REPUBLIC OF TURKEY
YILDIZ TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

END-TO-END AUTOMATED UI TESTING FRAMEWORK FOR WEB SITES WITH INTENSIVE USER-SYSTEM INTERACTIONS

Ramazan Faruk OĞUZ

MASTER OF SCIENCE THESIS
Department of Computer Engineering
Program of Computer Engineering

Supervisor
Assoc. Prof. Dr. Mehmet Sıddık AKTAS

June, 2022

REPUBLIC OF TURKEY
YILDIZ TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

END-TO-END AUTOMATED UI TESTING FRAMEWORK FOR WEB SITES WITH INTENSIVE USER-SYSTEM INTERACTIONS

A thesis submitted by Ramazan Faruk OĞUZ in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE** is approved by the committee on 22.06.2022 in Department of Computer Engineering, Program of Computer Engineering.

Assoc. Prof. Dr. Mehmet Sıddık
AKTAS
Yildiz Technical University
Supervisor

Approved By the Examining Committee

Assoc. Prof. Dr. Mehmet Sıddık AKTAS, Supervisor
Yildiz Technical University _____

Prof. Dr. Oya KALIPSIZ, Member
Yildiz Technical University _____

Assoc. Prof. Dr. Hasan Sozer, Member
Ozyegin University _____

I hereby declare that I have obtained the required legal permissions during data collection and exploitation procedures, that I have made the in-text citations and cited the references properly, that I haven't falsified and/or fabricated research data and results of the study and that I have abided by the principles of the scientific research and ethics during my Thesis Study under the title of End-to-End Automated UI Testing Framework for Web Sites with Intensive User-System Interactions supervised by my supervisor, Assoc. Prof. Dr. Mehmet Sıddık AKTAS. In the case of a discovery of false statement, I am to acknowledge any legal consequence.

Ramazan Faruk OĞUZ

Signature

This study was supported by the Scientific and Technological Research Council of Turkey (TUBITAK) under project id: 3191534.

Dedicated to my family

ACKNOWLEDGEMENTS

First and foremost, I would like to express my sincere gratitude to my supervisor Assoc. Prof. Dr. Mehmet Sıddık Aktaş for his patience, continuous support and motivation throughout all stages of preparing this thesis and his valuable contributions to the publications we produced. His scholarly advice and scientific approach helped me to a very great extent to accomplish these tasks.

I also owe great thanks to all my application development friends who helped me with my study. I would like to thank my family, who have stand by me through thick and thin throughout my life, for their moral support and constant encouragement. I have once again realized the importance of my family's support on this long journey.

I am grateful to TUBITAK for supporting this research (under project id: 3191534) and Saha Bilgi Teknolojileri Egitim Danismanlık San. Tic. A.S. R&D Center for providing the computational materials that made this study achievable.

Ramazan Faruk OĞUZ

TABLE OF CONTENTS

LIST OF SYMBOLS viii

LIST OF ABBREVIATIONS ix

LIST OF FIGURES x

LIST OF TABLES xi

ABSTRACT xii

ÖZET xiii

1 INTRODUCTION 1
- 1.1 Abstract . . . 1
- 1.2 Objective of the Thesis . . . 2
- 1.3 Purpose . . . 2
- 1.4 Hypothesis &Contributions . . . 2
- 1.5 Organization . . . 3

2 GENERAL INFORMATION AND LITERATURE REVIEW 4
- 2.1 GAN . . . 4
- 2.2 Literature Review . . . 5
  - 2.2.1 Large-scale Graph Data Processing For UI Testing . . . 5
  - 2.2.2 Extracting Information from Large Scale Graph Data . . . 7

3 RESEARCH PROBLEMS 9

4 PROPOSED METHODOLOGY FOR EMBEDDING APPROACHES FOR USER BROWSING BEHAVIOUR 10
- 4.1 Labeled URLs . . . 10
- 4.2 Usage on Labeled URLs . . . 10

5 PROPOSED METODOLOGY FOR UI TEST SCRIPT GENERATION WORKFLOW 13
- 5.1 Crawling Module . . . 13
- 5.2 Browsing Graph Data Clustering Module . . . 15

5.3 Mapping Based Data Representation Module 15
5.4 GAN Based Sequence Generator Module 15
5.5 Filtering Module 16
5.6 Sequence2Script Module 18

6 PROTOTYPE IMPLEMENTATION **19**
6.1 Datasets 19
6.2 Test Design 20

7 EVALUATION **22**
7.1 Dataset 22
7.2 Experimental Study Design 23
7.3 Evaluation Results 24
7.4 Threats to Validity of Results 25

8 CONCLUSION **26**

REFERENCES **27**

PUBLICATIONS FROM THE THESIS **30**

LIST OF SYMBOLS

w_i	i^{th} word in the current sentence
N	number of words

LIST OF ABBREVIATIONS

GAN	Generative Adversarial Network
UI	User Interface
emb_dim	Embedding Dimension
hid_dim	Hidden Dimension
gen_seq_num	Generated Sequence Number
seqGAN	Sequence Generative Adversarial Nets
SDLC	Software Development Life Cycle
BDD	Behavior-driven Development
GPU	Graphics Processing Unit
AWS	Amazon Web Services

LIST OF FIGURES

Figure 4.1 Not clustered graph of navigation 11
Figure 4.2 Clustered graph of navigation 11
Figure 4.3 Markdown table for not clustered graph testing 12
Figure 4.4 Markdown table for clustered graph testing 12
Figure 5.1 Overview of the proposed pipeline 14
Figure 5.2 An example of the browsing data 14
Figure 5.3 The mapping process used in the system 16
Figure 5.4 Building Trie and HashMap on example training sequences 18

LIST OF TABLES

Table 5.1 Sample dataset 15
Table 5.2 Objective Function 17
Table 7.1 Best scores for Mother&Baby Products site 24
Table 7.2 Best parameters for Mother&Baby Products site 24
Table 7.3 Best scores for Loadium site 24
Table 7.4 Best parameters for Loadium site 25
Table 7.5 Best scores for Testinium site 25
Table 7.6 Best Parameters for Testinium Site 25

ABSTRACT

End-to-End Automated UI Testing Framework for Web Sites with Intensive User-System Interactions

Ramazan Faruk OĞUZ

Department of Computer Engineering
Master of Science Thesis

Supervisor: Assoc. Prof. Dr. Mehmet Sıddık AKTAS

Even though a large-scale graph structure is a powerful model to solve several challenging problems in various applications' domains today, it can also preserve various attributions regarding user behavior, especially in the e-commerce domain. Information extraction is a promising research area in deep learning algorithms using large-scale graph data. This thesis focuses on understanding users' implicit navigational behavior on an e-commerce site that we can represent with the large-scale graph data. We propose a GAN-based e-business workflow by leveraging the large-scale browsing graph data and the footprints of navigational users' behavior on the e-commerce site. With this method, we have discovered various frequently repeated clickstream data sequences, which do not appear in training data at all. Therefore, a prototype application is developed to demonstrate performance tests on the proposed business e-workflow. The experimental studies that are conducted in this thesis show that the proposed methodology produces noticeable and reasonable outcomes for our prototype application.

Keywords: Deep learning, large scale graph data, GAN, distributed e-business workflows, distributed systems

YILDIZ TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

ÖZET

Yoğun Kullanıcı-Sistem Etkileşimli Web Siteleri için Uçtan Uca Otomatik Kullanıcı Arayüzü Test Metodolojisi

Ramazan Faruk OĞUZ

Bilgisayar Mühendisliği Anabilim Dalı
Yüksek Lisans Tezi

Danışman: Doç. Dr. Mehmet Sıddık AKTAŞ

Büyük ölçekli bir grafik yapısı, günümüzde çeşitli uygulama alanlarındaki birçok zorlu sorunu çözmek için güçlü bir model olsa da, özellikle e-ticaret alanında, kullanıcı davranışıyla ilgili çeşitli öznitelikleri koruyabilir. Bilgi çıkarma, büyük ölçekli grafik verilerini kullanan derin öğrenme algoritmalarında umut verici bir araştırma alanıdır. Bu tez, büyük ölçekli grafik verileriyle temsil edebileceğimiz bir e-ticaret sitesinde kullanıcıların örtülü gezinme davranışlarını anlamaya odaklanmaktadır. Büyük ölçekli tarama grafiği verilerinden ve e-ticaret sitesinde gezinme kullanıcılarının davranışlarının ayak izlerinden yararlanarak GAN tabanlı bir e-iş akışı öneriyoruz. Bu yöntemle, eğitim verilerinde hiç görünmeyen, sık tekrarlanan çeşitli tıklama akışı veri dizilerini keşfettik. Bu nedenle, önerilen iş e-iş akışı üzerinde performans testlerini göstermek için bir prototip uygulama geliştirilmiştir. Bu tezde yürütülen deneysel çalışmalar, önerilen metodolojinin prototip uygulamamız için fark edilir ve makul sonuçlar ürettiğini göstermektedir.

Anahtar Kelimeler: Derin öğrenme, büyük ölçekli grafik verisi, GAN, dağıtık e-iş akışı, dağıtık sistemler

YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

1
INTRODUCTION

It is very important to be able to test the user interface on websites where human user interaction is high. The user interface is one of the software components that affects the customer satisfaction of a software. There is a need for frameworks that will enable extensive testing of user interfaces.

Testing is the process step in which the least amount of time is spread in the software development process steps and the maximum number of errors is expected to be fixed as soon as possible. It is the software phase where the expectation is too high, but very little time is allocated for this high expectation. Therefore, there is a need for frameworks that are fast, automated and will catch errors comprehensively and, can deal with large-scale graph data.

1.1 Abstract

We encounter large-scale graph data in many different areas. Extracting information from this type of data for various purposes using deep learning algorithms is a promising research area currently being studied. This study focuses on understanding user behaviour on an e-commerce site from large-scale graph data. We are proposing a GAN-based business e-workflow as a solution to this problem. We analyze graph data with deep learning algorithms by transforming the traces left by users on the e-commerce site. With this method, we can detect frequently repeated clickstream-based data sequences.

There are tools such as Selenium and Cucumber that can test the current user interface. Here, these tools process standard known and written inputs by test engineers as inputs. These test scripts only run the relevant tests when they receive them as input. Another situation is that the test engineers create the test scripts completely from their own perspectives and deepen on their own experiences while creating the user interface tests. They cannot perform the automatic generation of test scripts. While

the test stages are being carried out [1], there are studies where User interfaces that have been used for a certain period of time are tested, not at the time of t0. In addition to these, studies also do not allow testing of a user interface at time t0.

1.2 Objective of the Thesis

Within the scope of this study, our purpose is to create a framework that will performs end to end user interface testing. To achieve this purpose, we developed a prototype application to test the proposed e-business workflow. We performed performance tests on the prototype application we developed. The results we have obtained from the experimental study revealed that the proposed methodology works and gives successful results.

Motivation: We argue that automating the testing process and generating the testing scripts are highly important in software engineering. Based on the standard software development practices, we already know that the user interface is constantly changing. On the other hand, the logical flow of the back-engine codes and services does not change so frequently. Due to the changes in user interfaces, there is a need to generate new click-stream-based data sequences that the end-users can use and then provide automating testing of the data sequences in advance. Our motivation is to design and implement an e-business workflow that can automate UI testing by dynamically generating and testing new data sequences that the end-user might use.

1.3 Purpose

Comprehensive creation of tests for a UI costs time for programmers. Of course, the URLs that make up the websites must also be produced correctly in the test environment and run in the appropriate test steps. It is essential to reduce the effort here and automate the work. In order to meet this need, which we see in the literature, and to develop existing solutions, we have prepared a workflow for the automatic generation of test scripts. We propose a flow based on an e-commerce start page, followed by automatically generating URL flows for test cases.

1.4 Hypothesis &Contributions

Our study proposes a UI testing framework that can be automated for websites with heavy usage, without user interaction before. We claim that before a website meets its users, we can still create new test sequences for such an interface.

We suggest that it can also be used for websites of different styles, as it does not need any prior data to perform this task. In this way, we argue that without collecting user data within the websites, user data can be generated artificially and thus the user interface of the relevant website can be tested in detail.

1.5 Organization

Chapter 2 presents the literature review on automated UI testing and deep learning approaches. Chapter 3 describes the research problem we focused on, while Chapter 4 and 5 presents the proposed methodology. Chapter 6 details the prototype of the proposed methodology. Chapter 7 represents the evaluation metrics and the results of the proposed framework. Finally, chapter 8 concludes the thesis.

2
GENERAL INFORMATION AND LITERATURE REVIEW

Our study focuses on analyzing and learning from browsing graph data for enabling automating UI testing. Software testing includes requirement analysis, test case design, test case writing, test code development, test execution, test report preparation etc. [2]. In the test phase, there are too many scenarios for the software to be tested. Usually, business analysts create these scenarios.Then, the testers perform the tests semi-automatically or automatically by utilizing the software. Selenium is an example of software that automatically performs tests. It tests all of the operations for the functional requirements on the interface. The testing process of the UI interfaces must follow specific steps in the user interface so that it can test some of the functional requirements [3]. For example, to buy a dress, the user must first enter the dress category, put the dress in the basket, and then complete the buying steps. Hence, automating the testing process and generating the testing scripts are highly important in UI testing for efficiency and time savings. To address this need, we propose a workflow that can learn from the large-scale graph data to generate automated testing scripts.

In e-commerce web applications, the user interfaces are constantly changing. Conversely, the processing steps and services at the back-engine of these e-commerce websites are generally less subjected to change. As a result, alternatives are continually increasing from the beginning to the last desired page. Due to these changes and the complexity of the business processes at the user interfaces, the need to renew or re-create new test data as the page changes appear. Hence, there is a need for automated testing methodology that can learn user behavior from large-scale behavioral data.

2.1 GAN

Since our study aims in sequence production, we decided to use GAN networks, a new production model that has proven successful. In this study, we use a customized

SeqGAN network for sequence generation. In our approach, LSTM and CNN are used respectively in generator and discriminator networks that form SeqGAN. The additional information about how sequences are generated using SeqGan can be found in [4]. The steps of generating sequences using SeqGan are explained in detail by the author.

First, the generator is trained to generate negative samples for training the discriminator network. This generator training and the discriminator training using these negative samples are called pre-training of generator and discriminator. Until SeqGan converges, the following two producers take place: (1) generate a sample from the generator and then compute a Q value and update the generator parameters via policy gradient g-steps times. (2): Generate a group of negative samples and combine them with the positive samples to train the discriminator network k epoch and apply this procedure d-steps.

2.2 Literature Review

Automated UI-testing speeds up the testing process. This way, test experts have more time for different software quality control activities. In turn, this saves time by avoiding repetitive and time-consuming tasks.

2.2.1 Large-scale Graph Data Processing For UI Testing

An example test flow in UI testing is discussed in [5]. Automated UI-testing can be used for testing e-commerce web functionalities. In our study, we investigate a methodology that can automate IU-testing by processing large-scale graph data, i.e. dynamic URLs.

AutoClicker [6] is an example study that fully automates UI tests on Android based smartphones. Interaction Flow Modeling Language [7] is a modeling language that manages user interaction and testing flows. It provides information about UI and is platform-independent. It is used in different approaches such as model-based test generation. Furthermore, by using the IFML and some well-defined rules, test case transformation can be applied. In our study, we are mainly focused on the pre-processing of dynamic URLs so that one can utilize this information in generating automated UI test scripts.

Clickstream data provide information about the sequence of URLs viewed by users while navigating a Web site. Similar to our work, Alan Montgomery and his colleagues investigated the categorization of the URLs that is available within the clickstream data [8]. They collected their data by using a tool called Jupiter Media Metrix which is also

known as Comscore Media Metrix (CMM). The main goal was finding information in the sequence of web viewings for predicting user's paths. However, in this study, we particularly focus on categorizing dynamic URLs, seen within the clickstream data, to identify the most frequently used functionalities to produce test scripts. In this manuscript, we focus on discussing a workflow that identifies the most frequently used functionalities by means of clustering.

Clickstream data is analyzed to understand the browsing behavior using a nested Hidden Markov Model in [9]. This model can classify visitor sessions into predefined categories [10]. In another study, researchers utilized logical sessions corresponding to user visits within a log file. Here, they classified the click-stream data by using k-NN [11] based classifier. To do this, they used a novel similarity metric, which is used to measure similarity for sequenced click-stream data [12]. Different from these work, in this study, we focus on unsupervised machine learning methods to identify hidden structures within the large-scale dynamic URLs.

Data provenance is often analyzed to understand the data lineage related to the user behaviour. Examples of such studies [13], [14], [15], [16] investigate the methodologies that extract meaningful information to analyze social provenance data and to detect misinformation on the social networks. Different from the previous work in data provenance field, in this study, we analyze click-stream data to understand the user browsing behaviour and to generate test scripts.

Considering e-commerce platforms, Wendy W. Moe has focused on user behavior patterns, purchasing products, and their characteristics when doing clustering [17]. In this study, we also focus on identifying user behavior patterns. However, we use the resulting patterns to create testing scripts for automated testing.

Changes in the UI is a challenge for automated UI testing [18]. A change in the UI may cause many changes in the generation of test scripts. In this study, dynamic URLs are clustered for a given website, so that they can be used in automated UI testing.

Inma Hernández's study [19], proposes a method that splits the URLs into tokens and then calculates statistical features to classify the web pages. Eda Baykan [20] uses all-gram method over tokenized URLs to obtain features by detecting sub-words. Then, they apply supervised learning algorithms for topic classification. Blanco [21] uses an approach, which is based on length-based minimum description for URLs and page content, to cluster web pages. This study ignores the terms that have small cardinality in URLs. In turn, they only use the terms, which can represent the clusters. Huaping Yuan uses Word2vec for obtaining the vector representations to classify URLs [22]. Bo Li's study also splits the URLs from non-alphanumeric characters and applies

Word2vec [23], to compute the vectors of the URLs. Our work differs from the previous work, as we mainly focus on identifying hidden structures within the space of dynamic URLs. We present a unique methodology that can pre-process dynamic URLs in such a way that can lead to high-quality clustering results.

2.2.2 Extracting Information from Large Scale Graph Data

To addition for these labeling URLs, automatic determination of scenarios, such as our motivation mentioned in the introduction, becomes more important day by day. There are different approaches in the literature for determining scenarios automatically. After this, a research group work on [24] automatically generates test cases and codes using pre-existing UML sequence diagrams. Thus, many operations can be performed in the user interface in large-scale desktop and web applications.

Considering that these processes come consecutively, in a subsequent study to find these scenarios, these many possible scenarios are created by using genetic algorithms [25]. In another study in [26], allows users and developers to automatically save the tests they want to be performed automatically for android applications. Unfortunately, these can be unaffordable.

From this point of view, the algorithm that use the basis of the GAN algorithm is selected and proceeded. [27]. The purpose of the algorithms and other deep learning algorithms selected here in the flow we propose is to produce synthetic flows that are not included in the training dataset and contain actions that the user can perform.

On the other hand, our previous studies are interested in whether topic sequences can be tested by grouping URLs with regexes, using the outputs here, and extracting the steps that the user can move forward [1]. In the continuation of these studies, methods that can be progressed without the need for user interaction are also presented [28].

Our previous work proposed a framework that generates unprecedented URL sequences from user click-stream navigation data using HMM and Viterbi algorithms. In our calculations to measure the success of the HMM and Viterbi algorithms in that study, we found similarities between the topic sequences produced using GAN and given as input and the URL sequences produced by the topic sequences [29]. We have seen that this calculation is correct but incomplete. In our thesis, we changed our success criteria to complete it.

The data we use in our system is the flow data that may occur on the website. Websites are kept graphically due to their structure. Motivated by the growing need for fast distributed processing of large-scale graphics such as website graphics and various

social networks, the researchers wanted to introduce lower limits to keep calculations distributed. Therefore, they set a lower bound to calculate the spanning tree (ST) value in the website graph. As a result of their work, they determined algorithms and lower limits for distributed computation of various graph problems [30]. In the proposed system, we use GPU-supported processors and a flow to work on multiple GPUs to provide training.

3
RESEARCH PROBLEMS

In this study, we propose an e-business workflow for automating user interface testing. We propose the architecture of an e-business workflow. We also realize the prototype implementation. The most crucial function of our business workflow is to determine the success or failure in creating the synthetic sequences. The research questions we are interested in are as follows:

1) How can such a business workflow be realized and learn from large-scale graph data?

2) What might be the deep learning algorithms in this proposed e-business workflow process? What should be the approach used for this learning algorithm to learn from the page in the optimum way?

3) How can the success and correct outcomes of the selected deep learning algorithms be observed?

We observe challenges that we have stated above and we also do our study in utilizing the dynamic URLs in UI testing: (1) The dynamic URLs do not have simple form that can be managed and tracked semantically in the SDLC by analysts and test engineers. (2) They have an untraceable dynamic form which is difficult to be used in test scripts, since they are often generated on-the-fly by user's interaction with the application. (3) Dynamic URLs should be used while writing test scripts for functional UI testing to create more comprehensive testing scenarios.

As we stated above, we propose a flow on generating artificial test scripts over the data obtained by crawling the structure of an e-commerce website. The proposed workflow creates synthetic sequences by getting the URL structure of the website. Furthermore, to facilitate testing of the proposed approach, we implement a prototype application. We test the prototype application via an e-commerce website and dataset collected from that website. The experimental study results are promising and show the usability of the proposed approach.

4
PROPOSED METHODOLOGY FOR EMBEDDING APPROACHES FOR USER BROWSING BEHAVIOUR

In overview, we advance the generation of test scripts through the proposed pipeline to expedite high-quality in model-based testing. We generate new test scripts based on the hotspot points, which are mined from large scale navigation URLs. We cluster the dynamic URLs and, as for clustering the dynamic URLs, we propose a data processing pipeline. To this end, we use a data pre-processing approach that is using URL splitting and word embedding methods.

4.1 Labeled URLs

One way to obtain labeled URLs is using clustering algorithms. In the following chapters labeled URLs and topics are used interchangeably. To cluster the dynamic URLs, features must be extracted from URL strings. To achieve this, we use word embedding algorithms. Word embedding algorithms are good for extracting meanings from words. They traverse sentences word by word and generate vectors for each word in the corpus.

To extract features from URLs, each URL is split into words, which are then used as input in the word embedding algorithm. Here, we use Word2vec as the embedding algorithm. After splitting the URLs and passing them to Word2vec, we obtain the word vectors. Then, we cluster the dynamic URLs by using the features obtained from the word embedding.

4.2 Usage on Labeled URLs

Test experts can use URLs and adds user sequences to generate tests. For example, users can navigate many different types of pages under the same e-commerce website such as product, brand, category, and so on. Figure 4.1 illustrates the model-based

testing graph for an example website. Here, a user navigates from different products and categories. Product pages are shown in gray and category pages are shown in green color. The starting point is shown in yellow. In this graph, it is hard to understand all the transactions. With our study, this whole graph can be simplified as in Figure 4.2, and is easy to understand many scenarios.

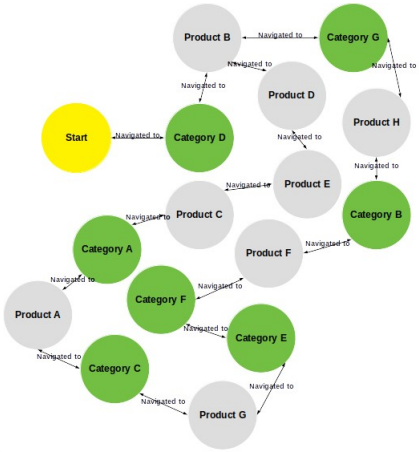


Figure 4.1 Not clustered graph of navigation

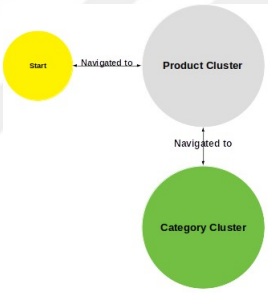
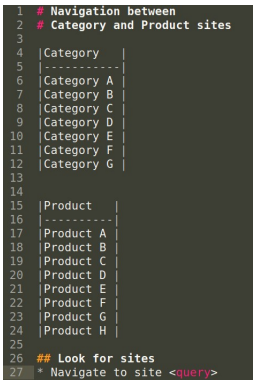


Figure 4.2 Clustered graph of navigation

As can be seen from the figures, by clustering, a simpler graph structure is obtained. The clustered graph allows testers to write more concise test scripts. Figure 4.3 shows an example BDD test script for the graph in Figure 4.1. After the clustering, this script can be shortened as in Figure 4.4.

We proposed a data processing workflow to cluster large-scale dynamic URLs, so that categories, i.e. functionalities, for each cluster can be identified for generating the User Interface test scripts.

This proposed methodology splits the URLs into words, generates word vectors by running Word2vec on the split URLs, computes weighted average word vectors for every URL (by multiplying each of them with a coefficient) to obtain URL vectors, then clusters these URL vectors. These stages are run repeatedly with different hyper-parameters to find the best clustering.



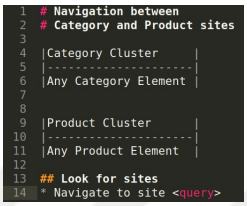
```
# Navigation between
# Category and Product sites

|Category   |
|-----------|
|Category A |
|Category B |
|Category C |
|Category D |
|Category E |
|Category F |
|Category G |


|Product    |
|-----------|
|Product A  |
|Product B  |
|Product C  |
|Product D  |
|Product E  |
|Product F  |
|Product G  |
|Product H  |

## Look for sites
* Navigate to site <query>
```

Figure 4.3 Markdown table for not clustered graph testing



```
# Navigation between
# Category and Product sites

|Category Cluster     |
|---------------------|
|Any Category Element |


|Product Cluster      |
|---------------------|
|Any Product Element  |

## Look for sites
* Navigate to site <query>
```

Figure 4.4 Markdown table for clustered graph testing

The workflow is implemented on Apache Spark, and tested on there different websites' dynamic URLs datasets. Besides, the clustering algorithms used as part of the pipeline are evaluated on our dataset from the perspective of scalability.

5
PROPOSED METODOLOGY FOR UI TEST SCRIPT GENERATION WORKFLOW

To address the research problem aforementioned, we propose a framework that can process large-scale browsing graph data and generate UI testing scripts. Figure 5.1 shows the proposed workflow. The proposed e-business workflow is designed to be a distributed system. Therefore, its components are implemented on distributed software development platforms. The proposed workflow consists of several components: The Crawling Module, The Browsing Graph Data Clustering Module, The Mapping Based Data Representation Module, The GAN Based Sequence Generator Module, The Filtering Module, and The Sequence2Script Module.

Each module has its own responsibilities. As we will mention in the flow while doing these, we aim to automate this flow and complete the end-to-end automated UI testing framework proposal. Furthermore, there is a transfer of outputs in the continuing step. This transfer continues in series and the order of the flow can be controlled. We explain these modules in detail below.

5.1 Crawling Module

This module takes the starting page of the e-commerce website to be tested as to its input and generates all possible URL sequences. The starting page of the website is the first page that users can interact.

The importance of this first page carries necessary information for our crawler. Because the crawler we have implemented will know this page as the start and will take its steps according to this page. To be more precise, the start page will come from the first step, while the other pages will only come from the start page or another self-referential parent page. Of course, this self-referential parent page will eventually be accessible from the home page. In addition to these, thanks to the start page, we will start to create the structure of the website.

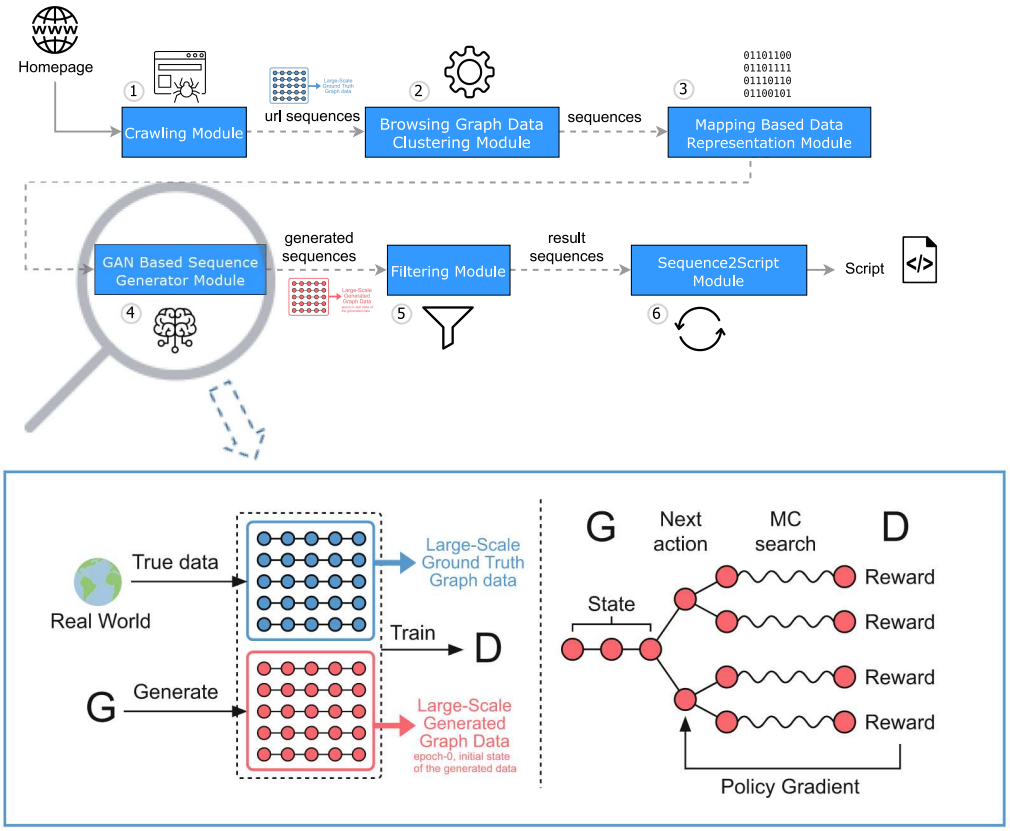


Figure 5.1 Overview of the proposed pipeline

The crawler module generates clickstream data sequences that represent the user's browsing behaviors on the website. We record each clickstream data sequence as graph data. Figure 5.2 depicts an example browsing graph data.

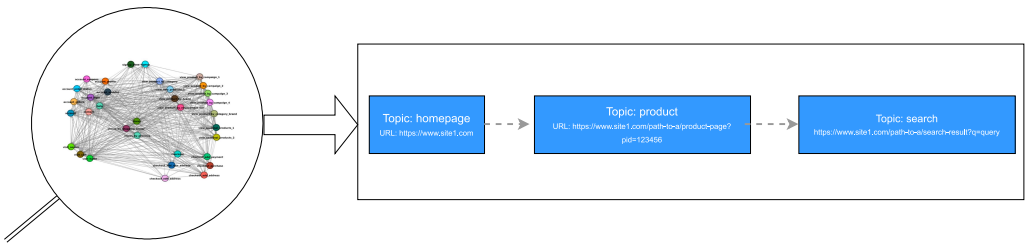


Figure 5.2 An example of the browsing data

The 5.2 on the left represents the large scale browsing graph data. The figure on the right represents the one particular browsing graph data that belongs to a user. Note that the user browsing graph data is referred as clickstream data sequence throughout the manuscript.

5.2 Browsing Graph Data Clustering Module

This module is responsible for processing the clickstream data-sequences data to map each URL to a related topic. In this module, we apply clustering algorithms on the clickstream data sequence graph data to categorize them and determine the appropriate topic label for each category. To do this, we use the word2vec embedding approach to represent each URL as encoded feature vectors. We cluster the URLs encountered in clickstream data sequences using different clustering algorithms such as partition-based clustering algorithms (e.g., k-means) and hierarchical clustering algorithms (e.g., bisecting k-means). We obtain the web site's URL tree structure by crawling the website.

After completing the clustering, The Browsing Graph Data Clustering Module also determines the label for each cluster. This module maps each URL within a cluster to the corresponding topic of that cluster. This way, we use the topics to represent the URLs. As an output, this module produces "topic sequences" that correspond to URL sequences. Table5.1 shows the example topic labels. The first column indicates the URL which the the user visits during browsing the website. The second column indicates the topic, which is associated to each URL.

Table 5.1 Sample dataset

Input Data: URL	Output Data: Label
https://www.site1.com/path-to-a/product-page?pid=123456	product
https://www.site1.com/path-to-a/search-result?q=query	search

5.3 Mapping Based Data Representation Module

The results we get after applying regular expressions to streams are sequences. In this module, we use a vocabulary-based mapping methodology to represent "topic sequences." Here, we create a vocabulary dictionary from all the unique keywords found in the sequences. By using this vocabulary dictionary, we map each keyword in sequence dataset to its corresponding numerical value. Figure 5.3. represents the way we represent the topic sequences with numerical values.

5.4 GAN Based Sequence Generator Module

In this module, we use a GAN-based deep learning algorithm to generate new data sequences. There are many different versions of GAN based deep learning algorithms in the literature. The SeqGan network proposed by Yu et al. inspires the method used in this study [4]. Figure 5.1 depicts our deep learning approach. GAN (Generative

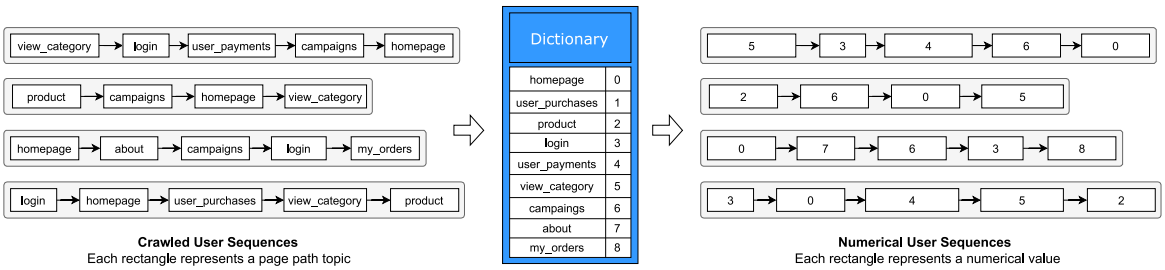


Figure 5.3 The mapping process used in the system

adversarial networks) [27] architectures consist of two neural networks. These two nerves are called heavy Generator and Discriminator. When GAN is given a training set, it learns to create new data with the same statistics as the training set. Here, the generator tries to trick the discriminator into producing new artifacts. And the discriminator will gradually improve the distinction.

We use the ground truth data sequences created in the previous steps by the clustering module to train the model. At each iteration of the GAN-based neural network, we also produce generated data sequences to trick the discriminator into creating new data sequences.

As an output of this module, the new artificial data sequences (i.e., topic sequences) are produced.

To improve our proposed workflow, we have tried to optimize the parameters that we have used in our deep learning approach. Of course, hyperparameter optimization is required in order to improve the efficiency and evaluation of the produced topic sequences. Moreover, the parameters of deep learning algorithms must be optimized to be used with the best efficiency.

Here we want to optimize three parameters for each iteration. The flow algorithm in the objective function can be followed from Table 5.2.

5.5 Filtering Module

In this module, we implement two important features in our workflow for data sequence generation. These features are:

1) eliminate the sequences in the dataset produced by the proposed workflow and,

2) check if a page path-topic sequence is valid.

In this context, valid sequences are those sequences that can be executable using

Algorithm 1 Objective Function

Input:EMB_DIM, HID_DIM, GEN_SEQ_NUM
Output: Perplexity
Purpose: Ensure minimized perplexity score

1. $generated_sequences \leftarrow seqGAN(emb_dim, hid_dim, gen_seq_num)$
2. $training_sequences \leftarrow fileRead(train_sequence_directory)$
3. $encoded_training_sequences \leftarrow []$
4. $encoded_generated_sequences \leftarrow []$
5. **for** $train_sequence\ in\ training_sequences$ **do**
6. $\quad encoded_training_sequences.append(encode(train_sequence))$
7. **end for**
8. **for** $generated_sequence\ in\ training_sequences$ **do**
9. $\quad encoded_generated_sequences.append(encode(generated_sequence))$
10. **end for**
11. $trie \leftarrow Trie(encoded_training_sequences)$
12. $trie.fit()$
13. $encoded_unique_sequences, encoded_duplicate_sequences \leftarrow trie.check(encoded_generated_sequences)$
14. $sequenceMap \leftarrow SequenceMap(encoded_training_sequences)$
15. $sequenceMap.fit()$
16. $encoded_valid_sequences, encoded_invalid_sequences \leftarrow sequenceMap.check(encoded_unique_sequences)$
17. $perplexity \leftarrow calculatePerplexity(encoded_training_sequences, encoded_valid_sequences)$

Table 5.2 Objective Function

a test executor. Figure 5.4 illustrates the construction of Trie and HashMap data structures on a example training sequences 5.4. We use Trie data structure to eliminate duplicated sequences generated by the model. At the end of the filtering module, the sequences that are not eliminated are used to generate test scripts.

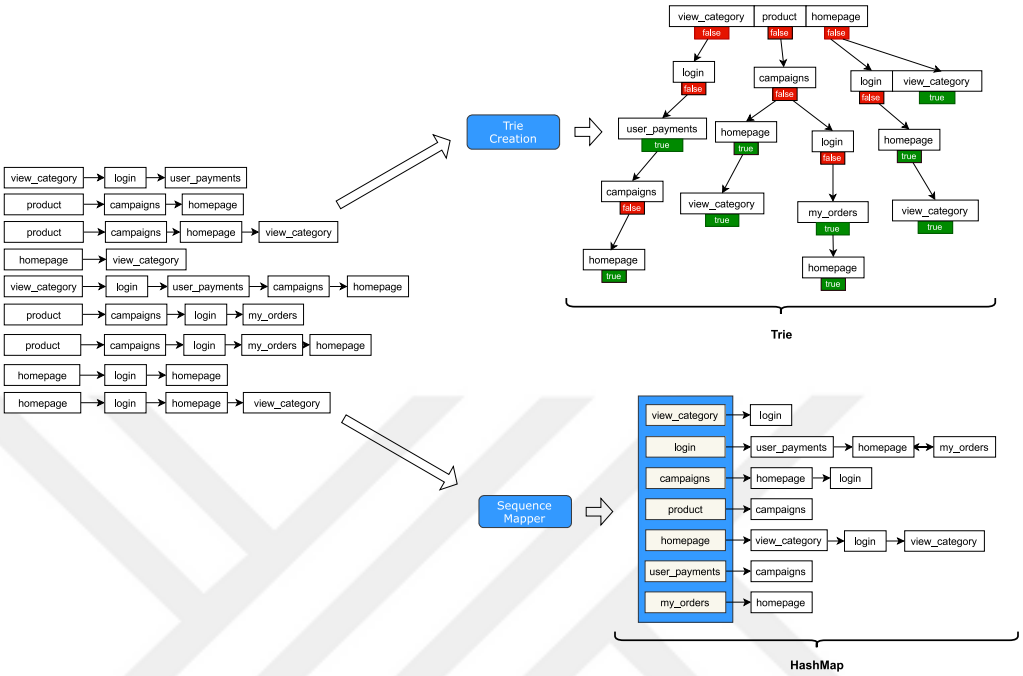


Figure 5.4 Building Trie and HashMap on example training sequences

5.6 Sequence2Script Module

The purpose of our last module in the system is to generate executable test scripts using sequences from the previous module. Up to now, we create data sequences with a deep learning approach using GAN architecture for representing possible website topics that can be sequentially visited.

To create executable test scripts, we need website URLs and the identifiers of elements showing which element interacts with the generated sequences on those websites. This is a concept module for creating test scripts from giving data sequence.

6
PROTOTYPE IMPLEMENTATION

To facilitate testing of the proposed e-business workflow, we provide a prototype implementation. We implemented a crawler to collect data on the website. The Scrapy library in Python was used to implement a crawler. Here, for tracking the pages where the crawler goes, we used the topics of the website whose following page structure we know and extracted regular expressions [3].

After the URL sequence data was obtained, the URL sequences were clustered with the k-means and bisecting k-means algorithms in Python. Then the v-measure scores were calculated. We chose the best performing clustering methodology based on the v-measure values. Here, we used Apache Spark ML library to implement the clustering module [3].

We used Sagemaker on the AWS machine to train the deep learning model and determine its results. Here, KernelGateway-ml.g4dn.xlarge was selected for the kernel, and the training was completed with GPU support.

We created a Dom tree while generating scripts. Then, we produced scenarios suitable for the BDD approach by using Cucumber. In this way, we created a UI test script that can be used for the relevant flow, and as a result, we completed the process of obtaining the test scenarios. During the test script production we perform here, these codes are generated with negligible periods.

6.1 Datasets

In this study, we used data from 3 different websites, and we showed the sample working flow of our system using this data. The first of these datasets was an e-commerce site selling mother and baby products, the other two datasets were loadium and testinium datasets belonging to Saha Bilgi Teknolojileri Egitim Danismanlık San. Tic. A.S. R&D Center (Testinium) company.

6.2 Test Design

In order to implement the proposed workflow, it was necessary to obtain the structure of the website of interest. By using crawler, we both obtained the structure of the website and extracted the user flows that could be an example. In this way, the process is also done in order to generate the flows that users can prefer. In the crawler implementation, the scrapy library was used, and the structure of the website and the URL sequences were found.

After the URL streams were found, the obtained data was preprocessed and used in different steps for easier understanding. These preprocess and steps were briefly;

1. Determining the parts of the data that were advanced by click,
2. Determination of regexes,
3. Application of regex rules for user action sequence,
4. Generating new topic sequences by giving the obtained topic sequences to the GAN algorithm,
5. Calculating the similarities of topic sequences produced with GAN and topic sequences in user action sequences,
6. Preparation of test scripts that could be used in tools such as Selenium and Cucumber as it progressed.

Deep learning was run on topic sequences taken as input with seqGAN, which was a GAN type. As it is known, GAN flow contains two different networks in itself. These are the discriminator and generator. It is used to create LSTM and CNN discriminator and generator networks in seqGan, respectively. Definitely, due to the nature of the GAN, the generator produces streams that will be very close to reality while the discriminator tries to classify these generated streams.

We tried to optimize 3 of the parameters used in the discriminator and generator, which we used here in our study. These were briefly the number of sequences to be produced, the embedding dimension used in LSTM and CNN, and the hidden dimension parameters. In hyperparameter optimization, 100 trials were made for each dataset, and the emb_dim and hidden_dim dimension parameters used in the generator and the number of sequences to be produced were tried to be optimized. Although the trial spaces were the same for each site, the resulting values differed in the sites.

The perplexity score was selected in the objective function to be used for optimization. Streams that were optimized and gave the lowest perplexity score passed to the control section as optimized sequences. After running Trie and SequenceMap, both perplexity and bleu score values were calculated as 4-grams.

7
EVALUATION

To facilitate testing of the prototype implementation and show the usability of the proposed workflow, we conducted an experimental study. We measure the proposed system performance with Perplexity and BLEU evaluation metrics. We describe the details of the evaluation study below.

7.1 Dataset

We collect the website's URL tree structure using a crawler. Then, we created data sequences using this tree structure as the ground truth data. Note that browsing data sequences can include a high number of steps. For example, quintillions of possible data sequences can be a matter of question for a 6-step browsing data sequence. An increase in the number of elements in a data sequence also increases, the total number of possible data sequence. To facilitate testing of the proposed approach, in this study, we only focused on up to 7-node data sequences. Since we divided sequences including at least one duplicated token into subsequences, we generally obtain smaller sequences. Therefore we accounted up to 7-node data sequences. Figure 5.2 shows an example browsing graph data.

Note that, by crawling the website does not able to capture all possible paths in the website's URL tree structure. Because the e-commerce websites limit crawlers' activities within the sites for various reasons such as they don't allow robots to crawl the websites for security purposes. Hence, we only have a partial picture of the URL tree structure. Based on the obtained URL structure, we generated the possible 1-node, 2-node, ... n-node data sequences as the ground truth data. By utilizing the proposed workflow, we tried to generate new data sequences that did not exists in the ground truth dataset and that could be used to create new UI testing scripts.

7.2 Experimental Study Design

To understand the usability of the proposed workflow, we compared the final output generated data-sequences against the training dataset (i.e., ground truth browsing data-sequences). We investigated whether the proposed workflow could find new data sequences that the system has not seen before. Then, the new dataseqeunces should represent the same browsing behavior that one would encounter in the initial input of the ground truth browsing data sequence dataset.

We run the whole workflow and generated new topic sequences by utilizing the seqGAN algorithm. The proposed workflow used the Trie and HashMap data structures to make generated sequences valid and non-duplicated. We compared the final generated data sequence against the initial dataset by using perplexity and BLUE metrics. The tests were conducted on Jupyter in AWS Sagemaker employed a Tesla T4 GPU.

Evaluation Metrics There are so many metrics to compute similarity, difference, distance, and so on. We chose the two metrics to evaluate our generated graph data sequences: Bleu score metric and Perplexity metric.

By using Blue score we evaluate our generated sequences over training sequences. Results close to zero indicate that it is not good enough for translations in data sequences and results close to 1 mean better translations [31]. The Bleu Score metric is be calculated with the following equation:

$$BLEU_c = \prod_{i=1}^{N} P(i)^{w_i} \tag{7.1}$$

where $\sum_{i=1}^{N} w_i = 1$ and c is the n-gram order

We also use perplexity to evaluate the generated sequences over training sequences. The perplexity metric does not have a range like the blue score. It is particularly used for evaluating language models [32]. We took inverse probability to calculate this metric. Therefore, a lower perplexity score meant that we produce better data sequences. The Perplexity could be calculated with the following formula:

$$\text{Perplexity} = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P\left(w_i \mid w_1 w_2 \ldots w_{i-1}\right)}} \tag{7.2}$$

7.3 Evaluation Results

The results we obtained have been arranged in order to find answers to the questions of the system we mentioned above in the research motivation and the research problems. We calculate the BLEU and Perplexity scores for a random number of data sequences generated as the output of the whole workflow. After these calculations, we compute an average of Perplexity and BLUE metrics separately for the sequences to obtain the overall system scores.

After the proposed system was realized as a prototype, trials were carried out. Here, EMB_DIM and HIDDEN_DIM parameters used in datasets were between 2^3 and 2^{15}, and the number of sequences to be produced was between 10k and 1m.

The first selected dataset is an e-commerce site that generally markets mother and baby related products. Table 7.2 shows the best parameters produced for this dataset and the table 7.1 indicates for perplexity and bleu score values measured in 4-grams can be followed.

Table 7.1 Best scores for Mother&Baby Products site

	1-gram	2-gram	3-gram	4-gram
Bleu Score	0.966	0.857	0.776	0.711
Perplexity	6.772	7.629	5.994	4.790

Table 7.2 Best parameters for Mother&Baby Products site

	EMB_DIM	HIDDEN_DIM	Generated Sequence Num
Optimized Parameter	64	4096	165616

The second data used was the dataset of the loadium site. Here, the operations applied in the first dataset were applied on the dataset, and then the results were run. The point that draws attention here is that the results in the table 7.3 are not similar to the results in the mother and baby products site. This is because the training dataset used is diversed and does not contain much data. The best parameters of the relevant site can be seen in the table 7.3.

Table 7.3 Best scores for Loadium site

	1-gram	2-gram	3-gram	4-gram
Bleu Score	0.882	0.728	0.59	0.452
Perplexity	8.437	8.672	6.493	5.042

The last dataset we used in our research was the testinium dataset. In this dataset, we obtained similar to the ones in the first dataset. We verified and confirmed that data

Table 7.4 Best parameters for Loadium site

	EMB_DIM	HIDDEN_DIM	Generated Sequence Num
Optimized Parameter	2048	64	132075

Table 7.5 Best scores for Testinium site

	1-gram	2-gram	3-gram	4-gram
Bleu Score	0.979	0.859	0.777	0.701
Perplexity	6.256	7.332	5.988	4.892

Table 7.6 Best Parameters for Testinium Site

	EMB_DIM	HIDDEN_DIM	Generated Sequence Num
Optimized Parameter	32768	512	304761.0

must contain several and multiple streams in order to achieve these results. The best parameters and evaluation results of the relevant site can be seen in the table 7.5 7.6.

The results show that the proposed workflow can generate new data sequences successfully. Furthermore, these newly generated data sequences could be imported to automated UI script executor tools such as Selenium and Cucumber. The workflow we have suggested could ease our study to automate with these tools.

7.4 Threats to Validity of Results

The results we obtained in this study depend on the website URL structure of the websites we have carried out. However, the proposed end to end automated ui testing framework's workflow that we have used constitutes a step-by-step mechanism independent of data. Moreover, we have conducted our experiments on 3 different websites to make sure that our introduced workflow is accurate. The frameworks flow is also detailed and described above. The proposed workflow can also be applied to different datasets.

8
CONCLUSION

In this study, we proposed an end-to-end framework that can generate automatic UI test scripts on websites that have not been used before or are less used. We are testing this recommended framework on 3 different sites. We believe that the results will shed light on future studies.

The designed framework create unprecedented artificial URL sequences by using seqGAN from GAN variations of deep learning methods. The proposed workflow utilizes a GAN-based deep learning approach, which allows the system to generate new clickstream-based data sequences after training on the large-scale graph data.

In our previous study in the thesis, we stated that we want to work on different datasets in the [29] future works section. In this thesis, we also achieved this goal by testing on 3 different datasets in test script generation.

In the proposed workflow, to represent the graph data, we use a vocabulary-based mapping approach. We reckon that embedding approaches such as word2vec and node2vec can also be used and integrated with the proposed workflow.

To facilitate testing of the system, we implement a prototype application. We evaluate the prototype application via an experimental study. The results of the empirical research results indicate that the proposed approach can generate new synthetic data sequences. In turn, these data sequences can be used to create UI test scripts automatically.

REFERENCES

[1] M. Oz, C. Kaya, E. Olmezogullari, M. S. Aktas, "On the use of generative deep learning approaches for generating hidden test scripts," *International Journal of Software Engineering and Knowledge Engineering*, vol. 31, no. 10, pp. 1447–1468, 2021.

[2] P. Akpinar, M. S. Aktas, A. B. Keles, Y. Balaman, Z. O. Guler, O. Kalipsiz, "Web application testing with model based testing method: Case study," in *2020 International Conference on Electrical, Communication, and Computer Engineering (ICECCE)*, IEEE, 2020, pp. 1–6.

[3] Y. Uygun, R. F. Oguz, E. Olmezogullari, M. S. Aktas, "On the large-scale graph data processing for user interface testing in big data science projects," in *2020 IEEE International Conference on Big Data (Big Data)*, IEEE, 2020, pp. 2049–2056.

[4] L. Yu, W. Zhang, J. Wang, Y. Y. SeqGAN, "Sequence generative adversarial nets with policy gradient. arxiv e-prints, page," *arXiv preprint arXiv:1609.05473*, 2016.

[5] S. Al-Zain, D. Eleyan, J. Garfield, "Automated user interface testing for web applications and testcomplete," in *Proceedings of the CUBE International Information Technology Conference*, 2012, pp. 350–354.

[6] T. Ki, A. Simeonov, C. M. Park, K. Dantu, S. Y. Ko, L. Ziarek, "Fully automated ui testing system for large-scale android apps using multiple devices," in *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, 2017, pp. 185–185.

[7] N. Yousaf, F. Azam, W. H. Butt, M. W. Anwar, M. Rashid, "Automated model-based test case generation for web user interfaces (wui) from interaction flow modeling language (ifml) models," *IEEE Access*, vol. 7, pp. 67 331–67 354, 2019.

[8] A. L. Montgomery, S. Li, K. Srinivasan, J. C. Liechty, "Modeling online browsing and path analysis using clickstream data," *Marketing science*, vol. 23, no. 4, pp. 579–595, 2004.

[9] H. Kawazu, F. Toriumi, M. Takano, K. Wada, I. Fukuda, "Analytical method of web user behavior using hidden markov model," in *2016 IEEE International Conference on Big Data (Big Data)*, IEEE, 2016, pp. 2518–2524.

[10] S. L. Scott, I.-H. Hann, "A nested hidden markov model for internet browsing behavior," *Marshall School of Business*, pp. 1–26, 2006.

[11] D. Pai, A. Sharang, M. M. Yadagiri, S. Agrawal, "Modelling visit similarity using click-stream data: A supervised approach," in *International Conference on Web Information Systems Engineering*, Springer, 2014, pp. 135–145.

[12] O. Kashefi, B. Minaei, “A novel similarity measure for sequence data,” *Journal of Information Processing Systems*, vol. 7, no. 3, 2011.

[13] Y. Tas, M. J. Baeth, M. S. Aktas, “An approach to standalone provenance systems for big social provenance data,” in *2016 12th International Conference on Semantics, Knowledge and Grids (SKG)*, IEEE, 2016, pp. 9–16.

[14] M. J. Baeth, M. S. Aktas, “Detecting misinformation in social networks using provenance data,” in *2017 13th International Conference on Semantics, Knowledge and Grids (SKG)*, IEEE, 2017, pp. 85–89.

[15] ——, “An approach to custom privacy policy violation detection problems using big social provenance data,” *Concurrency and Computation: Practice and Experience*, vol. 30, no. 21, e4690, 2018.

[16] M. Riveni, T.-D. Nguyen, M. S. Aktas, S. Dustdar, “Application of provenance in social computing: A case study,” *Concurrency and computation: practice and experience*, vol. 31, no. 3, e4894, 2019.

[17] W. W. Moe, “Buying, searching, or browsing: Differentiating between online shoppers using in-store navigational clickstream,” *Journal of consumer psychology*, vol. 13, no. 1-2, pp. 29–39, 2003.

[18] R. Coppola, E. Raffero, M. Torchiano, “Automated mobile ui test fragility: An exploratory assessment study on android,” in *Proceedings of the 2nd International Workshop on User Interface Test Automation*, 2016, pp. 11–20.

[19] I. Hernández, C. R. Rivero, D. Ruiz, R. Corchuelo, “A statistical approach to url-based web page clustering,” in *Proceedings of the 21st International Conference on World Wide Web*, 2012, pp. 525–526.

[20] E. Baykan, M. Henzinger, L. Marian, I. Weber, “Purely url-based topic classification,” in *Proceedings of the 18th international conference on World wide web*, 2009, pp. 1109–1110.

[21] L. Blanco, N. Dalvi, A. Machanavajjhala, “Highly efficient algorithms for structural clustering of large websites,” in *Proceedings of the 20th international conference on World wide web*, 2011, pp. 437–446.

[22] H. Yuan, Z. Yang, X. Chen, Y. Li, W. Liu, “Url2vec: Url modeling with character embeddings for fast and accurate phishing website detection,” in *2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom)*, IEEE, 2018, pp. 265–272.

[23] B. Li, G. Yuan, L. Shen, R. Zhang, Y. Yao, “Incorporating url embedding into ensemble clustering to detect web anomalies,” *Future Generation Computer Systems*, vol. 96, pp. 176–184, 2019.

[24] R. Anbunathan, A. Basu, “Automation framework for test script generation for android mobile,” in *2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, IEEE, 2017, pp. 1914–1918.

[25] A. Rauf, S. Anwar, M. A. Jaffer, A. A. Shahid, “Automated gui test coverage analysis using ga,” in *2010 Seventh International Conference on Information Technology: New Generations*, IEEE, 2010, pp. 1057–1062.

[26] J. Kaasila, D. Ferreira, V. Kostakos, T. Ojala, “Testdroid: Automated remote ui testing on android,” in *Proceedings of the 11th International Conference on Mobile and Ubiquitous Multimedia*, 2012, pp. 1–4.

[27] I. Goodfellow *et al.*, “Generative adversarial nets,” *Advances in neural information processing systems*, vol. 27, 2014.

[28] R. F. Oguz, M. Oz, E. Olmezogullari, M. S. Aktas, “Extracting information from large scale graph data: Case study on automated ui testing,” in *Euro-Par 2021*, 2021.

[29] I. Erdem, R. F. Oguz, E. Olmezogullari, M. S. Aktas, “Test script generation based on hidden markov models learning from user browsing behaviors,” in *2021 IEEE International Conference on Big Data (Big Data)*, 2021, pp. 2998–3005. DOI: 10.1109/BigData52589.2021.9671312.

[30] H. Klauck, D. Nanongkai, G. Pandurangan, P. Robinson, “Distributed computation of large-scale graph problems,” in *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*, SIAM, 2014, pp. 391–410.

[31] K. Papineni, S. Roukos, T. Ward, W.-J. Zhu, “Bleu: A method for automatic evaluation of machine translation,” in *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, 2002, pp. 311–318.

[32] F. Jelinek, R. L. Mercer, L. R. Bahl, J. K. Baker, “Perplexity—a measure of the difficulty of speech recognition tasks,” *The Journal of the Acoustical Society of America*, vol. 62, no. S1, S63–S63, 1977.

PUBLICATIONS FROM THE THESIS

Journal Papers

1. Oguz, Ramazan Faruk, Izzettin Erdem, Olmezogullari, E. and Aktas, M.S., 2022. End-to-End Automated UI Testing Framework for Web Sites with Intensive User-System Interactions. International Journal of Software Engineering and Knowledge Engineering (Submitted).

Conference Papers

1. Oguz, Ramazan Faruk, Oz Mert, Olmezogullari, E. and Aktas, Mehmet S., 2021 Extracting Information from Large Scale Graph Data: Case Study on Automated UI Testing, Euro-Par 2021.

2. Erdem, I., Oguz, R.F., Olmezogullari, E. and Aktas, M.S., 2021, December. Test Script Generation Based on Hidden Markov Models Learning From User Browsing Behaviors. In 2021 IEEE International Conference on Big Data (Big Data) (pp. 2998-3005). IEEE.

3. Uygun, Yasin, Oguz, Ramazan Faruk, Olmezogullari, Erdi, and Aktas, Mehmet S. On the large-scale graph data processing for user interface testing in big data science projects. In 2020 IEEE International Conference on Big Data (Big Data), pages 2049–2056. IEEE, 2020.