dissertation, King's College, University of Cambridge, 1989.

[71] P. Dayan and C. J. Watkin, "Technical Note Q,-Learning," Machine Learning vol. 8, no. 3, pp. 279-292 1992. Springer Netherlands

[72] DavidLo, L. Mariani, and M. Santoro, "Learning extended FSA from software: An empirical assessment," Systems and Software, vol. 85, no. 9, pp. 2063-2076, 2012. ELSEVIER.

[73] Z. Xu and J. Miller, "Estimating similarity of rich internet pages using visual information," International Journal of Web Engineering and Technology, vol. 12, no. 2, 2017.

[74] A. Shahbazi and J. Miller, "Extended Subtree: A New Similarity Function for Tree Structured Data," IEEE Transactions on Knowledge & Data Engineering, vol. 26, pp. 864-877, April 2014.

[75] A. Mesbah, A. v. Deursen, and S. Lenselink, "Crawling Ajax-Based Web Applications through Dynamic Analysis of User Interface State Changes," ACM Trans. Web, vol. 6, no. 1, pp. 1-30, 2012.

[76] Available: https://www.nltk.org/

[77] Y. Liu, Z. Liu, T.-S. Chua, and M. Sun, "Topical Word Embeddings," presented at the AAAI Conference on Artificial Intelligence 2015.

[78] Z. S. Harris, "Distributional Structure," in Papers in Structural and Transformational LinguisticsDordrecht: Springer Netherlands, 1970, pp. 775-794.

[79] P. D. Turney and P. Pantel, "From Frequency to Meaning: Vector Space Models of Semantics," Artificial Intelligence Research, vol. 37, pp. 141-188, 2010.

[80] D. Kiela, F. Hill, and S. Clark, "Specializing Word Embeddings for Similarity or Relatedness," in Empirical Methods in Natural Language Processing Lisbon, Portugal, 2015, pp. 2044–2048: Association for Computational Linguistics.

[81] Q. Le and T. Mikolov, "Distributed Representations of Sentences and Documents," presented at the Proceedings of the 31st International Conference on Machine Learning, Proceedings of Machine Learning Research, 2014, pp. 1188-1196. Available: http://proceedings.mlr.press

[82] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," arXiv preprint arXiv:1301.3781, 2013.

[83] gensim. Available: https://radimrehurek.com/gensim/

[84] G. Salton and M. J. McGill, Introduction to Modern Information Retrieval. McGraw-Hill, Inc., 1986, p. 400.

[85] C. Sitaula, "Semantic text clustering using enhanced vector space model using Nepali language," Computer Sciences and Telecommunications vol. 4, no. 36, pp. 41-46, 2012. Scientific and Educational Organization "Internet Academy"

[86] H. Hansson and B. Jonsson, "A logic for reasoning about time and reliability," Formal Aspects of Computing, journal article vol. 6, no. 5, pp. 512-535, September 01 1994.

[87] M. Kwiatkowska, G. Norman, and D. Parker, "Stochastic Model Checking," in Formal Methods for Performance Evaluation: 7th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM 2007, Bertinoro, Italy, May 28-June 2, 2007, Advanced Lectures, M. Bernardo and J. Hillston, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 220-270.

[88] C. Dehnert, S. Junges, J.-P. Katoen, and M. Volk, "A Storm is Coming: A Modern Probabilistic Model Checker," Cham, 2017, pp. 592-600: Springer International Publishing.

[89] Z. Bu, C. Zhang, Z. Xia, and J. Wang, "An FAR-SW based approach for webpage information extraction," Information Systems Frontiers, vol. 16, no. 5, pp. 771-785, 2014.

[90] H. F. Eldirdiery and A. Ahmed, "Web document segmentation for better extraction of information: a review," International Journal of Computer Applications, vol. 110, no. 3, 2015.

[91] C. Kohlschütter, P. Fankhauser, and W. Nejdl, "Boilerplate detection using shallow text features," in Proceedings of the third ACM international conference on Web search and data mining, 2010, pp. 441-450.

[92] G. Wen, X. Pan, L. Jiang, and J. Wen, "Modeling Gestalt laws for classification," in 9th IEEE International Conference on Cognitive Informatics (ICCI'10), 2010, pp. 914-918: IEEE.

[93] S. Wang, W. Zhou, and C. Jiang, "A survey of word embeddings based on deep learning," Computing, vol. 102, no. 3, pp. 717-740, 2020.

[94] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," arXiv preprint arXiv:1810.04805, 2018.

[95] Q. Liu, M. J. Kusner, and P. Blunsom, "A survey on contextual embeddings," arXiv preprint arXiv:2003.07278, 2020.

[96] A. Rogers, O. Kovaleva, and A. Rumshisky, "A primer in bertology: What we know about how bert works," Transactions of the Association for Computational Linguistics, vol. 8, pp. 842-866, 2020.

[97] S. Gupta, G. Kaiser, D. Neistadt, and P. Grimm, "DOM-based content extraction of HTML documents," in Proceedings of the 12th international conference on World Wide Web, 2003, pp. 207-214.

[98] Y. Chen, W.-Y. Ma, and H.-J. Zhang, "Detecting web page structure for adaptive viewing on small form factor devices," in Proceedings of the 12th international conference on World Wide Web, 2003, pp. 225-233.

[99] Q. Fan, C. Yan, and L. Huang, "Discovering Informative Contents of Web Pages," in International Conference on Web-Age Information Management, 2014, pp. 180-191: Springer.

[100] J. Kong et al., "Web interface interpretation using graph grammars," IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 42, no. 4, pp. 590-602, 2011.

[101] A. Sanoja and S. Gançarski, "Block-o-matic: A web page segmentation framework," in 2014 international conference on multimedia computing and systems (ICMCS), 2014, pp. 595-600: IEEE.

[102] T. Manabe and K. Tajima, "Extracting logical hierarchical structure of HTML documents based on headings," Proceedings of the VLDB Endowment, vol. 8, no. 12, pp. 1606-1617, 2015.

[103] D. Cai, S. Yu, J.-R. Wen, and W.-Y. Ma, "Extracting Content Structure for Web Pages Based on Visual Representation," Berlin, Heidelberg, 2003, pp. 406-417: Springer Berlin Heidelberg.

[104] M. Cormer, R. Mann, K. Moffatt, and R. Cohen, "Towards an improved vision-based web page segmentation algorithm," in 2017 14th Conference on Computer and Robot Vision (CRV), 2017, pp. 345-352: IEEE.

[105] R. R. Mehta, P. Mitra, and H. Karnick, "Extracting semantic structure of web documents using content and visual information," in Special interest tracks and posters of the 14th international conference on World Wide Web, 2005, pp. 928-929.

[106] W. Liu, X. Meng, and W. Meng, "Vide: A vision-based approach for deep web data extraction," IEEE Transactions on Knowledge and Data Engineering, vol. 22, no. 3, pp. 447-460, 2009.

[107] R. Kumar, J. O. Talton, S. Ahmad, and S. R. Klemmer, "Bricolage: A Structured-Prediction Algorithm for Example-Based Web Design," Proc. CHI 2011, 2011.

[108] J. Blustein, N. R. D. Matteo, and D. Macrini, "Designing Experiments to Compare Web Page Segmenters," presented at the Proceedings of the 2nd International Workshop on Human Factors in Hypertext, Hof, Germany, 2019. Available: https://doiorg.login.ezproxy.library.ualberta.ca/-10.1145/3345509.3349280

[109] M. Kovacevic, M. Diligenti, M. Gori, and V. Milutinovic, "Recognition of Common Areas in a Web Page Using Visual Information: a possible application in a page classification," in 2002 IEEE International Conference on Data Mining, 2002. Proceedings., 2002, pp. 250-257: IEEE.

[110] S. S. Sajjadi-Ghaemmaghami and J. Miller, "A New Semantic Approach to Improve Webpage Segmentation," Journal of Web Engineering, vol. 20_4, pp. 963–992, June 2021, DOI: 10.13052/jwe1540-9589.2042.

[111] J. Hirschberg and C. D. Manning, "Advances in natural language processing," vol. 349, no. 6245, pp. 261-266, 2015. American Association for the Advancement of Science.

[112] G. Liu, C. Guo, L. Xie, W. Liu, N. Xiong, and G. Chen, "An intelligent CNN-VAE text representation technology based on text semantics for comprehensive big data," arXiv preprint arXiv:2008.12522, 2020.

[113] J. Yan, "Text Representation," in Encyclopedia of Database Systems, L. Liu and M. T. ÖZsu, Eds. Boston, MA: Springer US, 2009, pp. 3069-3072.

[114] X. Qiu, T. Sun, Y. Xu, Y. Shao, N. Dai, and X. Huang, "Pre-trained models for natural language processing: A survey," Science China Technological Sciences, pp. 1-26, 2020.

[115] T. Mikolov, Q. V. Le, and I. Sutskever, "Exploiting similarities among languages for machine translation," arXiv preprint arXiv:1309.4168, 2013.

[116] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," arXiv preprint arXiv:1310.4546, 2013.

[117] K. Babić, S. Martinčić-Ipšić, and A. Meštrović, "Survey of Neural Text Representation Models," Information, vol. 11, no. 11, p. 511, 2020.

[118] T. Ming, Z. Lei, and Z. Xianchun, "Document vector representation based on Word2vec," Computer Science, vol. 43, no. 6, pp. 214-217, 2016.

[119] S. Robertson, "Understanding inverse document frequency: on theoretical arguments for IDF," Journal of documentation, 2004.

[120] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), 2014, pp. 1532-1543.

[121] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," arXiv preprint arXiv:1607.01759, 2016.

[122] Y. Wang, Z. Liu, and M. Sun, "Incorporating linguistic knowledge for learning distributed word representations," PloS one, vol. 10, no. 4, 2015.

[123] M. Alsuhaibani, D. Bollegala, T. Maehara, and K.-i. Kawarabayashi, "Jointly learning word embeddings using a corpus and a knowledge base," PloS one, vol. 13, no. 3, 2018.

[124] Y. Li et al., "Incorporating knowledge into neural network for text representation," Expert Systems with Applications, vol. 96, pp. 103-114, 2018.

[125] J. H. Lau and T. Baldwin, "An empirical evaluation of doc2vec with practical insights into document embedding generation," arXiv preprint arXiv:1607.05368, 2016.

[126] C. Xing, D. Wang, X. Zhang, and C. Liu, "Document classification with distributions of word vectors," in Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2014 Asia-Pacific, 2014, pp. 1-5: IEEE.

[127] A. Miaschi and F. Dell'Orletta, "Contextual and Non-Contextual Word Embeddings: an in-depth Linguistic Investigation," in Proceedings of the 5th Workshop on Representation Learning for NLP, 2020, pp. 110-119.

[128] M. E. Peters et al., "Deep contextualized word representations," arXiv preprint arXiv:1802.05365, 2018.

[129] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018.

[130] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," arXiv preprint arXiv:1908.10084, 2019.

[131] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning, "A large annotated corpus for learning natural language inference," arXiv preprint arXiv:1508.05326, 2015.

[132] A. Williams, N. Nangia, and S. R. Bowman, "A broad-coverage challenge corpus for sentence understanding through inference," arXiv preprint arXiv:1704.05426, 2017.

[133] "dataset-popular 2014. A dataset of popular pages (taken from dir.yahoo.com) with manually marked up semantic blocks. Retrieved from https://github.com/rkrzr/dataset-popular.," ed.

[134] "dataset-random 2014. A dataset of random pages with manually marked up semantic blocks. Retrieved from https://github.com/rkrzr/dataset-random.," ed.

[135] "Alexa. 2016. The top 500 sites on the web. Retrieved from http://www.alexa.com/topsites.," ed.

[136] VIPS-JAVA [n.d.]. Implementation of Vision Based Page Segmentation Algorithm in Java. Retrieved from https://github.com/tpopela/vips-java.

[137] A. S. Bozkir and E. A. Sezer, "Layout-based computation of web page similarity ranks," International Journal of Human-Computer Studies, vol. 110, pp. 95-114, 2018.

[138] L. Hubert and P. Arabie, "Comparing partitions," Journal of classification, vol. 2, no. 1, pp. 193-218, 1985.

[139] K. Y. Yeung and W. L. Ruzzo, "Details of the adjusted rand index and clustering algorithms, supplement to the paper an empirical study on principal component analysis for clustering gene expression data," Bioinformatics, vol. 17, no. 9, pp. 763-774, 2001.

[140] J. Jiang, X. Song, N. Yu, and C. Lin, "FoCUS: Learning to Crawl Web Forums," IEEE Transactions on Knowledge and Data Engineering, vol. 25, pp. 1293-1306, 2013.

Appendix A

Additional Scholarly Contributions

Poster:

Saeedeh Sajjadi Ghaemmaghami, Sepideh Imam, Ralf Vierich, and James Miller, "An Automated Reward Calculation Approach for User Intensive Behavioral Models", Poster presented at: IBM CASCON 2016, 8500 Warden Ave, Markham, Toronto, ON L6G 1A5