

VILNIUS GEDIMINAS TECHNICAL UNIVERSITY

Kiril GRIAZEV

INTERNET WEB PAGE CONTENT BLOCK DATASET AND SOLUTIONS FOR ITS DATA LABELLING SIMPLIFICATION

DOCTORAL DISSERTATION

TECHNOLOGICAL SCIENCES,
INFORMATICS ENGINEERING (T 007)

Vilnius, 2024

The doctoral dissertation was prepared at Vilnius Gediminas Technical University in 2017–2024.

Supervisor

Prof. Dr Simona RAMANAUSKAITĖ (Vilnius Gediminas Technical University, Informatics Engineering – T 007).

The Dissertation Defence Council of the Scientific Field of Informatics Engineering of Vilnius Gediminas Technical University:

Chairman

Prof. Dr Dalius MAŽEIKA (Vilnius Gediminas Technical University, Informatics Engineering – T 007).

Members:

Dr Jolita BERNATAVIČIENĖ (Vilnius University, Informatics – N 009),

Dr Robertas DAMAŠEVIČIUS (Kaunas University of Technology, Informatics Engineering – T 007),

Prof. Dr Arnas KAČENIAUSKAS (Vilnius Gediminas Technical University, Informatics Engineering – T 007),

Dr Kirsto KARJUST (Tallinn University of Technology, Estonia, Informatics Engineering – T 007).

The dissertation will be defended at the public meeting of the Dissertation Defence Council of the Scientific Field of Informatics Engineering in the SRA-I Meeting Hall of Vilnius Gediminas Technical University at **14 a.m. on 12 June 2024**.

Address: Saulėtekio al. 11, LT-10223 Vilnius, Lithuania.

Tel.: +370 5 274 4956; fax +370 5 270 0112; email: doktor@vilniustech.lt

A notification on the intended defence of the dissertation was sent on 10 May 2024.

A copy of the doctoral dissertation is available for review at the Vilnius Gediminas Technical University repository <https://etalpykla.vilniustech.lt>, at the library of Vilnius Gediminas Technical University (Saulėtekio al. 14, LT-10223 Vilnius, Lithuania) and the library of Kaunas University of Technology (K. Donelaičio g. 20, LT-44239 Kaunas, Lithuania).

Vilnius Gediminas Technical University book No. 2024-032-M

<https://doi.org/10.20334/2024-032-M>

© Vilnius Gediminas Technical University, 2024

© Kiril Griahev, 2024

kiril.griahev@vilniustech.lt

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

Kiril GRIAZEV

INTERNETO TINKLALAPIŲ TURINIO BLOKŲ
DUOMENŲ RINKINYS IR JO DUOMENŲ
ŽYMĖJIMĄ SUPAPRASTINANTYS
SPRENDIMAI

DAKTARO DISERTACIJA

TECHNOLOGIJOS MOKSLAI,
INFORMATIKOS INŽINERIJA (T 007)

Vilnius, 2024

Disertacija rengta 2017–2024 metais Vilniaus Gedimino technikos universitete.

Vadovė

prof. dr. Simona RAMANAUSKAITĖ (Vilniaus Gedimino technikos universitetas, Informatikos inžinerija – T 007).

Vilniaus Gedimino technikos universiteto Informatikos inžinerijos mokslo krypties disertacijos gynimo taryba:

Pirmininkas

prof. dr. Dalius MAŽEIKAS (Vilniaus Gedimino technikos universitetas, Informatikos inžinerija – T 007).

Nariai:

dr. Jolita BERNATAVIČIENĖ (Vilniaus universitetas, Informatika – N 009),
dr. Robertas DAMAŠEVIČIUS (Kauno technologijos universitetas, Informatikos inžinerija – T 007),
prof. dr. Arnas KAČENIAUSKAS (Vilniaus Gedimino technikos universitetas, Informatikos inžinerija – T 007),
dr. Kristo KARJUST (Talino technikos universitetas, Estija, Informatikos inžinerija – T 007).

Disertacija bus ginama viešame Informatikos inžinerijos mokslo krypties disertacijos gynimo tarybos posėdyje **2024 m. birželio 12 d. 14 val.** Vilniaus Gedimino technikos universiteto SRA-I posėdžių salėje.

Adresas: Saulėtekio al. 11, LT-10223 Vilnius, Lietuva.

Tel. (8 5) 274 4956; faksas (8 5) 270 0112; el. paštas doktor@vilniustech.lt

Pranešimai apie numatomą ginti disertaciją išsiųsti 2024 m. gegužės 10 d.

Disertaciją galima peržiūrėti Vilniaus Gedimino technikos universiteto talpykloje <https://etalpykla.vilniustech.lt>, Vilniaus Gedimino technikos universiteto (Saulėtekio al. 14, LT-10223 Vilnius, Lietuva) ir Kauno technologijos universiteto (K. Donelaičio g. 20, LT-44239 Kaunas, Lietuva) bibliotekose.

Abstract

The dissertation explores the intricacies of identifying, extracting, and documenting content blocks in internet web pages. The research object is the methodologies for these processes to improve the computer perception of online web page data. The primary goal is to conduct an in-depth analysis of datasets containing web page content blocks to enhance their granularity and minimise the volume of blocks requiring manual labelling.

The dissertation undertakes several essential tasks: (1) conducting a systematic analysis of the latest research in the field of data extraction from internet web pages; (2) developing a structured dataset for web pages that accommodates a variety of features for different content blocks and is compatible with various data extraction methods; (3) creating a solution for partly automated content block labelling in web pages, which establishes relationships between content blocks and groups them, thereby reducing the need for manual review; (4) evaluating the effectiveness of this developed dataset and labelling solution in identifying, grouping, and establishing relationships between web page content blocks.

The dissertation comprises four parts: an introduction, four main chapters, conclusions, references, and appendices. The introduction presents the research problem, significance, objectives, methodology, novelty, practical implications, defended statements, lists of the author's conference presentations and outlines the dissertation's structure.

The first chapter focuses on Web Mining and examines the challenges and evolution of data extraction and classification techniques.

The second chapter explores methods to determine HTML block similarity, considering data and structure.

The third chapter details creating a dataset for improved data extraction, highlighting the need for diverse information about block types, features, and structures.

The fourth chapter presents advanced methods for identifying HTML content blocks and enhancing content extraction accuracy and efficiency.

Several articles were published on the topic discussed in the dissertation: two in publications of the main list of *Clarivate Analytics Web of Science* and two in the publications of scientific conference proceedings. Research results were presented at three international conferences: *6th Workshop on Advances in Information, Electronic and Electrical Engineering* (AIEEE), 2018, Vilnius, Lithuania; *Open Conference of Electrical, Electronic and Information Sciences* (eStream), 2018, Vilnius, Lithuania; *International Conference on Science & Technology* (STRA), 2023, Prague, Czech Republic.

Reziumė

Disertacijoje nagrinėjami interneto svetainių turinio blokų identifikavimo, išgavimo ir dokumentavimo iššūkiai. Internetinių tinklalapių turinio blokai, jų išskyrimo, susiejimo ir dokumentavimo duomenų rinkinyje metodai. Pagrindinis tikslas yra atlikti išsamų duomenų rinkinių, kuriuose yra svetainių turinio blokai, analizę, siekiant padidinti jų detalumą ir sumažinti blokų, kuriems reikalingas rankinis žymėjimas, kiekį.

Disertacijoje nagrinėjama keletas svarbių užduočių: 1) sisteminė naujausių tyrimų interneto svetainių duomenų išgavimo srityje analizė; 2) struktūrizuoto svetainės skirtų duomenų rinkinio, kuriam būdingos įvairios turinio blokų savybės ir suderinimas su įvairiais duomenų išgavimo metodais, sukūrimas; 3) iš dalies automatizuoto turinio blokų žymėjimo svetainėse sprendimo, kuris nustato ryšius tarp turinio blokų ir juos grupuoja, taip mažindamas rankinio peržiūrėjimo poreikį, sukūrimas; 4) sukurtų duomenų rinkinio ir žymėjimo sprendimo efektyvumo vertinimas, identifikuojant, grupuojant ir nustatant ryšius tarp svetainės turinio blokų.

Disertacija susideda iš keturių dalių: įvado, keturių skyrių, išvadų, šaltinių ir priedų. Įvade pristatoma tyrimo problema, jo svarba, tikslai, metodika, naujumas, praktinė nauda, ginamos tezės, autoriaus konferencijų pranešimai ir apžvelgiama disertacijos struktūra.

Pirmajame skyriuje nagrinėjamas interneto duomenų gavybos (angl. *Web Mining*) klausimas, išryškinami duomenų išgavimo ir klasifikavimo technikų iššūkiai bei evoliucija.

Antrajame skyriuje tyrinėjami HTML blokų panašumo nustatymo metodai, atsižvelgiant į duomenis ir struktūrą.

Trečiajame skyriuje išsamiai aprašomas duomenų rinkinio sukūrimas, pritaikytas duomenų išgavimui, išryškinant įvairių blokų tipų, savybių ir struktūrų informacijos įvairovę.

Ketvirtajame skyriuje pristatomi pažangūs metodai HTML turinio blokams identifikuoti ir turinio išgavimo tikslumui bei efektyvumui didinti.

Publikuoti straipsniai disertacijos tema: du pagrindinio *Clarivate Analytics Web of Science* sąrašo leidiniuose ir du mokslinių konferencijų leidiniuose. Tyrimo rezultatai buvo pristatyti trijose tarptautinėse konferencijose: *6th workshop on Advances in Information, Electronic and Electrical Engineering* (AIEEE), 2018 m., Vilnius, Lietuva; *Open Conference of Electrical, Electronic and Information Sciences (eStream)*, 2018 m., Vilnius, Lietuva; *International Conference on Science & Technology, STRA*, 2023 m., Praha, Čekija.

Notations

Abbreviations

AI – Artificial Intelligence: technology simulating human cognitive functions (liet. *dirbtinis intelektas: technologija, simuliuojanti žmogaus kognityvines funkcijas*).

API – Application Programming Interface: a set of rules for software interaction (liet. *programų programavimo sąsaja: taisyklių rinkinys programinės įrangos sąveikai*).

CGI – Common Gateway Interface (CGI): an interface specification that enables web servers to execute an external program to process HTTP/S user requests (liet. *CGI yra sąsajos specifikacija, kuri leidžia interneto serveriams naudoti išorines programas, kad būtų apdorotos HTTP/S užklausos*).

CSS – Cascading Style Sheets: a style sheet language used for describing the presentation and formatting of documents written in HTML or XML (liet. *kaskadinės stilių lentelės: stilių aprašymo kalba, vartojama dokumentų, parašytų HTML ar XML kalbomis, pateikimui ir formatavimui aprašyti*).

DOM – Document Object Model: a programming interface for web documents, representing the page structure (liet. *dokumento objektų modelis: programavimo sąsaja interneto dokumentams, atvaizduojanti puslapio struktūrą*).

FN – False Negative: in statistical classification, this is used to describe outcomes where the model incorrectly predicts the negative class for a positive instance (liet. *neteisingai neigiamas: statistinėje klasifikacijoje tai taikoma atvejams apibūdinti, kai modelis neteisingai prognozuoja neigiamą klasę teigiamam atvejui*).

FP – False Positive: in statistical classification, this denotes outcomes where the model incorrectly predicts the positive class for a negative instance (liet. *neteisingai teigiamas: statistinėje klasifikacijoje tai reiškia atvejus, kai modelis neteisingai prognozuoja teigiamą klasę neigiamam atvejui*).

HTML – Hypertext Markup Language: language for structuring web pages (liet. *hiperteksto žymėjimo kalba: kalba interneto puslapiams struktūrinti*).

HTML5 – Hypertext Markup Language version 5: the fifth and current major version of HTML, offering new features for presenting content on the World Wide Web (liet. *hiperteksto žymėjimo kalba, 5 versija: penktoji ir šiuo metu pagrindinė HTML versija, siūlanti naujas funkcijas turiniui pateikti pasauliniame žiniatinklyje*).

LCS – Longest Common Sequence: a method used in computer science to identify the longest sequence of elements that are common to two or more sequences (liet. *ilgiausia bendra seka – tai metodas, taikomas kompiuterių moksle, norint nustatyti ilgiausią elementų seką, kuri yra bendra dviem ar daugiau sekoms*).

MLOSS – Machine Learning Open Source Software: a framework that combines genetic algorithms with supervised and unsupervised learning methods for optimal data segmentation in web documents (liet. *atvirojo kodo mašininio mokymosi programinė įranga: tai struktūra, kuri derina genetinius algoritmus su prižiūravimo ir neprižiūravimo mokymosi metodais optimaliam duomenų segmentavimui interneto dokumentuose*).

NLP – Natural Language Processing: an interdisciplinary subfield of computer science and linguistics. It is primarily concerned with giving computers the ability to support and manipulate human language (liet. *natūralios kalbos apdorojimas yra tarpdisciplininė sritis, leidžianti kompiuteriams apdoroti, analizuoti, interpretuoti ir samprotauti apie žmogaus kalbą*).

TED – Tree Edit Distance: the minimum-cost sequence of node operations that transform one tree into another (liet. *mažiausios kainos operacijų seka, leidžianti paversti vieną medį kitu*).

TN – True Negative: in statistical classification, this refers to outcomes where the model correctly predicts the negative class (liet. *teisingai neigiamas: statistinėje klasifikacijoje tai reiškia atvejus, kai modelis teisingai prognozuoja neigiamą klasę*).

TP – True Positive: in statistical classification, this refers to outcomes where the model correctly predicts the positive class (liet. *teisingai teigiamas: statistinėje klasifikacijoje tai reiškia atvejus, kai modelis teisingai prognozuoja teigiamą klasę*).

URL – Uniform Resource Locator (URL): colloquially known as an address on the Web, it is a reference to a resource that specifies its location on a computer network and a mechanism for retrieving it (liet. *universalus informacijos šaltinio adresas yra būdas aprašyti tam tikro informacinio resurso, pvz. tinklalapio, adresą internete*).

VIPS – Vision-based Page Segmentation: a method combining DOM and visual analysis for web page segmentation (liet. *vaizdu pagrįstas puslapio segmentavimas: metodas, derinantis dokumento objektų modelį (DOM) ir vizualinę analizę interneto puslapiui segmentuoti*).

WYT – Without YouTube: a term used to indicate data or analysis excluding information or metrics derived from YouTube (liet. *be YouTube: terminas, vartojamas duomenims ar analizei nurodyti, neįtraukiant informacijos ar metrikų, gautų iš YouTube*).

WWW – World Wide Web: Global system of interconnected documents and resources on the Internet (liet. *pasaulinis žiniatinklis: globali susijungusių dokumentų ir išteklių sistema internete*).

XML – eXtensible Markup Language: a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable (liet. *praplečiama žymėjimo kalba: žymėjimo kalba, apibrėžianti taisyklių rinkinį dokumentams koduoti formatu, kuris yra suprantamas tiek žmogui, tiek mašinai*).

Contents

INTRODUCTION	1
Problem Formulation.....	1
Relevance of the Dissertation.....	2
Research Object.....	2
Aim of the Dissertation	2
Tasks of the Dissertation	3
Research Methodology.....	3
Scientific Novelty of the Dissertation	3
Practical Value of the Research Findings.....	4
Defended Statements.....	4
Approval of the Research Findings	5
Structure of the Dissertation.....	5
 1. SEGMENTATION AND CONTENT BLOCK IDENTIFICATION	
IN WEB MINING.....	7
1.1. Web Mining Taxonomies	8
1.1.1. Existing Web Mining Taxonomies	9
1.1.2. Proposed Taxonomy for Web Mining Classification.....	12
1.2. Web Mining Automatisation	15
1.2.1. Popular Web Mining Automation Approaches	15
1.2.2. Challenges in Automatic Content Block Identification on Websites.....	18
1.3. Website Content Block Datasets	21

1.4. Conclusions of the First Chapter and Formulation of the Dissertation Tasks ...	23
2. SIMILARITY OF HTML CONTENT BLOCKS	25
2.1. Similarity Estimation Techniques	26
2.2. XML Similarity Estimation and HTML Blocks	28
2.3. Research on HTML Block Similarity Estimation Utilising XML	
Approach Performance	29
2.3.1. Data Similarity Estimation.....	29
2.3.2. Structure Similarity Estimation.....	29
2.3.3. Calculating HTML Block Similarity Score	31
2.3.4. HTML Block Similarity Research Results	31
2.3.5. Experiment Results	34
2.4. Tailoring Similarity Estimation Method for HTML Block Similarity	
Calculation	35
2.5. Proposed HTML Block Comparison Method and its Suitability Evaluation ...	36
2.5.1. HTML Block Comparison Method.....	37
2.5.2. HTML Blocks Similarity Estimation Experiments.....	39
2.5.3. Evaluation of the Results and Possible Improvements	43
2.6. Conclusions of the Second Chapter.....	44
3. MULTI-PURPOSE DATASET OF WEBPAGES AND ITS CONTENT	
BLOCKS.....	45
3.1. Proposed Structure of Dataset for Websites and Their Content Blocks	46
3.2. Research on the Proposed Dataset Structure Suitability and Its Data	
Peculiarities.....	50
3.2.1. Dataset Structure Completeness Evaluation	50
3.2.2. Research Methodology for Manual Web Page Segmentation	52
3.2.3. Analysis of Labelling Experiment Results.....	54
3.3. Evolution of the Dataset Structure Based on Further Research Insights	57
3.4. Conclusions of the Third Chapter.....	58
4. WEB PAGE CONTENT BLOCK IDENTIFICATION WITH EXTENDED	
BLOCK PROPERTIES	61
4.1.1. Definition of Block Range and Variation	63
4.1.2. General Idea of Web Page Division to Content Blocks with	
Extended Properties.....	65
4.1.3. Method for Block Range Estimation	67
4.1.4. Method for Block Variation Estimation.....	70
4.1.5. Novelty of the Proposed Methods.....	71
4.2. Validation of the Proposed Method for Web Page Division to Content Block .	72
4.2.1. Obtaining Data Suitable for Proposed Method Validation	72
4.2.2. Results of Web Page Division to Content Block Test Cases	76
4.2.3. Comparison of Results between Web Page Division to Content Blocks	
and Manual Labelling	78

4.2.4. Comparison of the Proposed Method with Existing Segmentation Methods	80
4.2.5. Evaluation of Validation Results	81
4.3. Establishing Connections Between Content Blocks from Different Web Page Regions	82
4.3.1. Recognising Sibling Blocks on a Website	83
4.3.2. Implementation of Remote Sibling Block Detection	87
4.4. Sibling Block Detection Validation.....	95
4.4.1. Data and Principles of the Validation	95
4.4.2. Results of the Proposed Method Testing	96
4.4.3. Validation Insights for Sibling Block Detection and its Limitations	101
4.5. Conclusions of the Fourth Chapter.....	103
GENERAL CONCLUSIONS	105
REFERENCES	107
LIST OF SCIENTIFIC PUBLICATIONS BY THE AUTHOR ON THE TOPIC OF THE DISSERTATION	117
SUMMARY IN LITHUANIAN.....	119

Introduction

Problem Formulation

Artificial intelligence (AI) solutions are increasingly used to make decisions and automate. Intelligent decisions often stem from data sets based on which models are trained. Therefore, the accuracy of the created models heavily relies on the quality, size, and detail of the available dataset. However, for now, high-quality and freely available data is somewhat limited. Organisations are still reluctant to share open data; part of the data must be marked by a person, and the available amount of marked data is insufficient.

One of the areas where very large amounts of data are used is internet web pages. Their volume on the World Wide Web is enormous, and the content is constantly updated. However, web pages are based on technology that is focused on presenting data to a human rather than a computer, e.g., all data is presented as text without delving into the metadata of the presented data, an indication of purpose, etc. Meanwhile, for computer use, it is necessary to understand the context of the data. Therefore, until all the creators and owners of online websites have switched to the use of new-generation internet technologies or the presentation of the same data in two formats (for users in HTML format and computers as an

API), solutions for identifying the type and purpose of the content blocks of online websites are necessary.

Although text analysis and generation technologies have significantly advanced in recent years, accuracy levels still fall short of fully automating the segmentation of web page content and identifying the purpose of each content block. Therefore, to improve the computer perception of the data of online web pages, it is necessary to expand the existing data sets in this area and use improved data sets to help create new solutions for scanning the content of web pages.

Relevance of the Dissertation

A detailed dataset of web page content blocks enables research to test various data analysis and classification solutions. These solutions would automate content block type identification, eliminating the need for manual intervention or a separate interface for computerised data processing and interpretation. Such identification of content blocks would offer greater flexibility in extracting the primary or new content on websites while ignoring design elements or other less valuable content like navigation elements.

Additionally, clear identification of website content blocks would enable the implementation of web page quality determination support systems. These systems allow for identifying missing blocks during web page creation, evaluating their most appropriate position on the web page, and so forth. This could be used as a basis for automated web page content, design updates, and generation. Old, outdated web pages could be automatically renewed to implement the latest technologies and design standards to meet the new usability requirements.

Research Object

Internet web page content blocks: methodologies for their detection, extraction, linkage of relationships, and documentation within the dataset.

Aim of the Dissertation

The dissertation aims to enhance the granularity of datasets comprising internet web page content blocks and streamline the manual labelling process by reducing the number of blocks requiring labelling.

Tasks of the Dissertation

The following objectives had to be completed to achieve the dissertation's aim:

1. To conduct a systematic analysis of the latest research in the field of data extraction from internet web pages.
2. To develop a web page data set structure, which would include a wide variety of features, defining different level content blocks and making it compatible with various types of data extraction methods.
3. To develop a solution aimed at partly automated web page content block labelling that would automatically establish relationships between the content blocks and group them, reducing the number of blocks requiring manual review and labelling.
4. To assess the developed dataset's effectiveness in identifying, grouping, and establishing relationships between content blocks in a web page and to evaluate the performance of the partly automated data labelling solution.

Research Methodology

In conducting this research, the systematic review and analysis of scientific literature was performed. Creative idea-generation techniques were employed for hypothesis formulation. Comprehensive data collection, examination, and processing methods were implemented. Practical experiments further emphasised the findings, adding depth and validity to the dissertation's conclusions.

Scientific Novelty of the Dissertation

The following scientifically new results were achieved:

1. A new taxonomy aimed at web data mining, which expands on already existing taxonomies and implements a broader range of possible data attributes and their variations. The proposed taxonomy facilitates a more comprehensive view of the web data mining area.
2. An enhanced method for determining content block inter-similarity allows for increased precision in detecting content blocks serving similar purposes on websites. Block similarity is evaluated using a complex approach that evaluates the combined similarity of block content and structure, considering block variations and parent content blocks.

3. Comprehensive dataset structure for storing website content and content blocks data that allows for using the same dataset for training fundamentally different classification methods.
4. A method that reduces the number of blocks that require manual labelling and review by detecting all content blocks within a website and building relationships between them.

Practical Value of the Research Findings

The presented methodologies and strategies offer significant utility in developing comprehensive datasets pertaining to webpage content blocks. When implementing these proposed approaches during the data labelling phase, users are presented exclusively with the content blocks that warrant labelling. In contrast, prevalent techniques often necessitate both the assignment of a block label and the delineation of block boundaries. The advocated methodology extends beyond merely recognising primary content blocks and facilitates the discernment of subsidiary content units embedded within principal blocks.

By streamlining the content block labelling process, resources can be redirected to amass more detailed information concerning the processed content blocks. Alternatively, if such detailed data collection remains optional, the duration of the labelling process can be significantly reduced. Implementing the suggested approaches promises to simplify the compilation of content block datasets and increase the volume of labelled data.

A robust and comprehensive dataset focused on website content blocks lays the groundwork for more diverse scientific research focused on webpage content analysis. With the integration of insights gained from such research, it is possible to develop and adopt advanced, more efficient, and intelligent solutions designed to analyse website content.

Defended Statements

1. By implementing the proposed webpage content block dataset structure based on a relational database, the same data can be seamlessly adapted for use in website content block analysis solutions that require diverse types of input data.
2. When determining the similarity of content blocks, relying solely on the likeness of text or HTML code structure is not sufficient. Combining these metrics allows for the increase of the detection precision for content blocks of similar purpose.

3. Through automated website content block similarity detection, grouping, and block relation-building, the need for manual labelling can be reduced by an average of 80% while ensuring that none of the blocks that require to be labelled are overlooked.

Approval of the Research Findings

Two scientific articles on the dissertation topic have been published in peer-reviewed journals with a citation index in the *Clarivate Analytics Web of Science* database (Griazev & Ramanauskaitė, 2021; Griazev & Ramanauskaitė, 2023). The research findings were presented at three international conferences:

- International Conference: “2018 Open Conference of Electrical, Electronic and Information Sciences” (*eStream*), 2018, Vilnius, Lithuania.
- International Conference: “6th IEEE Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE’2018)”, 2018, Vilnius, Lithuania.
- International Conference: “International Conference on Science & Technology” (*STRA*), 2023, Prague, Czech Republic.

Structure of the Dissertation

The dissertation is composed of an introduction, four chapters and general conclusions. It comprises 161 pages with 38 figures and 14 tables. A total of 103 literature sources were used in writing the dissertation.

Segmentation and Content Block Identification in Web Mining

This chapter provides a detailed examination of the complexities and challenges in web mining, a specialised branch focused on web-based data. Amidst the rapid growth of web pages, now exceeding 1.1 billion, the chapter identifies the primary challenge as the need to filter and verify the massive volumes of available web data, often unstructured or semi-structured.

The need for a comprehensive classification and understanding of web mining methods is emphasised. A novel taxonomy is proposed to offer a more holistic view of web mining approaches. This taxonomy aims to categorise existing solutions and identify their limitations, facilitating a better understanding of the field.

Furthermore, the chapter delves into the automation of web mining, assessing current methodologies for data extraction from websites and recognising the need for more refined techniques and comprehensive datasets.

This chapter underscores the significance of developing advanced web mining strategies. It sets forth clear objectives for future research to enhance the efficacy of data extraction methods and broaden the understanding of the web mining domain.

The main research results on the topic of this chapter were published in one author's publication (Griazev & Ramanauskaitė, 2018, April). The main findings of the research of this chapter were presented at one international conference

(“2018 Open Conference of Electrical, Electronic and Information Sciences” (eStream), 2018, Vilnius, Lithuania).

1.1. Web Mining Taxonomies

The number of web pages is growing quickly, reaching around 1.1 billion (Siteefy, 2023). Web page development occurs without supervision, leading to information overlap and the occasional publication of misleading or inaccurate content. While users previously faced a shortage of current, up-to-date data, today’s challenge is to track and filter the available information to determine its relevance and accuracy (Lovaraju et al., 2017).

Collecting and analysing web data might help with unchecked data. For instance, systems can collect internet information and consolidate it for specific user groups, as Aljawarneh et al. (2017) noted. Besbes et al. (2017) also described systems that scan the web for inappropriate or unlawful content and flag it for legal action when necessary.

Data on web pages is often presented in unstructured or semi-structured formats, which are not ideal for data analysis systems. With the semantic web not yet fully operational (Vandenbussche et al., 2017), deciphering data types or the underlying meaning in web pages remains a significant hurdle for more extensive web page data utilisation and in-depth analysis.

While the significance of web mining is widely acknowledged, a comprehensive classification and definition of web mining categories and technologies still needs improvement. Abundant research is available on various web mining concepts and techniques, as summarised by Jayamalini et al. (2017), but most are either narrowly focused or do not provide a holistic perspective. To have an overview of possible approaches to web mining and better understand the field, it was decided first to introduce a novel web mining taxonomy. Since the performed research is focused on improving the understanding of the web mining domain area and offering a possible solution for more accessible data extraction with automated methods, the first focus of the research is on existing methods and their limitations. The proposed taxonomy results from this research and can help categorise or describe existing web mining solutions by establishing primary criteria and categories.

Web mining is a specialised form of data mining. According to the Cambridge Dictionary, data mining is described as “<...>the process of using special software to look at large amounts of computer data in order to find out useful information<...>”. In contrast, web mining focuses exclusively on processing web-based data. As highlighted by Sharma et al. (2011), the evolution of web mining encompasses multiple eras, each of which focused on a specific task: Data

Collection (1960s), Data Access (1980s), Data Warehousing and Decision Support (1990s), Data Mining (2000s), and ultimately, Web Mining. Web mining remains an actively developed field, gaining traction in academia and the industrial sectors since the need to process large amounts of data remains relevant. Since the field of web mining is constantly evolving and vastly new, different approaches are being developed and introduced constantly, and many techniques should be considered when taking on the task of web mining. Taxonomies can be used to make it easier to have an overview of the web mining field. However, at the moment, comprehensive taxonomies detailing the domain of web mining are scarce.

1.1.1. Existing Web Mining Taxonomies

The need for taxonomies arose when the focus started shifting before and during the Data Mining era. One early taxonomy for web mining was introduced in 1997 by Cooley et al. (1997). However, this taxonomy is relatively basic, splitting web mining into two main sectors: web content and web usage mining. Web content mining identifies two categories (the agent-based and database approach). Web usage mining details predominantly concern application areas, with little emphasis on classifying different web mining methodologies.

Subsequent taxonomies again shifted their focus, categorising web mining into three core areas: data, structure, and usage mining. Kosala et al. (2000) clearly distinguish each category in terms of data types and views, methodologies, application sectors, and representation methods. The authors also highlight two perspectives within the web content mining category: information retrieval and database views. Additionally, web pages are categorised as either unstructured or semi-structured. These classifications are interconnected; typically, information retrieval pertains to unstructured pages, whereas semi-structured pages relate to data record identification and extraction.

Due to the rising amount of data, how data is gathered has become critical. So now, not only data extraction approaches but also unprocessed data retrieval and collection are essential to ensure that the techniques used would perform best. Arotaritei et al. (2004) have delineated the relationship between Information Retrieval and various instances of web mining (Fig. 1.1). These authors also detail application scenarios for web page mining and pinpoint instances that may span multiple web mining categories.

Since processing power rose, the possibilities to process data also expanded. Processing power improvements had a dual impact; one part allowed the collection of more data that would be beneficial when testing possible innovative approaches, and another part was that approaches that used to be too slow are now becoming viable. Laender et al. (2002) focused on tools for web data extraction, introducing a novel criterion for web data mining classification. The classification

includes two main types: HTML-based and NLP-based. These categories specifically pertain to semi-structured web data. HTML-based tools lean on the intrinsic structural elements of an HTML document, while NLP-based tools employ Natural Language Processing (NLP) techniques to derive extraction rules.

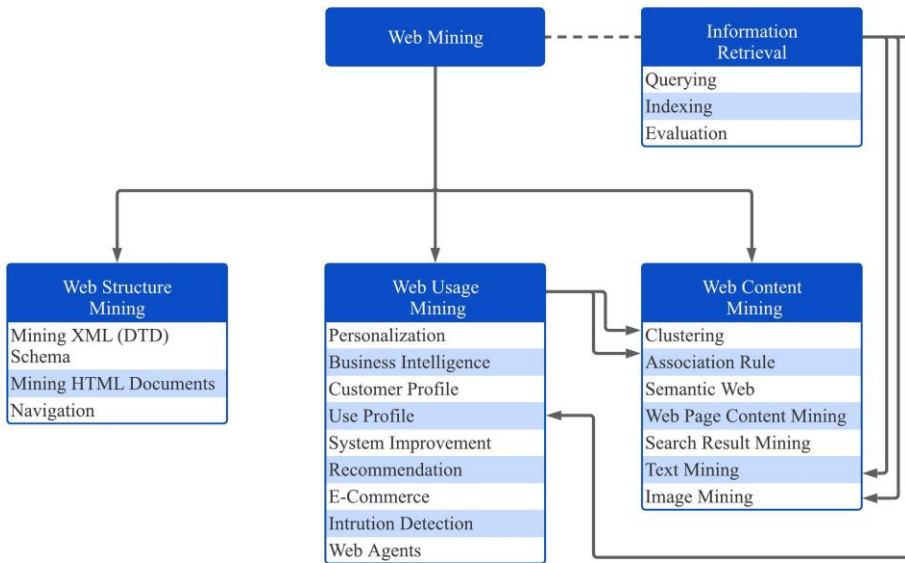


Fig. 1.1. Web mining taxonomy proposed by Arotaritei et al. (2004)

Another classification for web mining tools, specifically information extraction systems, is offered by Chang et al. (2006). Their contribution provides a fresh perspective on web data mining classification. Their research not only defines unstructured data but also maps out the relationship between the structure of a web page and the complexity of mining it, illustrated in specific instances (Fig. 1.2).

Conversely, Sharma et al. (2011) chose to overlook the structure of web pages, opting instead to categorise web content mining by the intended purpose of the web page: web page content, search page content, and result page content. Such a classification approach might be limited in its adaptability to emerging web page types since it primarily recognises only these three categories. Nevertheless, these researchers introduce a new set of criteria for classifying web structure mining, segmenting it into link structure, internal structure mining, and URL mining (Fig. 1.3).

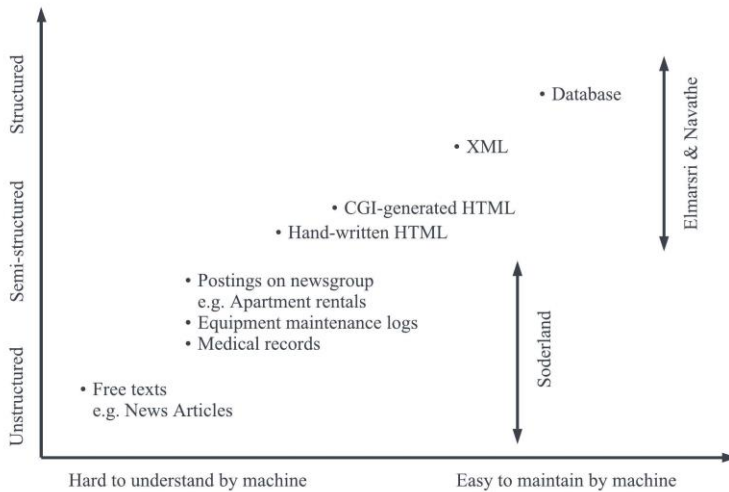


Fig. 1.2. Structuration of various documents based on data structure and adaptation for computer reading proposed by Chang et al. (2006)

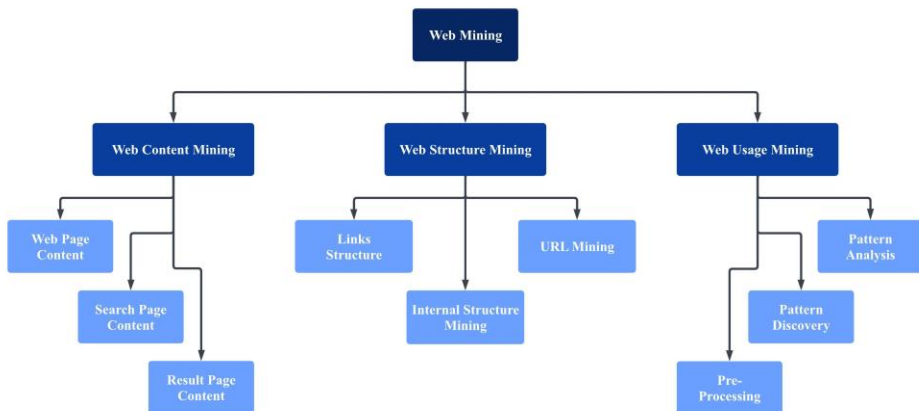


Fig. 1.3. Web mining taxonomy proposed by Sharma et al. (2011)

A thorough survey of web content extraction algorithms and techniques was provided by Al-Ghuribi et al. (2013), who introduced a new criterion for classifying web data mining solutions. One such criterion is the technique's applicability, differentiating between single-body and multi-body webpages. This classification is also pivotal for web structure mining, as multi-page web pages introduce new complexities.

Most recent classifications in web mining are focused on web usage mining, as described by Jain et al. (2017) and Kumar et al. (2017). This area was not well covered in earlier classifications, but the existing ones on web usage mining usually only focus on one specific aspect. For example, Jain et al. (2017) focused on the various stages of the web usage mining process, while Kumar et al. (2017) classified web usage mining based on the data source.

An analysis of the existing classifications for web mining and related areas shows that no detailed classification covers all the different aspects of web mining and includes the most recent categories. The lack of detailed classifications makes it difficult to understand the web mining area.

1.1.2. Proposed Taxonomy for Web Mining Classification

After evaluating other existing taxonomies, it was observed that most are specialised in specific segments of web mining and do not offer a broader perspective of the field. Consequently, a more comprehensive web mining taxonomy was proposed, building upon previous research while also introducing new classifications (Fig. 1.4).

Similar to the taxonomies reviewed, it is crucial to specify the type of information being mined; hence, the mining object (classification A) is included in the proposed taxonomy. Kosala et al. (2000) and subsequent authors have identified three principal object types, which are also encompassed in the proposed taxonomy: web content, structure, and usage. Web content focuses on mining the actual content displayed on a webpage, that is, the information presented on the site. On the other hand, web structure is concerned with mining the layout of information, disregarding the actual content and focusing solely on the web page's structure. Lastly, web usage zeroes in on mining information about the website's visitors, encompassing data about the user and their behaviour while navigating the site.

Kosala et al. (2000) suggested that web content can be further segmented based on its data structure (classification B) into unstructured, semi-structured, and web as a database. Textual content is typically unstructured, but it can sometimes exhibit a degree of structure, in which case it is classified as semi-structured. The semi-structured web page content is primarily displayed in a table-like format using HTML tables or other HTML tags to achieve a similar structure. This format is commonly used for listing data in rows or columns. Semi-structured content is relatively straightforward to extract and analyse. Additionally, some web pages can be categorised as "Web as Database". Such web pages are dynamically generated using database queries.

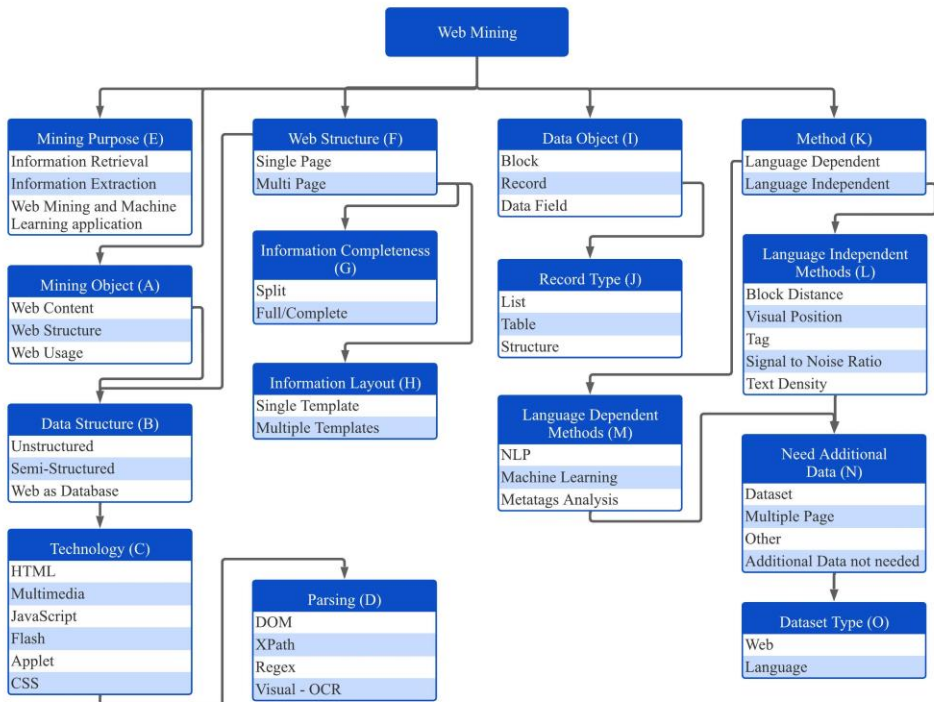


Fig. 1.4. Proposed Web Mining Taxonomy

The data structure is also significantly influenced by the technology (classification C) used to display website content. Content displayed using certain technologies is more straightforward to analyse; for example, HTML has an inherent structure, as does JSON. However, other technologies might necessitate a more sophisticated approach and yield less consistent results. For instance, analysing content displayed using Flash, which was popular some time ago but is not used nowadays, would require OCR (Optical Character Recognition) or similar technology for content analysis and extraction. The same applies to other multimedia content, such as images, videos, or sound, which only contain visual information that must be analysed and converted into other types of information or labelled. Content extraction based on visual cues and assumptions is discussed by Al-Ghuribi et al. (2013).

Various information parsing approaches (classification D) exist, and the optimal method depends on the type of information analysed. Structured or semi-structured data can be parsed using DOM, XPath, or RegEx, although RegEx is also suitable for unstructured data parsing. Meanwhile, visual data can be analysed and labelled using OCR.

In addition to the data structure, the structure of the web itself (classification F) is a critical factor in web data mining. This web structure can be divided into two main categories: single-page and multi-page websites. Moreover, the structure can be further broken down based on different parameters, such as the layout of the information and its completeness. The information layout refers to how the content or information is arranged on a web page, either using a single template or multiple templates (classification H). On the other hand, information completeness (classification G) refers to whether all the information is available on a single page or distributed across multiple pages, like when pagination is used to divide the content. This scenario necessitates analysis and data mining across several pages to obtain the complete information set.

The purpose of mining (classification E) specifies the analysis method and the actions required to process the website data to achieve the desired outcomes. For example, in Information Retrieval, a query is used to obtain information in a human-readable form. In contrast, information extraction involves the automatic extraction of data from structured or semi-structured, machine-readable sources, often necessitating the use of Natural Language Processing (NLP). In some scenarios, the purpose of mining might be to collect data for Machine Learning, which will then be used to train models for subsequent data extraction.

Lastly, the methods of data extraction or classification (classification K) can be divided into two main categories: language-dependent and language-independent. Language-dependent methods (classification L) mainly involve using NLP to process the website's content, employing Machine Learning algorithms, or analysing specific website elements, such as metatags. On the other hand, language-independent methods (classification M) focus on the specifics of the web page that enable the analysis and detection of the targeted content. These methods may involve measuring the distance between content blocks, analysing the visual positioning of the content within the website, tag analysis, and using the signal-to-noise ratio (SNR) to determine which parts of the website contain useful information and which parts contain secondary, additional content. Text density can also be employed to achieve comparable results and labelling within a webpage.

Language-dependent and language-independent methods may necessitate supplementary information (classification N) before any actions are undertaken. Specifically, language-dependent methods may need an initial dataset (classification O) comprising language data or actual web pages. For language-independent methods, stored web pages may be used to compare features with the target website during the analysis. Additionally, depending on the approach, data mining methods may require other supplementary information, such as additional pages within the webpage, to compare and predict the primary content location based on the layout of these extra pages. However, some methods may not require any extra information.

Data representation is another crucial factor in selecting the appropriate approach. For this purpose, the data object (classification I) can be specified. The mined data can be categorised as a block, record, or data field during mining. A record (classification J) can be further subdivided into a list, table, or structure. When the data object is defined as a block, it can represent any part of the webpage, meaning it could be a single information block or contain other blocks within it. The exact definition of a block is contingent on the information that needs to be mined. Records, while similar to blocks, usually have some structure since the data within a record is outputted from a database. Multiple records with similar structures are often found on the same page, albeit with some exceptions. Since records can be structured, it is possible to define the kind of structure a record uses, which can benefit data extraction. Various approaches can be employed to structure records; lists and tables can serve as default HTML tags for structured data, but other general-purpose HTML tags, such as DIV tags, can also achieve the same structure. Conversely, a data field represents a single endpoint information source, meaning it stores a single piece of information that can be extracted.

1.2. Web Mining Automatisation

The modern digital age has made automated data extraction a vital and constantly evolving field of research. This chapter evaluates different approaches for automated identification and extraction of data from websites. An assessment of what data is necessary to analyse webpage content and evaluate the effectiveness of chosen methods was performed. Additionally, it was investigated whether the data necessary for data extraction is available publicly and can be easily accessed and used for data extraction.

1.2.1. Popular Web Mining Automation Approaches

Web page segmentation methods are the most common techniques for extracting content blocks from web pages since many content blocks correspond to segments of web pages, as noted by Dias et al. (2014) and Win et al. (2014). These selected segments can be scrutinised repeatedly to reveal their internal composition. Many existing approaches to web page segmentation are rooted in visual page segmentation. Studies conducted by Cormier et al. (2016) and Zeleny et al. (2017) depend solely on visual analysis, thus removing the need for web page implementation technologies. However, based on Sanoja et al. (2015), approaches that rely on image segmentation usually require more computational resources than those based on the document object model (DOM). Research by Kiesel et al. (2021) indicated

that high performance is achieved through visual segmentation of web pages. Interestingly, the research found that the Vision-based Page Segmentation (VIPS) method showcased the best performance, combining DOM and visual segmentation solutions.

Incorporating DOM tree analysis allows for the evaluation of a wide range of metadata. For example, it can be used to identify related text on a web page, as done by Shu et al. (2022). Sun et al. (2022) used it to pinpoint malicious websites, and Ghaemmaghami et al. (2022) used it to segment blocks. Additionally, Kohlschütter et al. (2008) pointed out that the analysis of content and text density is also crucial for the segmentation of web pages.

The structure of the DOM and its associated characteristics are crucial for the segmentation of web pages (Xiang et al., 2007; Alcic et al., 2011). Some solutions are independent of language and explicitly designed for content extraction, such as the one presented by Wu et al. (2016). Certain web page segmentation solutions are designed with specific applications in mind. For instance, Sanoja et al. (2017) devised a method for converting HTML code from version 4 to version 5, which involves segmenting the web page and transitioning to HTML5. Image segmentation is utilised to identify various categories of images on a web page, and such a solution was presented by Manugunta et al. (2022).

Currently, the existing approaches do not allow for the complete retrieval of data on web page blocks, which is essential to broaden web mining capabilities. The current methods do not emphasise the HTML code scope corresponding to a particular block. Additional tags may surround the content and impact its presentation, affecting the variety of selectors or paths used to extract the content. Variations between blocks (relationships among siblings or blocks related hierarchically) are not considered when creating a more interconnected map of blocks.

Multiple datasets have been compiled to assess existing segmentation methods. One particularly popular dataset, created by Kreuzer et al. (2015), features a selection of web pages widely accessed at that time. The labelling of this dataset includes only a handful of main blocks on each web page, omitting details on composite elements. This dataset includes, on average, 13 labelled blocks per web page, with a median of 16 blocks. A comprehensive summary of accuracy metrics for different segmentation methods is presented in Table 1.1.

Table 1.1. Summary of existing web page segmentation methods accuracy metrics

Method	Precision	Recall	Accuracy	F-Score
BoM (Sanoja et al., 2014)	31%	26%	26%	28%
VIPS (Cai et al., 2003)	24%	26%	24%	25%

End of Table 1.1

Method	Precision	Recall	Accuracy	F-Score
SegBlock (Xu et al., 2016)	38%	40%	38%	39%
Semantic-Block (Jiang et al., 2019)	40%	43%	42%	42%
Fusion-Block (Ghaemmaghani et al., 2022)	45%	54%	48%	49%
Integrated Block (Ghaemmaghani et al., 2022)	52%	62%	54%	53%

Various approaches have been used in previous research for web page segmentation. Some of the most popular ones are:

- BoM (Sanoja et al., 2014), which integrates the structural, visual, and logical aspects of web pages.
- VIPS (Cai et al., 2003), which solely relies on the visual analysis of web pages.
- SegBlock (Xu et al., 2016), which considers the visual aesthetics, logic, and features of the content on a web page.
- Semantic-Block (Jiang et al., 2019), which employs the principles of Gestalt psychology.
- Fusion-Block (Ghaemmaghani et al., 2022), which is inspired by Gestalt principles and involves subsequential re-segmentation using semantic text similarity.
- Integrated Block (Ghaemmaghani et al., 2022), which utilises the DOM structure, vision-based analysis, and text-based similarity metrics analysis for web page segmentation.

Research efforts specifically focus on identifying content blocks within web pages (Debnath et al., 2005b; Li et al., 2009). These methods can accurately identify the main structural blocks of a web page and are designed to analyse multiple pages within the same website. Therefore, such methods are unsuitable for single-page websites or those with varying designs across sections or pages. These early solutions do not estimate block boundaries or variations despite the importance of relationships between similar DOM elements, as Grigera et al. (2021) noted.

Martinez-Rodriguez et al. (2020) stated that many researchers are working on identifying individual content blocks on web pages. This is typically done by analysing the text within the content blocks. However, the potential of this field is limited due to the need for high-quality datasets suitable for machine-learning models.

Most currently available datasets and methods are designed to identify segments of web pages rather than individual content blocks. A single segment may include multiple content blocks, which are merged into a more significant,

visually cohesive segment. Even though the datasets only contain segments and main content blocks and lack details on internal blocks or code ranges, the accuracy metrics could be better. These accuracy scores indicate the field's complexity and emphasise the need for diverse solutions to fully master web page content segmentation and division into content blocks.

1.2.2. Challenges in Automatic Content Block Identification on Websites

Web data is commonly presented in a semi-structured style, where information is incorporated into the design and layout of web pages rather than in separate, specifically designed content blocks that would allow for easier data identification. This is happening because the vision of a fully semantic web has yet to be fully realised (Sleiman et al., 2013). Although HTML5 introduced tags to enhance structural clarity, their primary function is defining the page layout. Newly added tags to define articles, primary (main) or secondary (aside) content, headers, footer, figures, and content sections combined with previously available HTML content tags like various headings, paragraphs, and lists help partially identify the information. However, several reasons exist why identification performed while relying exclusively on tags needs to be more precise and requires additional features for block identification. First, tags need to be more specific and clearly describe the enclosed information. This is impossible with the limited number of tags like paragraphs, headers, headings, and others. The distinctiveness of tags is further reduced because the same tags are inevitably repeated throughout a single page. Misusing tags may also impact SEO, but there are no other consequences, so the rules for tag use are not always followed, and content is not structured, as well as possibly using the latest technological advancements in the field. For example, lists frequently serve as navigation menus and images as background visuals instead of content-related imagery. Improper use of headings may also diminish their importance ranking (from `<h1>` – most crucial, to `<h6>` – least crucial). Additionally, the complexity introduced with CSS styles makes it challenging to differentiate elements based on visual indicators, as it allows various elements to appear alike.

The endeavours of search engines further corroborate this informational gap to interpret the presented data to improve their search outcomes. Although the precise techniques employed by search engines to discern website information remain undisclosed, it is clear that it is a complicated undertaking. It can be assumed that even search engines struggle to accurately identify information presented on websites, as evidenced by the provision of supplementary tools to web admins to augment the accuracy of this identification procedure. A prime example is schema.org, initiated by Google, Microsoft, Yahoo, and Yandex, which strives

to annotate web data in a format more digestible to search engines. To implement schema on a website, developers must classify web data using the lexicons supplied by schema.org. Search engines then leverage this information to yield more exhaustive and precise search results, which would otherwise be unattainable or highly susceptible to inaccuracies.

Web page content extraction is a multifaceted challenge. A prevalent task in web content extraction entails the removal of irrelevant content components. Content removal is commonly accomplished by pinpointing the web page template and amassing only the pertinent content Sleiman et al. (2013). Another distinct scenario involves extracting data from tables or lists exhibited on the web page, which was covered by Roldán et al. (2020). A method to extract such data from a web page was proposed by Grigalis et al. (2014). However, this approach is limited to situations where multiple records are visually organised and form a structure.

A crucial part of content extraction is web page segmentation, which involves assigning labels to content blocks based on their function or the content that one presents, such as header, menu, main content, and others. Cai et al. (2003) formulated these objectives into the previously discussed VIPS (Vision-based Page Segmentation) algorithm. Due to various reasons, specific data may be missing from a web page's source code, such as data that is asynchronously loaded to improve page performance. To harness such data, the extraction method must process the fully rendered page content. After page rendering, Nguyen et al. (2012) harnessed visual data and extracted ancillary information from the DOM (document object model). Burget et al. (2009) proposed a segmentation approach that considers the location of the data on the page layout and integrates visual indicators to aid in the process. Such an approach allows for the inference of some data based on its position within the page layout. Feng et al. (2016) proposed a novel approach based on the information display design. Their algorithm works in multiple phases, starting from the top and dividing the content into rows. Then, the rows are further subdivided into inner rows and columns using visual separators. Finally, labels are assigned.

Bing et al. (2014) advocated for segmenting a page using boundaries, which are instituted across the page based on visual indicators and subsequently labelled to ascertain if the boundary partitions content. This labelling process considers the local features of the analysed boundary and context features predicated on adjacent boundaries. Vargas (2015) introduced the Block-o-Matic (BoM) algorithm for web segmentation in his dissertation. It segments the page without prior knowledge, employing heuristic rules delineated by W3C Web standards. Figure 1.5 provides a sample of web page segmentation. To make it easier to understand, only the key blocks have been labelled. The main blocks identified in Figure 1.5 include:

1. Header, which is the topmost section of the web page.
2. Navbar, which is the menu dedicated to navigation on the web page.
3. Hero is the initial block on a website that typically contains images, texts, links, or forms.
4. Article Preview, which block shows the headline and snippet of the article.
5. Event, which provides the primary information about the event.
6. Contact Info Block, which displays the primary contact details.
7. Social Menu, which contains icons with links to social networks.



Fig. 1.5. Website content segmentation example

Using only the HTML structure for identifying data is complex and can sometimes lead to problems, so visuals are often used to identify blocks. The VIPS algorithm is one of the most popular tools used for the segmentation process (Andrew et al., 2019; Kuppusamy et al., 2011; Bartík et al., 2010). Akpınar et al. (2013) improved VIPS by adding HTML5 tag support to the original algorithm, improving the management of invisible nodes, and adapting the algorithm to run on Java. Zhang et al. (2013) noticed that, despite being popular, VIPS has some limitations, one being the difficulty in determining the partition granularity value. This value is unique for each case and changes with each segmented page when using VIPS. Having different granularity values for each analysed page involves

additional manual maintenance that slows the segmentation process. Zhang et al. (2013) resolved this issue by suggesting a method that uses a fixed granularity value for all segmented pages.

On the other side, blocks with the same purpose usually have a similar structure. As web development is becoming more automated, using various tools and systems to speed up the process, the chances of blocks with the same purpose having the same or similar structure are higher. Therefore, the HTML structure can be used for identifying blocks, reducing the need to analyse block contents, which can be in different languages or have different visual styles.

Some methods analyse multiple pages from the same domain to identify constant and variable information blocks. These information blocks could be different pages with different information or the same page but in a different language. However, this approach is possible only sometimes since single-page websites are becoming increasingly popular, and there may be only one page on the same website to compare. Additionally, different pages within the same domain may have different layouts or even designs, which can make it difficult to identify blocks using this method.

1.3. Website Content Block Datasets

Blocks can also be identified using machine learning methods. Depending on the extraction approach, these models require various source data for training. Researchers often use their datasets to test their algorithms. For example, Burget et al. (2009) gathered 100 documents from ten news sites, then manually classified visual areas into nine categories related to the article and a tenth for irrelevant data. Vargas (2015) curated a collection of websites organised by categories such as Blog, Forum, Picture, Enterprise, and Wiki. A total of 25 websites were chosen per category, and from each, a single webpage was collected and saved. This approach ensured that a diverse range of sources were included in the collection. The produced dataset consisted of a collection of DOM (Document Object Model), CSS (Cascading Style Sheets), and image files. Including these different file types allowed for a more comprehensive analysis of the web page under consideration, enabling a better understanding of its structure, design, and content. Andrew et al. (2019) utilised a dataset comprising 53 web pages from Tourism, E-Commerce, and News domains.

Open-source datasets are also used for this purpose. Utiiu et al. (2018) benchmarked their content extraction method using Cleaneval and Dragnet datasets. Cai et al. (2003) conducted research using the Web Track dataset to compare the effectiveness of the VIPS algorithm with the DOMPS algorithm. Meanwhile, Bartík et al. (2010) utilised several datasets, including WebKB, which comprises 4518

web pages from computer science department websites classified into six categories. Another dataset was created manually by Zhang et al. (2013) from multiple English news websites, annotated, and assigned six categories. It contained almost 500 pages in total. Kreuzer et al. (2015) created two datasets, one from popular websites and one from random websites, due to the expected complexity of popular websites. The labelling process was done for some blocks on the first and second levels, using labels like Header, Footer, Content, and Sidebar for the first level, and Logo, Menu, Title, Link-list, Table, Comments, Ad, Image, Video, Article, Search bar for the second level. Such labelling provided much data but was limited as the labelling process covered only two levels, and only 15 block labels were used. Yao et al. (2013) used the Dagnet and L3S datasets for their research. The latter included 621 articles from Google News that were manually classified.

Web datasets typically only contain the source code of a webpage, without any additional assets used on the page. The source code is stored to ensure that the webpage can be analysed later without worrying about being changed or deleted. However, many stored pages are old and contain only textual content. As a result, outdated web pages may not accurately reflect current web development trends and can differ in structure and code from modern web pages.

It is worth noting that some publicly available datasets only contain the initial or source data. These datasets do not have any extracted data, which is essential to test the performance of data extraction algorithms. A review of several datasets (Table 1.2) was conducted to determine the information contained within them. A “+” sign indicates data present in the dataset while missing data is marked with a “-”.

Almost all the reviewed datasets contained the HTML source code. Some even included extracted text data stripped of tags, making it “clean text”. Having HTML source code is often sufficient for specific tasks, like extracting the context. However, additional data should be included to make these datasets more versatile.

Two datasets, both created by the same authors (Kreuzer et al., 2015), distinguished themselves by including the HTML source code and the HTML DOM code. The DOM code is the result of the browser processing the HTML page. It can differ from the source code because browsers can automatically correct errors or include content generated dynamically, often through JavaScript. These comprehensive datasets also included labelled HTML and other assets like CSS, JavaScript, and images. Having these assets allows for the emulation of the web pages in a local setting. Without them, the rendered version of the saved web page would not match the original.

Table 1.2. Comparison of data stored by existing web page datasets

Datasets	Data Stored by Datasets							
	HTML	DOM	CSS	JavaScript	Images	Screenshots	Clean Text	Labelled Blocks
Cleaneval, Utiu et al. (2018)	+	–	–	–	–	–	+	–
Dragnet, Peters et al. (2013)	+	–	–	–	–	–	+	–
Gosh, Vargas (2015)	–	+	+	–	+	–	–	–
Dataset-random, Kreuzer (2023)	+	+	+ ¹	+ ¹	+ ¹	–	+	+
Dataset-popular, Kreuzer (2023)	+	+	+ ¹	+ ¹	+ ¹	–	+	+
wt10 g, Sleiman et al. (2013)	+	–	–	–	–	–	–	–
Web-KB, Zhang et al. (2013)	+	–	–	–	–	–	+ ²	+ ²
L3S, Kohlschutter et al. (2010)	+	–	–	–	–	–	–	+ ³

¹ There are 15 labels, four for 1st-level blocks and 11 for 2nd-level blocks.

² The text is cleaned by removing links and labelling them within the source code.

³ The data is categorised into five labels: headline, full-text, supplemental, user comments, and related content.

Despite offering a wealth of source code and assets for diverse data extraction methods, these datasets still needed some elements. None provided a snapshot of the web page and its content blocks, which is crucial for specific types of website analysis. For instance, some analysis methods rely on Optical Character Recognition (OCR) or other visual cues for data identification.

1.4. Conclusions of the First Chapter and Formulation of the Dissertation Tasks

1. The scientific literature indicates a notable absence of comprehensive taxonomies or classifications in web mining. This gap complicates the understanding of the domain as a comprehensive perspective must be pieced together from various sources, leading to potential inconsistencies in terminology. These challenges underscore the necessity for a more detailed and expansive web mining taxonomy.
2. The proposed taxonomy builds upon existing classification criteria, thereby maintaining consistency with currently used terminology, but also introduces new categories and classifications. This inclusion aims to

encapsulate the evolving dynamics and methodologies within web mining, offering a more comprehensive and up-to-date framework that reflects the field's current state and emerging challenges.

3. A review of existing solutions for automated web mining revealed various methods focusing on text, structure, and visual data. However, most must be more accurate (the accuracy metrics hover about 50%) for detailed web page content block identification at different levels. Therefore, a new approach is needed, oriented toward hierarchical content block identification and relationships between the blocks.
4. A critical factor impacting the accuracy of existing web mining methods is the inadequate diversity and depth of available datasets. The existing datasets provide a limited range of data that could be used for different types of methods or even combinations of them. By consolidating the structure of different datasets and supplementing them with tools for simplified data set filling, the web mining-adapted datasets could be developed faster and provide a better possibility for using a wider range of properties in the web mining machine learning models.

Considering all results and conclusions of the chapter, the dissertation tasks are reformulated as follows:

1. Conduct a systematic analysis of the latest research in the field of data extraction from internet web pages.
2. Investigate possibilities for web page content block similarity estimation and detection.
3. Expand the existing data sets structure for internet web page blocks to be compatible with various types of data extraction methods.
4. Develop a solution to detect website content blocks, with relationships and hierarchy between them.

Similarity of HTML Content Blocks

The current chapter of the dissertation addresses the challenge of automating data extraction from web pages by identifying relevant content blocks amidst a plethora of information. This chapter discusses various methods for estimating the similarity between different HTML blocks, which is crucial for distinguishing main and irrelevant web page content.

The focus is on analysing semi-structured or unstructured web data, where traditional methods fall short due to their reliance on manual processes. The dissertation proposes advanced context-based analysis methods, highlighting a gap in existing techniques for estimating HTML block similarity and the need for comprehensive labelled datasets.

This chapter introduces a novel method for assessing the similarity between two HTML blocks, considering data content and structural aspects. The method incorporates techniques from XML similarity estimation, adapted to the specific challenges of HTML. Through a series of experiments and analyses, the dissertation demonstrates the potential of this method in improving the accuracy of automated web data extraction.

The main research results of this chapter were published in one author's scientific publication (Ramanauskaitė & Griazev, 2018, November) and were presented at one international conference (*"6th IEEE Workshop on Advances in*

Information, Electronic and Electrical Engineering (AIEEE '2018)', 2018, Vilnius, Lithuania).

2.1. Similarity Estimation Techniques

Automating data extraction necessitates the crucial task of identifying information blocks within a website. A typical web page is laden with information, of which only a fraction is pertinent. Hence, it is imperative to distinguish different blocks on a web page to eliminate irrelevant content. Two approaches to this are either identifying the main content and discarding the rest or pinpointing the irrelevant content and retaining the remainder. Various methods can be employed for identification, such as data similarity estimation, structure similarity estimation, or a hybrid of both. Additionally, comparisons can be made against other pages within the same website or against a specific dataset.

An algorithm must process a page to identify HTML content blocks. The field of web page parsing and data mining is evolving swiftly, with structured web page data extraction methods being developed for straightforward and semi-automated practical use. However, the scenario for gathering semi-structured or unstructured data differs significantly: existing methods are predominantly manual, requiring a specific data template to extract data from a particular section of a web page.

For semi-structured or unstructured web page data gathering, context-based data analysis methods can be employed to advance the field. Despite its potential, this approach has yet to gain traction, and no application-level tools are available. The sluggish progress in this area is attributed to a dearth of methods for estimating similarity between HTML blocks and a lack of publicly accessible labelled datasets for web page structure, which is essential for developing data identification tools.

For humans, judging similarity is usually relatively straightforward. When asked, “Do these two objects look alike?” the typical responses are “Yes”, “No”, or “Somewhat”. On the other hand, computer-based decision systems require a much more precise metric to determine the degree of similarity between two objects. Usually, the degree of similarity between two objects is expressed as a floating-point number within the range of $[0; 1]$, where 0 indicates total dissimilarity, 1 indicates complete identity, and any values in between denote the level of similarity between the objects.

Calculating similarity for basic data types is a well-established practice in most computer applications. For instance, the similarity between two numeric values, V_1 and V_2 , can be computed using the formula (2.1) presented by Chbeir et al. (2005), where the similarity, $\text{sim}(V_1, V_2)$, is calculated as the difference between the two values divided by their sum, subtracted from 1.

$$sim(V_1, V_2) = 1 - \frac{|V_1 - V_2|}{|V_1| + |V_2|} \quad (2.1)$$

Estimating similarity for Boolean variables is also straightforward, as the values can either be the same or different, with no other possibilities. Traditionally, the similarity for Boolean variables can be calculated by simply comparing if the two values, V_1 and V_2 , are the same or by using a logical expression (2.2).

$$sim(V_1, V_2) = NOT(XOR(V_1, V_2)). \quad (2.2)$$

While these examples are relatively simple to implement, the challenge arises when dealing with more complex data types. Evaluating text similarity, is a critical task for search systems, as results must be ranked based on their text similarity. Various string metrics exist that quantify the differences between two strings: Levenshtein distance (Yujian et al., 2007), Hamming distance (Norouzi et al., 2012), Jaccard Coefficient (Niwattanakul et al., 2013), etc. Despite the differences in these algorithms, text comparison is typically represented as an edit distance, $dist$, interpreted as the number of single-character edits (insertions, updates, deletions) needed to transform one text value, V_1 , into another, V_2 . Since the edit distance will not exceed the sum of the lengths of both texts, text similarity can be estimated as the proportion between the edit distance and the sum of the lengths of the texts, $len(V_1)$ and $len(V_2)$, subtracted from 1 (2.3).

$$sim(V_1, V_2) = 1 - \frac{dist(V_1, V_2)}{len(V_1) + len(V_2)}. \quad (2.3)$$

The text similarity quantifies the likeness in the letters and their sequences in both texts. However, the abovementioned algorithms do not account for semantics, which is essential in content mining. For example, when comparing two synonyms or determining if one value is a subset of another, semantic similarity becomes more important than text similarity. Calculating semantic similarity necessitates additional data, usually in the form of a dictionary or ontology that defines topological similarities and relations between different words or terms. Although the methods can vary significantly, the main idea is to calculate the distance between two terms in the ontology, where each move (up or down) in the ontology adds 1 to the value as suggested by Pekar et al. (2002) or increases the distance value by one only if the second term, V_2 , is at a lower level compared to the first term, V_1 as suggested by del Pozo et al. (2006). The choice of semantic similarity measure is highly dependent on the context or the knowledge base.

In practice, multiple methods are often used to define the similarity between two XML elements (Fig. 2.1). XML documents have an advantage over HTML documents: tags in an XML file are self-explanatory. Such tags can be analysed as part of the context, whereas HTML tags are used for presentation purposes rather than explaining the presented data. Therefore, not all XML document similarity estimation methods can be applied to HTML document similarity estimation.

Due to the adaptation of HTML for viewing by individuals, some methods for web page similarity can be implemented using visual web mining like the one proposed by Youssefi et al. (2004). This approach bypasses the need for an HTML code structure and can compare the data and its presentation layout. However, the complexity, speed, and accuracy of this type of method remain challenging.

2.3. Research on HTML Block Similarity Estimation Utilising XML Approach Performance

To measure a specific part of a web page, including data, text, and presentation, it is necessary to consider the HTML code structure. Hence, the process of estimating HTML block similarity is divided into three phases: data similarity estimation, structure similarity estimation, and HTML block similarity calculation.

2.3.1. Data Similarity Estimation

The possibilities of HTML code formatting enable coding elements sequentially, one after another, and presenting them in a different order. Therefore, it is impractical to compare the text of the HTML code block as is. It is suggested that tags containing text (those starting with text rather than another tag) should be identified, and all possible sequences from these containers should be generated. If an HTML block has “ n ” text containers, there will be $N=n!$ combinations of text containers. The tags should be eliminated, leaving only the text, and all N combinations with merged containers should be used for semantic similarity. The use of the solution presented by Dong et al. (2009) is proposed, as it considers both the meaning and its relationships within the knowledge base. The maximum value of all N combinations should be selected as data similarity values.

2.3.2. Structure Similarity Estimation

Visualising how a code block will be presented becomes complicated when it is just a part of the HTML document and lacks context. Therefore, visual

mining techniques are unsuitable in this situation, and the HTML code structure should be analysed exclusively. For this purpose, building the tag tree of the HTML code block is suggested. The tags will form a tree, where only tags with no data and attributes will be used. Additionally, a text node will be added as a child element for those tags that contain text to reflect where the text was in the HTML code.

As Chbeir et al. (2005) summarised, there might be four main cases between HTML trees “i” and “j” (Fig. 2.2). The situation “Equal” will be equivalent to 1 for code structure similarity. However, “Overlap” is also suitable for HTML code structure and should be evaluated as 1. “Overlap” means two situations: the first HTML code is bigger, or the second one is bigger. In both cases, one code is certainly nested within the other. It is not a perfect situation for code structure similarity. However, if the code includes additional data and text, this fact will influence the final similarity mark by reducing it. Meanwhile, some additional tags are not significant enough to define the HTML block boundaries.

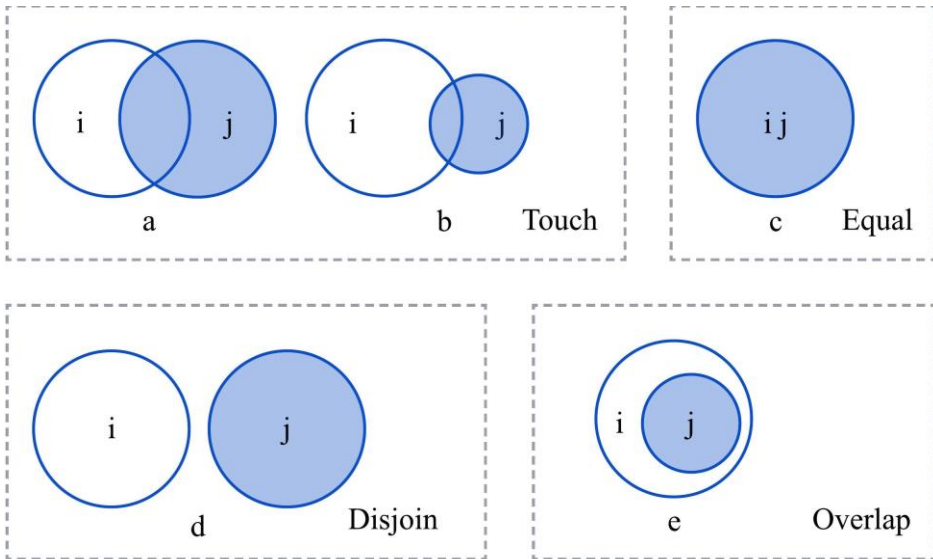


Fig. 2.2. Asymmetric similarity issue between XML documents (Chbeir et al., 2005)

“Disjoin” means no matching between these two HTML code structures, so its values will equal 0. The “Touch” situation will reflect the values from the interval (0; 1) as just a part of the structure is similar. The value will be calculated as follows:

- The distance will remain the same when climbing the tag tree down or up.

- Adding or removing a tag from the path will not change the distance.
- The distance will be increased by one if a tag has to be changed (removed and added for the same tag).
- If the tree has data nodes unmapped to another tree, it increases the distance by one for each unmapped data node.
- The distance will be divided by the number of elements in the tag tree.

Another critical aspect of HTML documents is using different tags for the same purpose. Therefore, the structure similarity evaluation must use an additional feature, e.g., a database of analogous tags. For example, a paragraph can be implemented by using “p” or “div” tags. This complicates the comparison of two trees but reflects the idea when different HTML codes produce the same view.

2.3.3. Calculating HTML Block Similarity Score

By utilising this approach, the computation of HTML block similarity considers data and structure aspects, providing a more comprehensive assessment of the similarity between two HTML blocks (2.4). This method allows for flexibility in evaluating different components of the HTML code and accounts for variations in tag usage and structure, making it suitable for practical applications in web mining and data extraction tasks.

$$sim(V_1, V_2) = sim_{data}(V_1, V_2) \cdot sim_{structure}(V_1, V_2). \quad (2.4)$$

The method proposed for calculation reflects a scenario where the similarity between HTML blocks will register as zero if there is no overlap between the data in the blocks. On the other hand, the chance of obtaining a structure similarity of zero is extremely low, given that there will almost always be some matching tags or data elements.

2.3.4. HTML Block Similarity Research Results

To assess the effectiveness of the HTML block similarity evaluation, several examples were meticulously scrutinised, and their similarity values were deduced. Here, one example is detailed to elucidate the core principles of this estimation technique (Figs. 2.3 and 2.4).

```

1 <div class="container">
2   <div class="row">
3     <div class="col-4">
4       Enterprise <b>contact</b> information
5     </div>
6     <div class="col-8">
7       Address of the correspondence 
8     </div>
9   </div>
10 </div>

```

Fig. 2.3. Example HTML code block A

```

1 <table>
2   <tr>
3     <td>Address of the correspondence</td>
4     <td>Enterprise contact information</td>
5   </tr>
6 </table>

```

Fig. 2.4. Example HTML code block B

When comparing two HTML code blocks, example A was selected as the primary focus. Various combinations of this code were then created and compared with the unchanged code B. The goal was to extract only the text starting a tag, leading to the identification of two data containers. Subsequently, this data was cleaned by removing all HTML code or formatting. This cleaned data and its similarity measurements are shown in Table 2.1. Since only three letters differ in the second case, the similarity is very high, at 0.98. This higher value is used to estimate the overall similarity of the HTML blocks later on.

Table 2.1. Comparison of data similarity estimation cases

No.	Block A data used for comparison	Block B data used for comparison	Similarity score, %
1.	Enterprise contact information Address of the correspondence	Address for the correspondence Enterprise contact information	0.9%
2.	Address of the correspondence Enterprise contact information		97.5%

The comparison of the HTML block structure, as illustrated in Figure 2.5, reveals that the tree structure of the tags is somewhat similar despite some differences in the tags used. A rule was applied that treats “tr” and “td” tags as

equivalent to “div” tags, considering their common purpose of defining a block of content. In Figure 2.5, the colours indicate which tags correspond to each other. Two tags at the top (div and table) are unnecessary and could be removed or added without affecting the comparison. Since adding or removing tags does not affect the distance calculation, and all other tags correspond perfectly, the structure similarity of these two HTML blocks is considered to be 1.

When comparing the HTML code blocks A and B, the overall similarity is calculated to be 97.5% because 0.975 (data similarity) multiplied by 1.00 (structure similarity) equals 0.975. This indicates that the texts are very similar, and these two blocks appear almost identical on an actual web page.

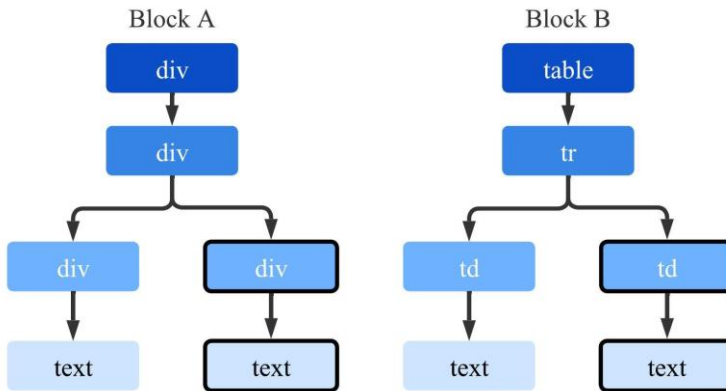


Fig. 2.5. Comparison of structures for blocks A and B

Conversely, comparing a different code block, such as code block C (Fig. 2.6), with code block B would involve calculating the data similarity using only one combination. It would result in a much lower similarity value of 56.8%. This means that, while there are some similarities between code blocks B and C, the resemblance is significantly lower compared to code blocks A and B.

```

1  <div>
2      <div>
3          <div>
4              My Address
5              <div>
6                  for correspondence
7              </div>
8          </div>
9      </div>
10 </div>

```

Fig. 2.6. Example HTML code for block C

In the case of comparing the structure similarity between code block B and code block C, the mapping information shown in Figure 2.7 reveals that one text block is missing in code block B. As a result, the structure similarity is calculated to be 66.6%. This percentage indicates that, although there are some structural similarities between the two code blocks, there is a significant difference due to the missing text block in code block B. Therefore, even though there is some overlap in the structure of the two blocks, they are not highly similar in terms of their structure.

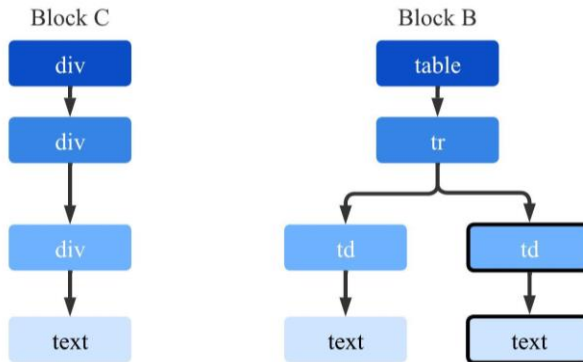


Fig. 2.7. Comparison of block C and B structures

The similarity score of 37.8% between HTML code blocks C and B underscores the significant differences. Although there are some similarities, as reflected by the score not being zero, the differences in data and structure result in a lower overall similarity. This score accurately reflects that the visual presentation of the two examples would be different. This is a logical outcome as the calculation considers both the data and the structure, which are critical components in determining how the content is ultimately presented and perceived by the end user.

2.3.5. Experiment Results

Analysing various methods to estimate similarity reveals standard techniques for simple data, whereas complex data heavily relies on context. Thus, in determining the similarity of HTML code blocks, methods akin to those for XML documents can be employed, albeit with consideration for HTML's distinct characteristics.

The suggested method for assessing the similarity between HTML code blocks encompasses both data and structural similarities. For the data component, existing methods are utilised with a modified approach to text preparation. A novel strategy, alongside a unique distance calculation method, is proposed for evaluating structural similarity. These adjustments enable the demonstration of similarities on a scale from 0 to 1, necessitating a match in both meaning and presentation style.

2.4. Tailoring Similarity Estimation Method for HTML Block Similarity Calculation

Estimating the similarity between HTML code structures typically involves comparing their DOM tree structures (Buttler, 2004; Nyein, 2011; Shasha et al., 1990). This approach is effective when the goal is to ascertain whether the compared code blocks are identical. However, this method encounters challenges. For instance, blocks that serve identical functions can possess distinct structures. More critically, blocks can exhibit identical structures but employ different tags, leading to a 0% similarity match when the blocks utilise divergent tags. Yang et al. (2001) noted that some algorithms attempt to mitigate this by comparing the attributes employed within tags. While this can slightly enhance the estimation score, it often fails to significantly improve the similarity score since attributes are optional and not all are obligatory. This is also true for attribute content, which can be tailored to the specific needs of each attribute on a webpage. Another approach is to compare the tags used and the tree structure, disregarding the tags. This method enables the assignment of a 50% similarity score to two blocks that have differing tags but identical structures.

While structure comparison is a favoured method for determining HTML block similarity, it often devolves into comparing the tags used between the HTML blocks analysed. This essentially transforms it into a structured text comparison. If different tags are employed to construct identical structures, comparing these blocks will yield a 0% similarity score wherever the tags differ.

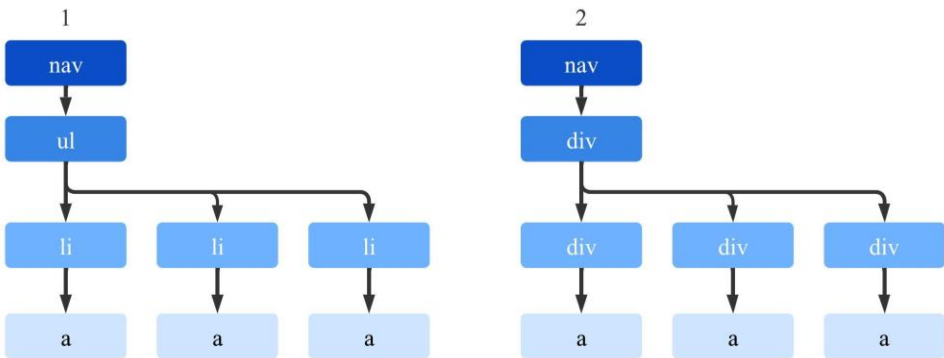


Fig. 2.8. Examples of different tag structures, which can be styled to have the same visual representation within a web browser

The tree structure illustrated in Figure 2.8 demonstrates that employing different tags can yield the same structure, enabling these blocks to appear identical

to the user within a browser window despite having divergent HTML code. Tang et al. (2015) discuss tree edit distance (TED) in their work, which can also be employed to compare HTML snippets based on their structure by computing tree edit operations. The primary operations considered are insertion, deletion, and renaming. Utilising these calculations may enable the comparison of differently structured HTML snippets and facilitate the determination of their similarity. This is crucial because different blocks serving identical purposes can exhibit slightly varied structures. For instance, a navigation block can have five items or ten items. Tang et al. (2015) stated that using TED makes it possible to ascertain the similarity between these two navigational blocks more precisely.

An alternate approach regarding attributes involves comparing classes, which can prove advantageous when a front-end framework is utilised, thereby unifying class names. Gowda et al. (2016) described this approach but also noted that without framework use, the developer solely determines class names and does not conform to any specific specification regarding class naming. Consequently, if the class comparison is employed, it is also necessary to compare the CSS properties associated with these classes to enhance the likelihood of obtaining a higher similarity score.

Comparing HTML does not warrant a singular approach due to its fluidity. Optimal results are attained by comparing blocks using various metrics. Consequently, Debnath et al. (2005a) proposed two algorithms: ContentExtractor and FeatureExtractor. The ContentExtractor algorithm compares different blocks and yields a similarity score based on the cosine between the feature vectors of the two blocks. These features include the number of terms, images, etc., and pose a challenge when comparing two text blocks. The FeatureExtractor, introduced by Debnath et al. (2005a), was devised to address this issue and assist in identifying text blocks, image blocks, or navigational blocks.

2.5. Proposed HTML Block Comparison Method and its Suitability Evaluation

At its core, the proposed method uses the Tree Edit Distance (TED) algorithm, which was initially proposed by Shasha et al. (1990). Because HTML can be represented as a tree structure, the TED algorithm fits very well for analysing trees built from HTML blocks. The Tree Edit Distance (TED) algorithm operates on a simple principle: it takes two trees as input for comparison and then generates multiple metrics based on the operations required to render the trees identical. These operations include node creation, node removal, and node updating. Each performed operation can have its own cost, which may be helpful for similarity estimation as it allows for a more detailed analysis of HTML blocks.

2.5.1. HTML Block Comparison Method

Given the structured nature of the documents in question, it is essential to accommodate this characteristic and accordingly adapt the Tree Edit Distance (TED) algorithm to suit these conditions. For instance, adding or removing a node from a lower tree level (closer to the root) should have a lower cost than performing the same operation inside one of the higher levels of the tree. This distinction is crucial because edit operations done at a lower tree level have less impact on the overall structure. However, this may not always be the case and will require extensive testing to confirm. The tested algorithms did not account for this; noting only the addition or removal of a node, without considering the tree level at which this occurred, had no impact on the final similarity estimation result. The final result of the block comparison should be a value between 0 and 1, where 0 means the blocks are entirely dissimilar, and 1 means the blocks are identical.

During the initial phase of development, hand-picked content blocks were used to create scenarios that can be difficult for the algorithm to evaluate when determining block similarity. This strategy facilitated rapid prototyping and testing of different approaches, making it possible to eliminate the ones that did not produce the expected results. This iterative process was used repeatedly to refine the final method.

Initially, block similarity was calculated based on the tag tree structure and the tags utilised. This block similarity estimation does not account for the content stored within the tags. Additionally, each operation has its own cost: node addition and removal is 0.5, as Demaine et al. (2009) suggested, while node modification is 0.75. Furthermore, a depth level coefficient was used so that operations on different tree levels would have different costs; it was set to 0.9.

Block similarity based on tag tree structure and tags used (2.5):

$$B_{sim} = \frac{\sum_1^{T_{count}} L_{score}}{T_{count}}. \quad (2.5)$$

Level score calculation (2.6):

$$L_{score} = \begin{cases} 1, & \text{tags match} \\ \frac{o_{cost} \times L}{L_{cost}}, & \text{tags don't match} \end{cases}. \quad (2.6)$$

Level cost calculation (2.7):

$$L_{cost} = \frac{L}{K^L \times o_{cost}}. \quad (2.7)$$

The block similarity score denoted as B_{sim} , is influenced by several factors. T_{count} represents the highest number of tags between block A and block B, L_{cost} is the cost associated with modifying the level, L represents the depth level of a node,

K is a coefficient that introduces a cost dependency based on the current depth level, and O_{cost} is the cost associated with performing a particular operation.

Moreover, another iteration of the algorithm incorporates the concept of tag interchangeability. Given the fluid nature of HTML, various tags can be utilised to achieve identical outcomes. For instance, DIV tags can be replaced with HEADER, FOOTER, and NAV, among others, and this principle applies to several other tags. Therefore, this aspect was integrated into the similarity estimation by modifying the operation cost calculation. Rather than having operation costs vary based on the type of operation (such as node addition, removal, or modification), a fixed operation cost of 0.5 was applied. This cost was then adjusted based on the tag similarity percentage, reflecting the likelihood that one tag could be substituted for another with the same result. The cost estimation was further refined by halving this probability and adding it to the fixed operation cost.

The calculation of the operation cost, considering tag or text similarity, is expressed in Equation (2.8):

$$O_{cost} = \begin{cases} O_c + \frac{T_{sim}}{2}, & \text{for tags} \\ O_c + \frac{TEXT_{sim}}{2}, & \text{for text} \end{cases} \quad (2.8)$$

Where O_c represents the fixed operation cost, T_{sim} denotes the tag similarity score, quantifying the likelihood that tags can be interchanged while still achieving identical outcomes and $TEXT_{sim}$ is the text similarity score, quantifying the resemblance between the contents within the tags

In addition to comparing the structure of the blocks, it is crucial to perform data analysis, as some blocks may exhibit identical or similar structures yet serve vastly different purposes due to the distinct content within those blocks. This analysis can be integrated as a supplementary step during block examination and factored into the final similarity score. The same HTML structure as previously mentioned was used for validation, ensuring that the content varied across the tested blocks. The Cortical.io API was employed to compute the content similarity. The content of the tags was incorporated into the equation as an additional level within the HTML block, thereby making its score dependent on the depth.

Incorporating content into the equation provides a more comprehensive analysis of HTML block similarity, as it accounts for both structural and content-based similarities. Ultimately, a holistic approach, which combines structural, tag, style, and content analysis, is essential for accurately determining the purpose and similarity of HTML blocks. By employing a multifaceted approach, one can assess the likeness of different HTML blocks more effectively despite potential differences in tag use, content, or structure. This is crucial in the world of web development, where subtle differences in HTML code can lead to significant variations in web page presentation and functionality.

2.5.2. HTML Blocks Similarity Estimation Experiments

The evaluation focused on various open-source algorithms designed to compare HTML structure similarity, aiming to ascertain whether the comparison addressed the actual structure or if it was exclusively between tags at the same tree structure level. Five snippets of the HTML navigation block (Figures 2.9–2.13) were employed to observe how the algorithms respond to different blocks while the tree structure remains consistent.

```

1 <div>
2   <ul>
3     <li><a href="index.html">Home</a></li>
4     <li><a href="news.html">News</a></li>
5     <li><a href="contact.html">Contact</a></li>
6     <li><a href="about.html">About</a></li>
7   </ul>
8 </div>

```

Fig. 2.9. The first (nav1) variant of the HTML Block used in the experiment

```

1 <nav>
2   <ul>
3     <li><a href="index.html">Home</a></li>
4     <li><a href="news.html">News</a></li>
5     <li><a href="contact.html">Contact</a></li>
6     <li><a href="about.html">About</a></li>
7   </ul>
8 </nav>

```

Fig. 2.10. The second (nav2) variant of the HTML Block used in the experiment

```

1 <div>
2   <div>
3     <div><a href="index.html">Home</a></div>
4     <div><a href="news.html">News</a></div>
5     <div><a href="contact.html">Contact</a></div>
6     <div><a href="about.html">About</a></div>
7   </div>
8 </div>

```

Fig. 2.11. The third (nav3) variant of the HTML Block used in the experiment

```

1 <nav>
2   <div>
3     <div><a href="index.html">Home</a></div>
4     <div><a href="news.html">News</a></div>
5     <div><a href="contact.html">Contact</a></div>
6     <div><a href="about.html">About</a></div>
7   </div>
8 </nav>

```

Fig. 2.12. The fourth (nav4) variant of the HTML Block used in the experiment

```

1 <div>
2   <nav>
3     <div><a href="index.html">Home</a></div>
4     <div><a href="news.html">News</a></div>
5     <div><a href="contact.html">Contact</a></div>
6     <div><a href="about.html">About</a></div>
7   </nav>
8 </div>

```

Fig. 2.13. The fifth (nav5) variant of the HTML Block used in the experiment

In the initial set of tests, the HTML's anchor elements remain constant across all blocks, while other surrounding blocks vary. Despite this, the overall DOM tree structure is consistently maintained. A different aspect of the tests includes evaluating the same HTML snippets but with variations in the anchor tag content so that each snippet possesses a unique text and href attribute value for the anchor tag. This aspect of the testing will aid in determining whether any of the tested algorithms are sensitive to changes in content.

The Perl HTML_similarity algorithm presented by Lin (2023) was tested first. This algorithm is relatively uncomplicated: it creates two arrays from the compared HTML snippets, with the arrays designed to represent the DOM tree structures of the snippets, including the tags used. Following this, the similarity between the two snippets is calculated using formula (2.9):

$$B_{sim} = \frac{LCS \times 2}{X_{len} + Y_{len}}. \quad (2.9)$$

The outcome of this formula provides the block similarity score (B_{sim}), with LCS representing the longest common sequence, X_{len} denoting the length of tree A, and Y_{len} representing the length of tree B.

Subsequently, the Python HTML_Similarity (Marca, 2023) and Python Page_Compare (Haase, 2023) algorithms were tested. The Page Compare algorithm is designed to evaluate both structure and style similarities and can generate

a similarity score that amalgamates both aspects. All HTML snippets were compared during the testing phase to evaluate their estimated similarities.

An interesting observation was made during the testing: using entirely different tags (Figure 2.14) while preserving the same structure could yield a 50% similarity score. For instance, comparing the given code snippets would result in a structural similarity score of 50%. However, the resulting score was slightly lower when testing the same structure using the proposed algorithm, which considers tag similarity, at 47.61%.

```
1 <header>
2   <span></span>
3 </header>
1 <div>
2   <a></a>
3 </div>
```

Fig. 2.14. Example of two HTML blocks that share the same tree structure but have different functionality/purpose due to used tags

This outcome is quite encouraging because identical structures can be created using varying tags. It underscores the necessity of employing multiple metrics when comparing HTML blocks, given that the same structure with different HTML tags can lead to entirely distinct purposes for the HTML block. Additionally, the comparison should not be limited to just structure, tags, and styles; the content itself plays a crucial role in determining the block’s purpose.

To estimate the similarity between two content blocks by using different comparison methods, five previously analysed HTML code examples were compared with each other. The table below summarises the comparison for each pair of HTML blocks. The first six columns are for existing similarity estimation methods, while the last three represent the newly proposed method with various features for combined similarity measurement. The similarity scores, which in all cases vary from 0 to 1, indicate that the proposed method’s usage can lead to slightly higher similarity scores. Only in the case of the proposed method with tag similarity estimation and different content was the average similarity score lower than the average scores of the other methods.

To gain more insights additional calculations were performed with Python Page_Compare, the Proposed method and the Proposed method with tag similarity estimation. Python Page_Compare was chosen since all of the available methods had 0.18 standard deviation and the most common average score was 0.71, except Perl HTML::Similarity which failed to produce a score for one of the tests.

By applying the Wilcoxon Signed-Rank test to compare the Python Page_Compare case (average similarity score of 0.71, the standard deviation of 0.18) and the proposed method case (average similarity score of 0.79, the standard deviation of 0.13), the Z-value is -2.7011, the p-value is 0.00694, the W-value is 1, the critical value for W at N = 10 (p < 0.05) is 8. The same calculations were

also performed with the modified version of the proposed method that accounts for tag similarity (average similarity score of 0.81, the standard deviation of 0.14), in this case, the Z-value is -2.8031, the p-value is 0.00512, the W-value is 0, and the critical value for W at $N = 10$ ($p < 0.05$) is 8. These results indicate the proposed methods achieve statistically different results. The results would be a statistically significant event if a 0.01 significance level is used. Considering that the achieved average similarity score for the proposed methods is higher than the existing ones and all the cases were similar in content, the proposed approach can be assumed as a good direction towards HTML block pairwise similarity comparison.

Table 2.2. Comparison of existing and newly proposed HTML similarity estimation methods with all combinations of selected example code (Fig. 2.9–2.13)

	Perl HTML::Similarity	Perl HTML::Similarity different content	Python HTML_Similarity	Python HTML_Similarity different content	Python Page_Compare	Python Page_Compare different content	Proposed method	Proposed method with tag similarity estimation	Proposed method with tag similarity estimation and different
nav1+nav2	0.96	0.96	0.92	0.92	0.92	0.92	0.95	0.97	0.75
nav1+nav3	0.62	0.62	0.58	0.58	0.58	0.58	0.71	0.71	0.56
nav1+nav4	0.64	0.64	0.58	0.58	0.58	0.58	0.66	0.68	0.54
nav1+nav5	0.64	0.64	0.58	0.58	0.58	0.58	0.71	0.73	0.57
nav2+nav3	0.56	0.56	0.50	0.50	0.50	0.50	0.66	0.68	0.54
nav2+nav4	0.58	0.58	0.58	0.58	0.58	0.58	0.71	0.71	0.56
nav2+nav5	0.58	0.58	0.58	0.58	0.58	0.58	0.66	0.70	0.55
nav3+nav4	0.96	0.96	0.92	0.92	0.92	0.92	0.95	0.97	0.75
nav3+nav5	0.96	0.96	0.92	0.92	0.92	0.92	0.95	0.97	0.74
nav4+nav5	-	-	0.92	0.92	0.92	0.92	0.90	0.94	0.74
Average	0.65	0.65	0.71	0.71	0.71	0.71	0.79	0.81	0.63
Standard deviation	0.18	0.18	0.18	0.18	0.18	0.18	0.13	0.14	0.10

For all existing methods, the standard deviation is 0.18, while for the proposed methods it is 0.14 or less. This indicates that the similarity scores of the proposed methods are more stable and not so skewed. In the case of the proposed method with tag similarity estimation and different content, the standard deviation is almost two times smaller than in the case of the existing methods.

2.5.3. Evaluation of the Results and Possible Improvements

During the testing phase, it was observed that the examined algorithms did not consider depth when calculating the operation cost for tree modification. The similarity scores witnessed a marginal increase after incorporating depth level into the operation cost. This is a significant observation, as modifications made at higher levels can have a more pronounced impact on the HTML block and vice versa. Moreover, there was an additional enhancement in the similarity scores when the estimation was performed while considering tag interchangeability. This is crucial because introducing many semantic tags in HTML 5 means that HTML blocks with the same purpose can employ different tags. Therefore, it is essential to consider this when estimating block similarity. To refine this facet of similarity estimation, a comprehensive database of tag pairs and their interchangeability scores is planned to be developed.

Additionally, as anticipated, testing for content similarity resulted in marginally lower similarity scores with a 15–20% reduction. However, depending on the content, the score could be higher than the score obtained without content comparison. This variance is mainly dependent on the amount of content. The blocks tested had minimal content, just a single word, which generally resulted in a very low content similarity score. However, it was observed that as the content increases, the score can improve, thereby enhancing the overall score.

In summary, the testing phase revealed crucial insights that will be instrumental in refining the algorithm for HTML block similarity estimation. Accounting for depth, tag interchangeability, and content are key factors that influence the similarity score. For the algorithm to perform its calculations, it needs to be supplied with HTML blocks that must be compared. This process needs to be automated due to the large number of blocks on the websites. Also, after the comparison is performed, the data needs to be systematically stored to be available for reuse or further analysis. All these requirements result in the need for a dataset that would allow storage of such data so that it is not lost after analysis is performed and is also easily accessible on demand. Access to such data may be performed in various ways, either by the block's purpose if the block is identified or by the block's similarity score.

2.6. Conclusions of the Second Chapter

1. Analysis of existing HTML-related block similarity methods revealed a multi-dimensional problem. It became evident that a comprehensive content block similarity score should consider content, structure, and design properties. Combining these properties and similarity scores should produce a single combined similarity score for general-purpose comparison.
2. The proposed new method of similarity estimation offers multiple variations of how the combined similarity score is estimated and which metrics are used during the estimation.
3. The proposed method's testing showed better stability than existing solutions, as indicated by standard deviation values across all proposed method variations (ranging from 0.10 to 0.14, in contrast to the 0.18 standard deviation observed in other methods). The average similarity score is also higher (it reaches 0.81, while the maximum value of existing methods was 0.71). The difference between averages is not significant ($p = 0.18$).
4. The obtained results indicate that using multi-criteria HTML block similarity estimation produces a more accurate similarity score.

Multi-Purpose Dataset of Webpages and its Content Blocks

The following chapter focuses on creating a comprehensive dataset to aid in efficiently collecting and processing data from websites. This dataset is specifically designed to address the challenges in identifying and extracting relevant content blocks from web pages, supporting a range of data extraction methods and machine learning applications.

The dataset captures the complexity of modern websites, which often contain a mix of primary and secondary information. It includes a structure that archives a website's data, code snippets, and visual representations of each HTML node.

An experimental evaluation assessed the dataset's structure and ability to accommodate various labelling approaches. The experiment involved manual segmentation and labelling of web pages, resulting in a comprehensive collection of labelled blocks. The results highlighted the necessity for a standardised approach to labelling, as about half of the unique blocks received multiple labels.

Enhancements to the dataset structure have been made based on research insights, including fields for clean text and unique CSS selectors and additional tables for block types and text comparison.

The main research results of this chapter were published in one author's scientific publication (Griazev & Ramanauskaitė, 2021).

3.1. Proposed Structure of Dataset for Websites and Their Content Blocks

The rapid growth of digital information necessitates automated data exchange systems for efficient data collection and processing. Websites are one such rich source of publicly available data. However, modern websites often contain extraneous information that can clutter data collection. To improve and develop tools that can be used to extract valuable data from websites, a multi-purpose dataset of web pages and content blocks is needed to facilitate website content extraction solutions. The proposed dataset aims to solve various problems in content block identification and also allows for the use of machine learning applications.

Contemporary websites frequently incorporate extraneous content, which might dilute primary information. Strategically filtering out these redundant elements from inception streamlines the focus on pertinent information. Numerous methodologies exist for this extraction process. Some techniques analyse the intrinsic data, others probe the site's structural layout or visual aspects, and a subset combines multiple approaches (Ferrara et al., 2014; Laender et al., 2002). Each technique exhibits unique advantages and challenges, with none presenting an absolute solution, as Al-Ghuribi et al. (2013) suggested in their survey. Moreover, the heterogeneity in the data sources and testing environments further complicates the direct comparison of these methodologies.

A multifaceted examination of the website content, considering various parameters, allows for determining the content block's purpose and type. The latest tools for web content analysis predominantly rely on supervised learning, necessitating labelled datasets for effective implementation. Such datasets are invaluable, serving both developmental and research-oriented functions.

Diverse research objectives demand different datasets. Some methodologies examine the site's source code, others scrutinise its stylistic elements, while some require a visual representation of the site (Al-Ghuribi et al., 2013; Utu et al., 2018). This variability in methodology introduces challenges when comparing results across diverse techniques (Debnath et al., 2005b; Lin et al., 2002). There is a compelling requirement for a multi-purpose dataset catering to a broad spectrum of methodologies, particularly beneficial for machine learning applications given their need for extensive data.

The proposed novel webpage dataset design is optimised for content block annotation. This dataset solves the issue of representation of diverse data types and introduces the idea of block variations while also storing detailed hierarchical data of content blocks.

The suggested novel dataset structure can archive a website's data and its content blocks. This proposal stems from carefully examining the current methods for identifying website content blocks and the limitations of existing datasets, as

provided that can be applied based on a given situation's specific requirements and challenges.

The dataset's structure includes block types, which are basic web page elements like menus, menu items, articles, headings, etc. There are over 70 block types, but this list is incomplete. It can be expanded and modified to reflect the evolving trends in website development.

Block variations are also considered, especially when the same HTML structure can represent multiple block types. For example, an article heading can also function as a link. In such cases, two entries (heading and link) will be saved in the dataset as type variations.

The hierarchical context in which each dataset entry occurs in an HTML document is also essential. The labelling of a block can change depending on its position in the hierarchy. Therefore, each dataset entry is associated with a parent block, which applies to all block variations.

Block labelling can be done by analysing the code or evaluating visual features. If all web page assets are available, a screenshot of a web page can be generated. However, this can be complicated if some sources are remote, such sources can become unavailable or modified. For this reason, a node screenshot was included with each block entry to facilitate the visual approach applications. This ensures the stability of the dataset and simplifies its use.

Some HTML blocks are only visible at specific screen sizes. Therefore, three screenshots were included for each HTML block, representing desktop, tablet, and mobile viewports. The chosen viewports were based on the most popular resolutions currently listed on statcounter.com – 1366 px for desktops, 768 px for tablets, and 360 px for phones.

Figure 3.1 illustrates the proposed dataset structure. A relational database was chosen for the implementation of the dataset. The dataset becomes valuable when it has a well-designed structure, a large number of records, and a functional user interface. Although the dataset is currently populated with data, it will be made available as a service rather than as a separate text-based file or an archive of files. This approach will allow users to generate data formats and structures tailored to their specific needs. Users can specify the required data and the amount needed, such as limiting the number of web pages or content block types.

The primary data is stored in a single table called the "blocks" table (Figure 3.1). This table holds vital information about the HTML blocks present on a page, encompassing the hierarchy (parent of the block) and the order of the block within its parent. Additionally, it contains the block's code, its type, an indication of whether the block is a variation of another (Figure 3.2), and the id of the source file. Every block is assigned a predefined type, all of which are contained in the "block_types" table. These block types are organised hierarchically, allowing for

the existence of multiple parent blocks. The “block_type_hierarchy” pivot table facilitates the creation of a hierarchy between parent and child block types.



Fig. 3.2. Example of HTML block variations where red blocks can be considered containers and all three cases can be assumed as correct bounds of the block

The “Attachments” table stores information about the different assets included in the dataset, such as the source files of HTML pages and source archives, which comprise images, CSS, scripts, and other website assets. Additionally, it stores full-page screenshots and individual node screenshots in three different sizes (desktop, tablet, and phone). Due to the requirement of multiple screenshot files for each HTML node, an “attachment_block” pivot table is also present. Each attachment is associated with a predefined type listed in the “attachment_types” table. This list of attachment types enhances the dataset’s flexibility by allowing the addition of new types and restricts the user to selecting only from the predefined attachment types.

The “sources” table holds information about the pages used as sources for the HTML data. It enables the storage of complete HTML and all associated assets (images, CSS, JavaScript, etc.) in an archive format. This feature is handy if the live web page changes, ensuring the original code remains accessible. Moreover, a screenshot of the entire page is saved. If the data originates from an inner page, the parent page is specified, enabling the construction of a hierarchical sitemap of the analysed website.

The evaluation of tag similarity is crucial for determining whether blocks have analogous structures but utilise different tags to present identical types of

information. If the blocks are found to be similar, the blocks can be assigned the same label. To facilitate this, two tables are employed: “tags”, which stores HTML tags, and “tag_similarity”, which stores a value representing the estimated similarity between two tags. A higher value indicates a greater likelihood that the tags can substitute for one another.

3.2. Research on the Proposed Dataset Structure Suitability and Its Data Peculiarities

Moving forward, the focus will be on a detailed evaluation of the dataset structure’s completeness and suitability for storing various combinations of data and data relations. This evaluation enables the examination of the proposed dataset’s ability to accommodate the necessary data-storing features for effective web page segmentation and labelling, thereby allowing for improvements in the dataset structure.

3.2.1. Dataset Structure Completeness Evaluation

The completeness of the proposed data structure was assessed by analysing the data required for existing web page segmentation and labelling solutions. The necessary data were mapped to the proposed dataset’s structure using queries. Table 3.1 showcases the web page segmentation solutions analysed in the related work chapter. Each segmentation method is accompanied by a summary of the data it utilised as input to execute web page segmentation tasks and the output that would be generated. It was necessary to extract input and output values from the dataset to carry out the segmentation task and validate the results by comparing them with the dataset data. The ability to extract the required data was illustrated as a query, which can be employed to retrieve the necessary data from the proposed dataset structure.

The comparison indicated that all necessary data for existing web page segmentation solutions could be extracted from the proposed dataset structure. Using a custom aggregation function during dataset data extraction provides additional flexibility. Researchers can obtain either the minimum or maximum block boundary or choose to use the most common block boundary. The aggregation function will reduce the number of block variations and provide only one block boundary result. However, researchers can also extract multiple variations of the same block and consider them during web page segmentation.

Table 3.1. Summary of proposed dataset suitability for existing web page segmentation and labelling solutions

Web page Segmentation solution	Input data	Output	Relational Algebra Expression to Extract Needed Data from the Proposed Dataset Structure
Bing et al. (2014)	Web page DOM tree	Block bounds	$\pi_{s.id, sc.code, AGR(bc.code)} \sigma (\rho_s (\text{Source}) \bowtie_{s.id = sc.source_id \text{ AND } sc.parent_block_id \text{ is NULL}} \rho_{sc} (\text{Blocks}) \bowtie_{sc.id = bc.parent_id \text{ AND } bc.parent_block_id \text{ is not NULL}} \rho_{bc} (\text{Blocks}))$
Vargas (2015)	Web page DOM tree	Block bounds	
Akpınar et al. (2013)	Web page DOM tree and image	Block bounds	$\pi_{s.id, sc.code, AGR(bc.code), at.filename} \sigma (\rho_s (\text{Source}) \bowtie_{s.screenshot_id = at.id} \rho_{at} (\text{Attachments}) \bowtie_{s.id = sc.source_id \text{ AND } sc.parent_block_id \text{ is NULL}} \rho_{sc} (\text{Blocks}) \bowtie_{sc.id = bc.parent_id \text{ AND } bc.parent_block_id \text{ is not NULL}} \rho_{bc} (\text{Blocks}))$
Nguyen et al. (2012)	Web page DOM tree and image	Block bounds and type	$\pi_{s.id, sc.code, AGR(bc.code), at.filename, bt.title} \sigma (\rho_s (\text{Source}) \bowtie_{s.screenshot_id = at.id} \rho_{at} (\text{Attachments}) \bowtie_{s.id = sc.source_id \text{ AND } sc.parent_block_id \text{ is NULL}} \rho_{sc} (\text{Blocks}) \bowtie_{sc.id = bc.parent_id \text{ AND } bc.parent_block_id \text{ is not NULL}} \rho_{bc} (\text{Blocks}) \bowtie_{bc.block_type_id = bt.id} \rho_{bt} (\text{Block_types}))$
Zhang et al. (2013)	Web page DOM tree and image	Block bounds and type	
Burget et al. (2009)	Web page DOM tree, CSS code, and image	Block bounds and type	$\pi_{s.id, sc.code, AGR(bc.code), at.filename, bt.title, css.file-name} \sigma (\rho_s (\text{Source}) \bowtie_{s.screenshot_id = at.id} \rho_{at} (\text{Attachments}) \bowtie_{s.archive_id = css.id \text{ AND } css.attachment_type_id = "css"} \rho_{css} (\text{Attachments}) \bowtie_{s.id = sc.source_id \text{ AND } sc.parent_block_id \text{ is NULL}} \rho_{sc} (\text{Blocks}) \bowtie_{sc.id = bc.parent_id \text{ AND } bc.parent_block_id \text{ is not NULL}} \rho_{bc} (\text{Blocks}) \bowtie_{bc.block_type_id = bt.id} \rho_{bt} (\text{Block_types}))$
Feng et al. (2016)	Web page DOM tree, CSS code, and image	Block bounds and type	

AGR() is an aggregation function. Depending on the situation, it can be minimum, maximum, or most frequent block bounds.

Another significant feature of the proposed dataset structure is block hierarchy. Some of the analysed web page segmentation solutions consider the hierarchy of segmented blocks. This hierarchy information is gathered or calculated during the segmentation process. Since the proposed dataset structure stores hierarchy data for each block, it can be utilised to reduce data processing time.

3.2.2. Research Methodology for Manual Web Page Segmentation

The suitability of a dataset is evaluated not only by its data coverage but also by understanding how new data should be added to the dataset and the specifics of such data. To account for this, an additional experiment was designed to refine the situations when web page content block variations occur while also evaluating the suitability of the proposed dataset structure to store all block variations.

Manual web page segmentation and labelling were employed to represent real-world websites in the dataset. Using synthetic, auto-generated websites and their blocks would not guarantee existing web pages' natural variety and variations.

During the experiment, the content blocks of selected websites were labelled. This block labelling data enabled the evaluation of whether there were any redundant or missing elements in the proposed dataset schema. Since this kind of evaluation can be influenced by an unconscious bias towards the designed dataset structure, additional respondents were gathered for the experiment. Six respondents labelled the same set of websites independently from each other. This also facilitated the comparison between the block labels pre-selected by the authors and the labels created by the respondents, as illustrated in Figure 3.3. This helped refine the pre-set labels used when populating the dataset with data.

The research participants, a group of developers with varied experience levels from junior to senior, were tasked to label different blocks of six one-page websites. These websites were carefully chosen to represent a broad spectrum of information domains: three service-providing company websites, one personal website offering services, one restaurant website, and one mobile app promotion website. The selection criteria for these websites were based on the diversity of block types contained and the variety of visual designs used for similar blocks, thereby enabling a comprehensive assessment of the block labelling approaches adopted by the participants. Each participant was presented with the websites randomly to minimise any order effects.

The task required the participants to create labels for the blocks without any predefined labels provided to them. This approach ensured that the labelling was based on the participants' interpretation of the blocks, which could vary from visual features to the information contained within the blocks or a combination of

both. The labels generated by the participants were stored separately for subsequent analysis.

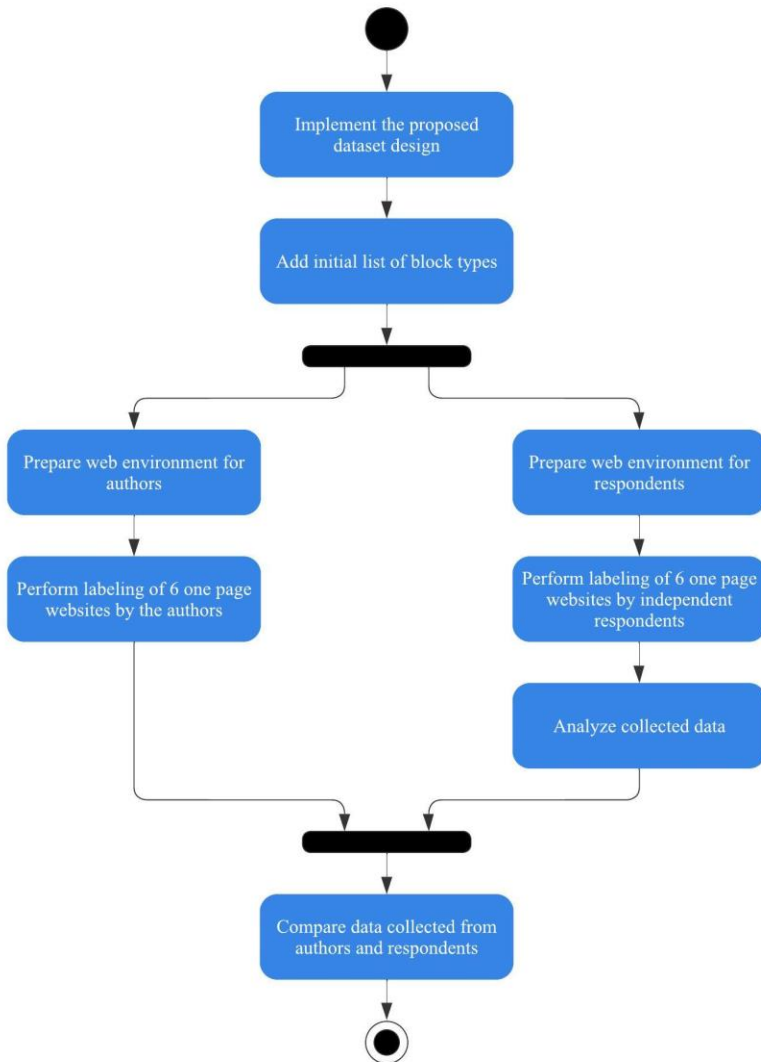


Fig. 3.3. Basic schema of the experiment for the dataset schema validation

Following the completion of the experiment, the collected labels from the participants were compared with the labels created by the authors of the research. A detailed representation of the experiment's design can be found in Figure 3.3.

3.2.3. Analysis of Labelling Experiment Results

The experiment involved labelling a total of 905 distinct blocks. On average, each page contained 82 unique blocks eligible for labelling. A block was considered unique based on its function or purpose. During the labelling process, when multiple adjacent blocks of the same type were labelled, only one was counted as unique, and the rest were excluded from the experiment statistics to minimise duplicate labelled blocks within a page. For instance, consider a navigation menu block, as illustrated in Figure 3.4, labelled as block No. 3. This block comprises six menu items (home, about, services, portfolio, blog, and contact). Although the text and link of each item differ, all serve the same function and, therefore, can be labelled identically in this context.

Nevertheless, a more detailed analysis of these blocks could be conducted later, as some metadata may not be immediately visible to the user but are present in the HTML code. For example, menu items may contain additional attributes that provide more detailed descriptions of the linked content. It is important to note that the exclusion of duplicate blocks was only implemented during the experiment to validate the dataset structure. When populating the dataset with actual data, all labelled blocks are included to ensure comprehensive labelling of the entire page.



Fig. 3.4. An example of duplicated block ignorance is when block No. 4 represents any menu item of the menu (block No. 3) rather than labelling all menu items

The experimental data evaluation observed that participants often used different terms to label identical blocks, yet the assigned labels were usually synonymous. For instance, in Figure 3.4, block No. 1 was labelled by participants as “navbar”, “header”, and “top navbar”. Similarly, block No. 3 received labels such as “main nav”, “navbar”, “main menu”, “nav”, and “navigation”. In both cases, the assigned labels can be considered synonyms, except for labelling block No. 3 as “navbar”, which was not entirely accurate, though it still roughly indicated the block’s function. This lack of uniformity in terminology is expected due to the absence of a universally accepted nomenclature for website elements, compounded by the myriad ways websites can be styled.

Nevertheless, the experiment demonstrated that the proposed dataset structure is robust enough to accommodate the issue of multiple synonymous labels being assigned to a single block. The structure allows for storing multiple variations of the same block, which is beneficial for website content block analysis. It

enables the analysis of synonymous labels for a block to determine the most suitable name for each block type.

The experiment also highlighted several instances where the ability to store block variations was advantageous. The need for block variations arose from differences in block sizes and data aggregation levels. Some variations differed in content length but were labelled the same. For example, in Figure 3.5, three blocks are depicted: No. 7 contained the text “Hello, I’m”, No. 8 included “Mark Parker”, and No. 80 encompassed both texts. Some participants labelled block 80 as a “heading” or “title”, thereby combining blocks 7 and 8, while others labelled block 7 as a “pre-title” and block 8 as a “title” or “heading”. The proposed dataset structure accommodated the labelled data in both scenarios, facilitating the mapping of block relationships based on their purpose (label) or content.



Fig. 3.5. Example of block variation

The collected data (Table 3.2) indicates a need for more consensus in the block labelling, with nearly 50% of unique blocks receiving at least two different labels. This trend also showed that as the number of labelled blocks increased, so did the percentage of variations, highlighting a significant degree of subjectivity in block labelling. These findings underscore the necessity of having the capability to store multiple variations of blocks in the dataset.

The survey yielded 905 labelled blocks, of which 383 were unique, indicating that respondents assigned different labels to the same area. Of these unique blocks, 192 had only one label, constituting about 50% of unique blocks, and can be considered easily identifiable and unambiguous. The remaining 50% had at least two different labels; around 18% had three or more, ~9% had four or more, ~3% had five or more, and <1% had six different labels. Given that six different individuals labelled the same websites, three or more additional labels for the same content block suggest that 18% of the blocks needed to be presented or had an ambiguous purpose and title.

A subsequent analysis was conducted to ascertain whether these variations were synonymous. Block labels were examined per block and compared against the authors' labels for the same block. Synonymous labels would validate the authors' labels, while nonsynonymous labels would indicate a need for modification or adjustment to reflect the block type accurately. The number of blocks with nonsynonymous variations differed significantly across the websites presented in the survey despite their similar content structure. This suggests that the variations are

influenced by the visual styling of a website, with more complex designs leading to greater block variety. On average, each block had 2.3 nonsynonymous variations (0.6186 std.), typically two or three per block, though there were outliers. Seven blocks had four nonsynonymous variations, the highest observed in the survey, and no block had more than five.

Table 3.2. Website labelling results obtained from respondents during the experiment

Website	No. of La- belled Blocks	No. of Unique La- belled Blocks	No. of Blocks with Two or More Variations	No. of Unique Blocks with Nonsynonymous Variations	Average Non- synonymous Variations count (Unique per Block)
Restaurant	223	85	44 (51.76%)	28 (32.94%)	2.32 (65 total)
Company 1	141	49	33 (67.34%)	23 (46.93%)	2.26 (52 total)
Company 2	115	54	21 (38.88%)	12 (22.22%)	2.33 (28 total)
Company 3	151	60	35 (58.33%)	9 (15%)	2.33 (21 total)
Personal	188	73	45 (61.64%)	13 (17.8%)	2.15 (28 total)
Phone App	87	62	13 (20.96%)	5 (8.06%)	2.40 (12 total)
Total	905	383	191 (49.86%)	90 (23.49%)	2.28 (206 total)

Analysing the collected data revealed insights into the naming conventions respondents applied to block labelling, which is essential for assessing alignment with the authors' proposed labels during dataset testing (75.41% match). Generally, the respondents' labels were either identical or synonymous with those used during dataset testing, confirming that the initial block types selected for labelling were appropriate and comprehensible to other users. However, some block types collected during the experiment were data-focused, describing the data rather than its purpose. This is unsuitable for the dataset as labels should be abstracted from the content. For instance, on the restaurant website, some labels contained dish names instead of being abstract (e.g., "egg sandwich" instead of "menu entry" or "dish name/title"). Tying block labels too closely to the data restricts the

reusability of such blocks. Data-focused labels can contaminate the labelling process, resulting in mislabelling blocks with similar purposes but different data.

3.3. Evolution of the Dataset Structure Based on Further Research Insights

The original dataset structure was designed to meet anticipated requirements, effectively storing various data types and ensuring compatibility with multiple datasets. It also aligned well with different data extraction techniques. As the research progressed, as described in subsequent chapters of this dissertation, the need arose to explore new methods for linking data within web pages. This led to a series of enhancements to the dataset structure, depicted in Figure 3.6.

A significant modification was the removal of the `tag_similarity` table. It became clear that its usefulness was more evident in larger datasets and not as necessary during the initial research phase, which focused on the interconnections between data within the same webpage.

Refinements to the `blocks` table included the addition of the `clean_text` and `unique_selector` fields. The `clean_text` field is intended for efficient text analysis, separating the textual content from a block's code for easy access. The `unique_selector` field maintains a CSS selector that precisely identifies a specific block, aiding in its recognition on a webpage.

Several tables were incorporated into the schema in preparation for new features, which will be discussed later in the dissertation. The `block_block_type` table was introduced to address the possibility of blocks correlating with multiple types, enhancing the accuracy of block categorisation. The `block_length_variations` table was also created to track the differences in block sizes, contributing to a deeper understanding of content block dimensions.

Text comparison is now facilitated by two new tables: `levenshtein_similarity` and `text_similarity`. These tables are dedicated to storing results from distinct text comparison algorithms, with one focusing on Levenshtein distance metrics and the other on Ian Oliver's text similarity estimation approach.

To define the relationships between blocks, three new tables were added. The `structural_siblings` table links blocks that share a parent. For blocks without a common parent, the relationships are stored in two forms: `remote_structural_siblings` for blocks with structurally similar tags and `remote_loose_structural_siblings`, where the similarity also encompasses the classes attached to the tags.

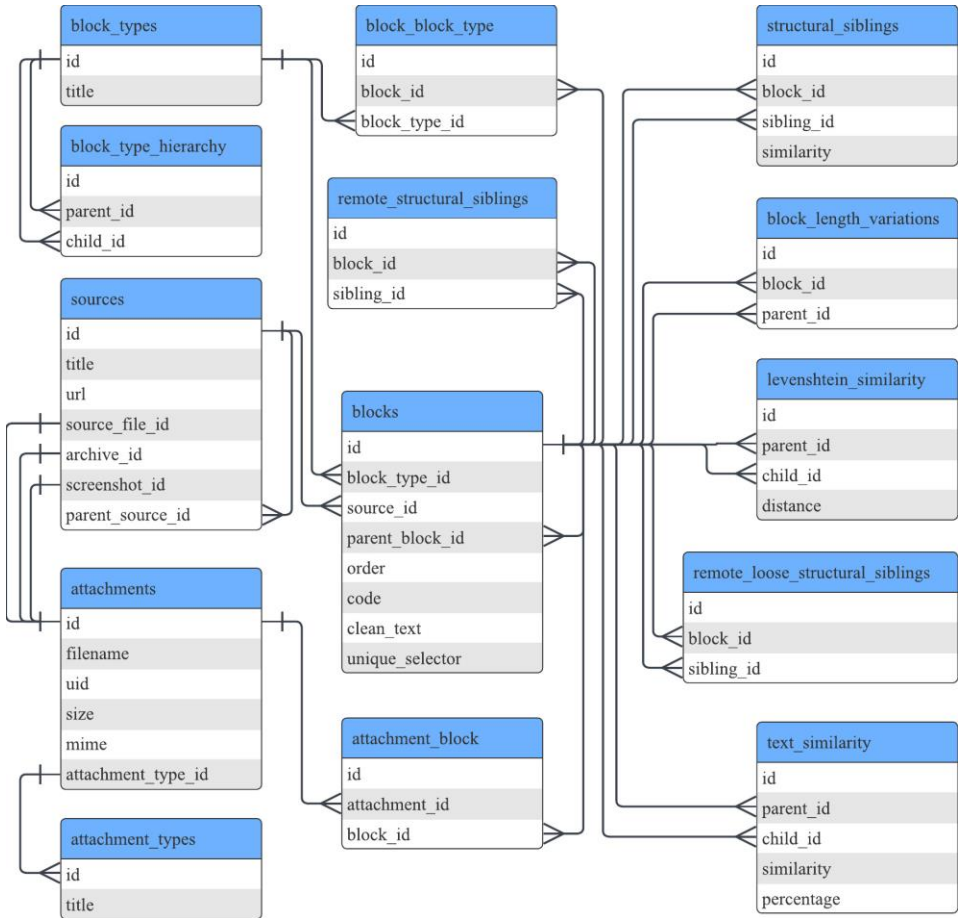


Fig. 3.6. Enhanced dataset data structure schema

These updates have significantly increased the flexibility and analytical capabilities of the dataset, laying the groundwork for the complex analyses discussed in the upcoming chapters.

3.4. Conclusions of the Third Chapter

Data extraction continues to pose a significant challenge. Even though various methods have been devised to overcome this issue, evaluating and comparing effectiveness becomes difficult when diverse approaches are utilised. This issue arises from the nature of the datasets used to test various data extraction methods.

Often, these datasets contain only the data required for the tested method, excluding additional data that other methods might utilise.

To address this:

1. A new comprehensive dataset structure was introduced. It was designed to offer a wide range of data points that can be leveraged to benchmark or develop different types of algorithms and facilitate a straightforward performance comparison.
2. Initial labelling experiment results suggest that the preliminary dataset structure and the labelling scheme for content blocks align with expectations, indicating that the dataset can be further developed in several stages. A subsequent experiment, mirroring the initial setup but providing predefined labels to participants, was conducted to evaluate the suitability of these labels and determine the necessity for adjustments or additional labels.
3. An experiment successfully demonstrated that the proposed dataset structure is viable and robust enough to handle different labelling variations. The ability to account for synonymous and nonsynonymous variations and the capacity to store block variations provide flexibility and resilience crucial for real-world applications.
4. Experiment data analysis revealed that participants' labelling practices varied significantly when faced with multiple similar blocks. Such inconsistencies necessitated further refinement in the results, aligning clusters of the same block types with the labelled blocks and eliminating duplicates. This refinement process proved labour-intensive as it was manually executed and showed the need to refine the labelling process by filtering out repetitive blocks within block clusters.

Web Page Content Block Identification with Extended Block Properties

This chapter provides detailed research into the methodologies for identifying and labelling content blocks within HTML web pages. This is pivotal for transforming web data into a machine-readable format. Subdivision of web content into smaller, more intricate blocks is addressed. This process is crucial for enhancing content analysis and labelling precision, facilitating more accurate machine interpretation of the data.

Datasets are an important part of the development of automated data processing solutions. When datasets are not readily available, creation is necessary, involving substantial manual effort to collect and label data for the development, testing, and refinement of automated solutions.

The enhancement of labelling efficiency is one of the chapter's main points. The labelling process is streamlined by identifying and eliminating superfluous blocks, substantially reducing labelling complexity. The effectiveness of this approach is theoretical and empirically validated. Findings demonstrate the method's accuracy in segmenting web pages into distinct content blocks. A thorough analysis of data and results is performed, including specifics on the number of tags and the number of content blocks identified.

The main research results of this chapter were published in one author's scientific publication (Griazev & Ramanauskaitė, 2023), and findings of the research were presented at one international conference (*"International Conference on Science & Technology"* (STRA), 2023, Prague, Czech Republic).

Block Identification Method for Block Range and Variation Estimation

Most data on the web are displayed using HTML, which was initially designed for human interpretation. However, as technologies become more inter-linked, it is crucial that machines can interpret the content, as noted by Xie et al. (2022). This necessitates machine-adapted labelling of web content, which is foundational for automation in content extraction, data mining, and content transformation, among other applications covered by Anami et al. (2014). However, the HTML standard is gradually evolving from a focus on presentation to data. Even though specific tags like header, nav, section, article, and figure are introduced for content organisation, it is common for such tags to coexist with general-purpose tags to achieve the intended design. Consequently, the automation of web page content block identification is a pertinent task that needs to supplant manual efforts.

Cheng et al. (2019) suggested that automated data extraction from web pages involves two steps after obtaining the HTML code. Hashemi (2020) further describes this approach, noting that first, the HTML code is segmented into content blocks, and then, each block is categorised into predefined types using intelligent solutions. Abbasi et al. (2022) also adopted such an approach. However, it is noted that both stages currently encounter challenges due to the web page blocks' hierarchical nature:

1. Some content blocks may be further subdivided into smaller internal blocks. For instance, a menu may contain menu items, and an article paragraph may include highlighted words or phrases. While the former division is useful, the latter, related to text formatting, is often unnecessary and leads to block redundancy.
2. The presentation structure of a content block may result in varying HTML code ranges. If several tags surround the block content, different selectors can gather it, causing variations in the segment and its content path. Hence, knowledge of potential ranges is crucial for more accurate block classification.
3. The hierarchical and sibling relationships between blocks can positively influence the accuracy of block classification. For example, navigation menus and lists have a hierarchical structure, so maintaining the links between blocks provides more features for accurate block type identification. Simultaneously, identifying relations between blocks may reduce

the dataset size without data loss. Only one sibling element needs a label, while the others can be identified as sibling blocks of the same type.

These challenges and other limitations contribute to the need for a solution capable of automatically extracting data from websites with unknown structures and associating it with the appropriate content type. Some of these issues were also described by Escalona (2019). Current solutions necessitate a predefined website structure or only extract specific data, such as listed items, links, or other blocks.

To overcome some of the mentioned challenges, the goal is to introduce a novel approach to web page block identification that offers a broader range of content blocks and more detailed information about them than existing web page segmentation solutions. Each identified block should reflect a specific structural element of the web page (e.g., menu, header, title, contacts, and paragraph). The method should identify all content blocks, including their internal structure, and provide more detailed block data than traditional web page segmentation solutions, such as block ranges, hierarchical relations, and siblings. This comprehensive content block information would enhance the capabilities of the segmentation data application and lead to improved website content block classification.

4.1.1. Definition of Block Range and Variation

HTML code is organised hierarchically, meaning a single tag can encompass other tags representing smaller parts of the overarching tag. Figure 4.1 illustrates this hierarchical relationship using “menu” elements. These elements, which can be visually identified on a website, consist of smaller sub-components called “menu items”. Items are organised based on the hierarchy of menu elements and are viewed as siblings due to being contained within the same block and sharing similar structural attributes.



Fig. 4.1. Example of a block (menu) with three hierarchically related inner blocks (menu items)

Various combinations of tags can be used to create a menu with multiple items. Figure 4.2 presents one such example, showing how different tags (<header>, <div>, or <nav>) can be used to define the boundaries of the menu while still visually creating the same block. Despite the potential differences in

the boundaries of the menu, users will always perceive it as a menu. However, in specific scenarios, such as machine learning or block structure matching, using specific or all available tags can impact the desired outcome. To enhance the properties of machine-oriented block segmentation, it is necessary to define the range (maximum and minimum boundaries) of HTML blocks.

```

1 <header><!-- B1: maximum boundary start -->
2   <div class="container"><!-- B1: length variation start -->
3     <section><!-- B1: minimum boundary start -->
4       <div><!-- B1.1: maximum/minimum boundary start -->
5         <!-- B1.1.1: maximum/minimum boundary start/end -->
6       </div><!-- B1.1: maximum/minimum boundary end -->
7     <nav><!-- B1.2: maximum boundary start -->
8       <ul><!-- B1.2: minimum boundary start -->
9         <li><!-- B1.2.1: maximum boundary start -->
10          <a href="page1.html">Menu item 1</a><!-- B1.2.1: minimum boundary start/end -->
11        </li><!-- B1.2.1: maximum boundary end -->
12        <li><!-- B1.2.2: maximum boundary start -->
13          <a href="page2.html">Menu item 2</a><!-- B1.2.2: minimum boundary start/end -->
14        </li><!-- B1.2.2: maximum boundary end -->
15        <li><!-- B1.2.3: maximum boundary start -->
16          <a href="page3.html">Menu item 3</a><!-- B1.2.3: minimum boundary start/end -->
17        </li><!-- B1.2.3: maximum boundary end -->
18      </ul><!-- B1.2: minimum boundary end -->
19    </nav><!-- B1.2: maximum boundary end -->
20  </section><!-- B1: minimum boundary end -->
21 </div><!-- B1: length variation end -->
22 </header><!-- B1: maximum boundary end -->

```

Fig. 4.2. HTML code example illustrating the range between the maximum and minimum boundaries of the blocks

The “maximum boundary” is the tag (or element in the DOM tree) that defines the largest possible area, which only includes the block’s content. In the DOM tree, this would be the highest element, which includes only the block’s content and child blocks, excluding siblings. On the other hand, the “minimum boundary” is the tag that results in a child block containing only a portion of the block’s content rather than the entire content. This would be the final element in the DOM tree before the child elements become visible.

In the example provided in Figure 4.2, the <section> tag includes both the image and the navigation items, making it the minimal boundary of block B1. As a result, block B1.1 has identical maximum and minimum boundaries. Going deeper than the <div> would reveal the content itself (the image) while going wider would include the content of another sibling block. The area between the minimum and maximum boundaries of each block is the part that does not contain any content.

“Block variations” are similar blocks that belong to the same parent block. In the case of menu items, as shown in Figure 4.2, the three menu items are variations of each other. However, not all child tags are variations of one another. For

example, blocks B1.1 and B1.2 are siblings, but those blocks are not variations because these blocks have entirely different structures. Whether siblings are considered variations depends on their structural similarity. Variants must share the same parent block and internal structure but not the content. In the example in Figure 4.2, the path of tags for all B1.2 child elements is the same (`<header>`, `<div>`, `<section>`, `<nav>`, `<ul>`, `<li>`, `<a>`), while the path between B1.1 and B1.2 is different. The level of similarity can be adjusted based on the web page type or segmentation requirements.

It is worth noting that the differentiation between block variations and siblings is crucial for accurate web page segmentation. Mistaking siblings for variations or vice versa can lead to erroneous segmentation and incorrect interpretation of the web page structure. Furthermore, precisely identifying block ranges and variations is fundamental for various applications, such as web scraping, web page analysis, and content extraction. For instance, web scraping tools must accurately identify content blocks to extract relevant information without including unnecessary content or missing essential details. Similarly, web page analysis tools require a precise understanding of the block structure to assess a web page's layout and design accurately.

Correct block range and variation identification are essential for a wide range of applications and form the foundation for accurate web page segmentation. This process involves defining the maximum and minimum boundaries of HTML blocks and accurately identifying block variations and siblings. The complexity of this task underlines the need for robust and sophisticated tools capable of navigating the intricate hierarchical structure of HTML code and ensuring accurate segmentation.

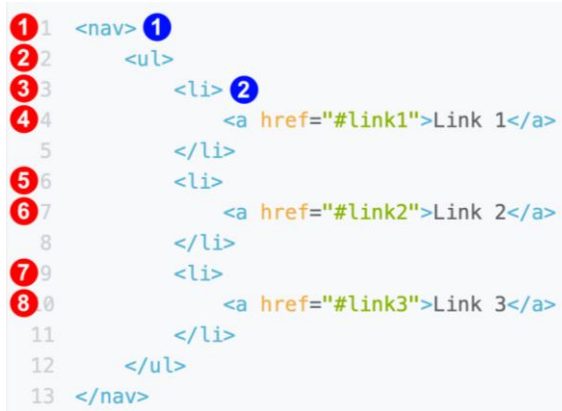
4.1.2. General Idea of Web Page Division to Content Blocks with Extended Properties

Combining the two features mentioned above, e.g., block boundaries and variations, provides increased flexibility in identifying content block data during both manual and automated tagging processes. This added flexibility stems from the ability to reduce the total number of blocks on a page by leveraging the relationships between them. So, when a user manually marks blocks, only one version of similar blocks needs to be labelled, while the rest will be automatically selected. This method requires less effort than marking the entire page.

Figure 4.3 shows an example of how block relationships make a difference. Without block relationships, this single code snippet would have eight separate blocks that must be identified to process this code snippet fully (marked by red circles). To isolate the blocks filled with content, without considering the boundaries, fewer blocks would need to be marked; four would be enough: one for the

menu (red dot No. 1) and three for its items (red dots No. 4, 6, and 8). However, this could lead to results that are harder to check, even if the labelling is correct. This is because, during validation, only the blocks marked exactly would be considered correct, so this approach requires a lot of precision. The number of separate blocks can be significantly reduced using block relationships.

Nevertheless, information about the entire structure is kept, making extracting all data easier. With relationships, only two blocks need labels (indicated by blue circles—one for the menu and one for a menu item). In contrast, all other related blocks would be reachable through hierarchical relations or structure variations.



```

1 1 <nav> 1
2 2 <ul>
3 3 <li> 2
4 4 <a href="#link1">Link 1</a>
5 5 </li>
6 6 <li>
7 7 <a href="#link2">Link 2</a>
8 8 </li>
9 9 <li>
10 10 <a href="#link3">Link 3</a>
11 11 </li>
12 12 </ul>
13 13 </nav>

```

The image shows a snippet of HTML code for a navigation menu. On the left, line numbers 1 through 13 are listed. Next to each line number is a red dot, except for line 1 which has a blue dot. On the right, the corresponding HTML code is shown. The code starts with a <nav> tag on line 1, followed by an opening tag on line 2. Line 3 has an opening tag. Line 4 has an opening Link 1 tag. Line 5 has a closing tag. Line 6 has an opening tag. Line 7 has an opening Link 2 tag. Line 8 has a closing tag. Line 9 has an opening tag. Line 10 has an opening Link 3 tag. Line 11 has a closing tag. Line 12 has a closing tag. Line 13 has a closing </nav> tag. The red dots are placed next to the line numbers 1, 2, 3, 4, 5, 6, 7, and 8. The blue dot is placed next to the line number 1.

Fig. 4.3. HTML code example, illustrating the reduction of labelled blocks when instead of 8 blocks (marked with red dots), a person needs to assign labels to two (marked with blue dots)

To divide web pages into blocks with more properties, the DOM tree is examined from top to bottom, starting with the <body> tag (Algorithm 1). This tag represents the most general block, e.g., web page content. Determining block boundaries will provide extra value, as the minimal boundary will outline where the actual content and internal blocks start. At the same time, the relationship between child blocks will help estimate the repeating structures of variant blocks.

The solution for dividing web pages into blocks was designed to match the dataset data structure described in the previous chapters. It takes a URL address as input and records all the identified blocks and their relationships in the database. The methods for identifying the maximum and minimum boundaries of a block and estimating variations are explained in the following chapters.

This approach not only simplifies the labelling process but also helps maintain the web page's structure, which is essential for applications that rely on a precise understanding of the web page content and structure. Additionally, the process of identifying block relationships and variations plays a crucial role in the

development of machine-learning models for web page segmentation. By reducing the number of blocks that need to be labelled manually, more extensive and more accurate datasets can be created for training these models, ultimately leading to better and more robust applications. Also, this approach makes extracting meaningful content from web pages more straightforward, which is very important for various applications, such as search engines, content aggregators, and web scrapers.

Algorithm 1. Segmentation pseudo code

```

input: DOM tree of the rendered web page HTML code
set analyzed block to <body>
set analyzed parentBlock to <body>
if analyzed block is not empty then
    call boundaryEstimation with block and parentBlock re-
    turn minBlock
    call getChildren with minBlock return children
    call getVariations with children
    for each children
        call segmentation with children
    end
end

```

4.1.3. Method for Block Range Estimation

Identifying the correct range of a content block is critical because even blocks with the same information can have different HTML structures due to various design needs. Sometimes, extra HTML elements are added to achieve a specific visual look. So, it is crucial to accurately figure out a content block's minimum and maximum boundaries.

When HTML is analysed from top to bottom, the maximum block boundary is detected first, and then, a deeper analysis is performed to find the minimum boundary. This is done by comparing content similarities as the HTML tree is processed. For example, in Figure 4.2, the maximum boundary starts at line 1 and goes all the way to the last line. This entire code represents the maximum boundary. On the other hand, the minimum boundary is defined by the <section> element. Comparing the content within these two boundaries reveals that the user perceives the same content in both scenarios. Therefore, the approach focuses solely on the actual data visible to the user, disregarding other elements that do not convey information to estimate block content similarity.

Sometimes, small content blocks are added for visual effects only and do not impact the content. Symbols can be added to separate elements, like at the beginning or end of a menu. A similar issue is discussed by Fauzi et al. (2009), where only meaningful images are extracted, and non-relevant images are ignored. Hamming normalised distance was used, which was described by Bookstein et al.

(2002), to measure the similarity between the content of the blocks while accounting for noise in the content. This was done by removing HTML blocks and only comparing clean text. The threshold value was set at 0.1 after analysing common trends in web pages. A comparison value below 0.1 means that the minimum boundary has not yet been found. As soon as the value of 0.1 or above is achieved, the minimum boundary was reached in the previous step. Then, all the blocks between the maximum and minimum boundaries (including the boundary blocks) are stored as the length boundaries of the blocks.

However, this method has a limitation: it does not work for blocks without text content. This can happen when self-closing HTML tags are used, like `<img>` (Figure 4.4). The data for these tags are stored in attributes, so after removing the HTML tags from the text, the content of such a block is empty. Another situation is when content is added via CSS rules. For example, links to websites often use social media icons, which are usually applied to an HTML tag via CSS, not HTML. This would also result in no content in the HTML tag. Nevertheless, these are rare cases and mainly deal with visual information.

```

1  <section><!-- B1 maximum boundary start -->
2      <div>
3          <div><!-- B1 minimum boundary start -->
4              <div><!-- B1* minimum boundary start if image is ignored -->
5                  <h1>Heading</h1>
6                  <p>Paragraph</p>
7                  <a href="url"><i class="fa fa-play"></i> Link text</a>
8              </div><!-- B1* minimum boundary end if image is ignored -->
9              
10         </div><!-- B1 minimum boundary end -->
11     </div>
12 </section><!-- B1 maximum boundary end -->

```

Fig. 4.4. HTML code example illustrating the range between maximum and minimum boundaries when empty content elements are included

Modifications to the algorithm were made to ensure accurate capture of blocks. Initially, the algorithm checks if any content is present in the parent block. If it is empty and the block being examined is its only child, it is logical to deduce that this child block can be established as the block length boundary.

Analysing content-rich blocks necessitates an additional check for images to determine block boundaries accurately. This involves tallying the image tags within both parent and child blocks, but only when the textual contents of parent and child blocks are identical and more than one child block exists. Figure 4.4 illustrates that an incorrect block length variation may be identified if image-like blocks are not considered. Ignoring the image tag in this scenario results in

incorrectly determining the minimum content block boundary, elevating it by one level compared to the accurate detection.

Algorithm 2 describes the process for estimating HTML block boundary ranges, considering content similarity and the existence of tags. The method is recursively applied to each candidate for the minimal boundary and each child block, traversing all DOM trees from top to bottom. This refined approach enhances the precision of content extraction, a crucial advantage for applications such as web scraping and content aggregation that demand accurate content extraction from web pages.

Algorithm 2. boundaryEstimation pseudocode

```

input: block for analysis and its parent block
set minBoundary to block
set maxBoundary to block
set distance to 0
repeat
    call getChildren with block return children
    if parentBlock clean text is empty then
        if number of children <= 1 then
            set distance to 0
        else
            set distance to 1
        end
    else
        if parentBlock clean text = block clean text and
        number of children >1 then
            call getImageCount with parentBlock re-
            turn parentImages
            call getImageCount with block return blockImages
            if parentImages = blockImages then
                set distance to 0
            else
                set distance to 1
            end
        else
            call hammingDist with block text and parentBlock
            text return distance
        end
    end
    set block to first children
    set minBoundary to first children
until distance >= 0.1
store block data with minBoundaries and maxBoundaries
store block data with children relations
retrun minBoundary

```

4.1.4. Method for Block Variation Estimation

The term “block type variation” refers to identifying blocks that serve the same purpose but contain different content. This identification process allows for the categorisation of similar blocks into clusters. Here, clusters are groups of blocks that share a common parent block. The process involves navigating through the HTML tree to search for neighbouring HTML blocks that exhibit similar structures. The HTML path similarity estimation algorithm aids in evaluating the structure similarity. During the navigation of the HTML tree, the focus is on blocks that contain multiple child blocks, while those with only a single child block are ignored. Upon encountering blocks that possess multiple child blocks, their structures are compared to assess similarity. At this stage, the evaluation is focused solely on the structure without considering the content. The Sequence Matcher method Kiesel et al. (2020) present helps calculate the structural similarity. A slightly modified version of the algorithm was utilised, with the autojunk heuristic turned off, leveraging pre-processed HTML structures for the analysis.

Algorithm 3 outlines the basic approach, which involves comparing each block with its adjacent sibling to determine their similarity. Various experiments were conducted under different scenarios to establish a threshold value for similarity. The scenarios selected for the experiments were independent of the web pages used subsequently. The experiments revealed that a sequence matcher similarity exceeding 0.92 is sufficient to confirm whether two adjacent sibling blocks are variations. Further experiments with XML schema similarity performed by Algergawy et al. (2010) suggested that optimal results could be obtained with a weight between 0.8 and 0.9. Since HTML tags are more generic, the threshold value was increased to 0.92. This adjustment facilitates the interconnection and estimation of block clusters. Consequently, during transformations, applying a transformation to one block automatically creates links to other blocks. These links can then be leveraged to transform the block variations as well.

Algorithm 3. getVariations pseudocode

```

input: blocks for analysis
for each block in blocks
    call getSiblings with block return siblings
    for each sibling in siblings
        call structuralSimilarity with block and sibling re-
        turn similarity
        if similarity > 0.92
            store variation between block and sibling
        end
    end
end

```

4.1.5. Novelty of the Proposed Methods

The originality of these methods lies in several aspects:

1. **Enhanced Data Extraction from Content Blocks.** The proposed method emphasises a more nuanced extraction of data from content blocks. It not only pinpoints content blocks but also establishes the boundaries of variations. This data is valuable for precisely comparing different web page blocks.
2. **Simplification of Web Page Data Labelling.** Methods are proposed for partitioning a web page into distinct content blocks. This approach streamlines the manual labour involved in web page data labelling. The identified content blocks are then further grouped to minimise the total number of blocks needing labelling. Moreover, the proposed structure ensures the traceability of all blocks, allowing labels of one element in a group to be linked with the rest of the elements in that group.
3. **Integration of Text and Structure Similarity.** The proposed approach is distinctive as it merges web page block text and structure similarity. Like the Hamming distance used for text similarity estimation, parent and child relationships are considered when identifying group boundaries.
4. **Comprehensive and Efficient Content Block Identification.** Compared to the solution provided by Andrew et al. (2019), the proposed method identifies as many content blocks as are present on the page without restricting the number of blocks. In contrast to other solutions that do not fix the segment number, like the one by Jayashree et al. (2022), the proposed method is more efficient as it does not necessitate two separate stages (first identifying the number of clusters and then partitioning the web page into that number of blocks) and extracts all possible content blocks from the web page. Furthermore, the proposed method is more comprehensive than just extracting text-containing structured blocks like the one proposed by Manickam et al. (2022). However, it extracts all blocks, including those that are not structured, presenting a capability similar to the solution by Amarnadh et al. (2019).

Overall, the proposed methods contribute significantly to the current body of knowledge by offering a more detailed, efficient, and comprehensive approach to web page content block identification and labelling.

4.2. Validation of the Proposed Method for Web Page Division to Content Blocks

After performing initial limited testing during method development, a complete and thorough test of the proposed segmentation method was needed. Multiple issues were solved to perform validation, such as the lack of a suitable open-source dataset that would have data suitable for validating the results of the proposed method. This evaluation aims to demonstrate the practical applicability and robustness of the proposed method in performing web page content segmentation.

4.2.1. Obtaining Data Suitable for Proposed Method Validation

Confirming the suggested techniques is intricate since all currently available datasets are designed for segmenting web pages and do not contain comprehensive information regarding the boundaries and variations of blocks, as noted by Wang et al. (2022). Additionally, most of the data sources referenced in past studies are not accessible for replication of experiments. Furthermore, the aims of the methods are diverse, making a precise comparison nearly impossible to achieve. Therefore, to confirm the effectiveness of the suggested solutions, a series of tests was conducted to compile the necessary dataset.

For the tests, ten pre-existing web pages were employed, and three additional web pages were created (<https://1.kiril.dev/>, <https://5.kiril.dev/>, <https://6.kiril.dev/>, accessed on 27 April 2023) to represent typical one-page websites with various content sections. These pages were randomly selected from a collection of one-page website designs and saved in a designated repository to prevent any alterations in the future. Designed by web professionals using the Bootstrap framework, each page includes the website's fundamental structure and creative touch. A thorough manual review was conducted to label as many distinct content sections as possible.

For these tests, ten additional pages were selected to reflect not only the Bootstrap layouts but also the web design used by the most popular websites at that moment. In total, 13 web pages were scraped, labelled and analysed. The websites for the developed dataset were selected considering the appropriate criteria:

1. <https://1.kiril.dev/> (accessed on 27 April 2023) was selected as a case of the most popular one-page design templates from Bootstrap to illustrate what structure is presented in up-to-date web designs.
2. <https://5.kiril.dev/> (accessed on 27 April 2023) was selected as another relevant example of Bootstrap templates, to use as an example of pre-loader usage, which potentially could limit the web data gathering.

3. <https://6.kiril.dev/> (accessed on 27 April 2023) is the last Bootstrap template, selected as an example of animation integration into the one-page web page structure to estimate its effect.
4. <https://www.youtube.com/> (accessed on 27 April 2023) was selected as one of the most visited open websites in the world.
5. <https://addons.mozilla.org/en-US/firefox/> (accessed on 27 April 2023) was selected as a web user who mostly used the website to adjust Firefox web browser functionality. Firefox was selected as one of the alternatives to the main web-related organisations, considering Google is already included with YouTube.
6. <https://www.apple.com/> (accessed on 27 April 2023) was selected to reflect one of the manufacturers actively working in web browser-related areas.
7. <https://www.apple.com/retail/business/> (accessed on 27 April 2023) was selected to investigate how different parts of the same website vary in design and its elements usage.
8. <https://www.buzzfeednews.com/> (accessed on 27 April 2023) was selected as a social-first news site, which was surrounded with big attention regarding its further existence.
9. <https://gridbyexample.com/> (accessed on 27 April 2023) was selected to illustrate the up-to-date solution for grid layout implementation, as this web constantly updates, depending on the leading technologies for grid layout implementation.
10. <https://www.nytimes.com/> (accessed on 27 April 2023) was selected as one of the main news portals in the USA.
11. <https://slack.com/> (accessed on 27 April 2023) was selected as one of the productivity platforms.
12. <https://stripe.com/en-gb-lt/connect> (accessed on 27 April 2023) was selected as one of the most popular payment providers for platforms and marketplaces. Also, it is often mentioned for its unique and modern UX/UI solutions.
13. <https://www.telegraph.co.uk/news/> (accessed on 27 April 2023) was selected as another news portal, leading in the United Kingdom.

The selected set of websites illustrates different kinds of websites popular with adult users that do not require login. The systems, which require logins were not included to make sure the content is not affected by individual profiles and are harder to access for later reuse.

One-page websites or a single web page from a larger site were chosen to showcase a wide range of sections within a single page, highlighting the diversity

of content that can be presented in this format. A portion of one of the web pages and a sample of manual block identification are illustrated in Figure 4.5. The first-level blocks are marked with a red border, the internal blocks are highlighted in purple, and the innermost blocks are outlined in green. This example encompasses all content sections, and their hierarchy is discernible, which might minimise the need for manual segmentation efforts in instances such as menu items, contact elements, or fields of contact forms.

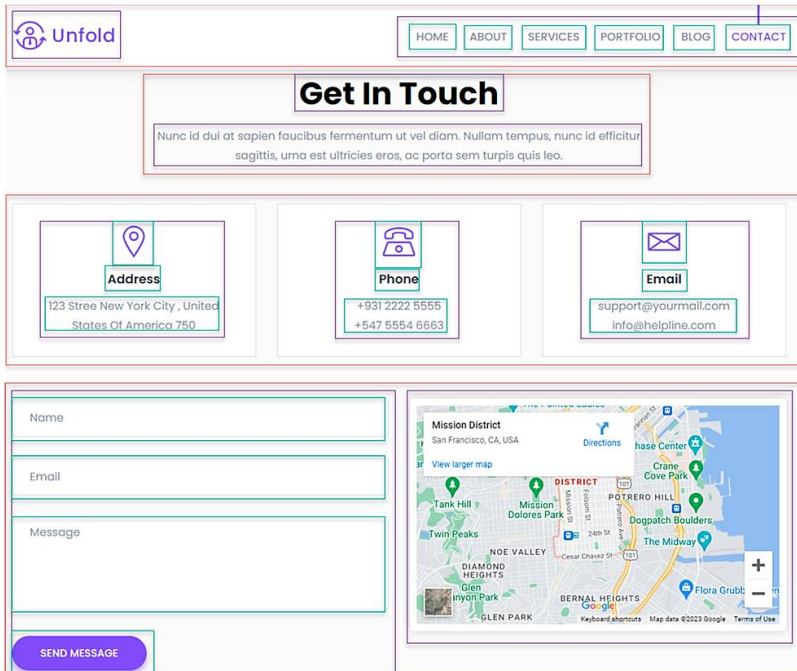


Fig. 4.5. Visual view of a web page fragment with identified blocks and their hierarchy

A web-based system was developed to enhance the accuracy of data labelling, emphasising that block coordinates and selectors are crucial in addition to just the labels. Participants labelling the web pages were instructed to identify all visible content sections, including those of different sizes. However, identification was limited to only one section among equivalent sections with different content, such as a single menu item instead of each item individually within the same menu. All the labelled data were stored in a database for later comparison with the content sections of web pages identified automatically. Although the section's label was

not mandatory for this research, this information is retained in the same database for potential utilisation in future research.

During the web pages’ manual labelling process, 40492 tags were identified, and 16453 excluding YouTube (WYT) (Table 4.1). Later, two sets of values for most metrics will be provided, acknowledging that YouTube employs many proprietary tags, which can significantly influence certain statistics. In any case, this quantity of labelling data is overwhelming for average users, and procuring expert labelling for a large number of websites could be exorbitantly expensive.

Table 4.1. Summary of manually labelled data

No.	Web Page	No. of Tags	Labelled blocks	Percentage of Labelled Tags
1	https://1.kiril.dev/ , accessed on 27 April 2023	521	89	17%
2	https://5.kiril.dev/ , accessed on 27 April 2023	531	84	16%
3	https://6.kiril.dev/ , accessed on 27 April 2023	393	64	16%
4	https://www.youtube.com/ , accessed on 27 April 2023	24039	75	0%
5	https://addons.mozilla.org/en-US/firefox/ , accessed on 27 April 2023	981	61	6%
6	https://www.apple.com/ , accessed on 27 April 2023	1087	69	6%
7	https://www.apple.com/retail/business/ , accessed on 27 April 2023	826	74	9%
8	https://www.buzzfeednews.com/ , accessed on 27 April 2023	1523	98	6%
9	https://gridbyexample.com/ , accessed on 27 April 2023	111	17	15%
10	https://www.nytimes.com/ , accessed on 27 April 2023	2474	203	8%
11	https://slack.com/ , accessed on 27 April 2023	768	87	11%
12	https://stripe.com/en-gb-lt/connect , accessed on 27 April 2023	4990	144	3%
13	https://www.telegraph.co.uk/news/ , accessed on 27 April 2023	2248	114	5%
	In total	40492	1179	3%
	In total, without YouTube (WYT)	16453	1104	7%

Individuals with basic HTML knowledge and no prior data labelling experience performed the labelling. Respondents labelled 1179 (1104 WYT) blocks across all websites, representing merely 2.9% (6.7% WYT) of the total HTML blocks on the inspected web pages. The percentage of labelled data across all web page tags indicates the ratio between unique labels and tags necessary to construct a one-page website. On the other hand, a detailed dataset encompassing all tag paths is essential to developing precise machine-learning solutions for labelling web pages. This necessitates an almost 35-fold or nearly 15-fold increase in labelling effort if YouTube data is excluded. Consequently, labels should be duplicated or linked to different variations of tag paths to compile a more accurate dataset.

4.2.2. Results of Web Page Division to Content Block Test Cases

The same set of web pages was segmented into content sections using the proposed methods. Table 4.2 summarises the content sections identified on the web pages, highlighting an 80% reduction in content tags (71% WYT).

This significant reduction was realised through a series of steps. First, it is crucial to recognise that certain tags are irrelevant to the content (e.g., head tag and its contents, scripts, styles, etc.). Some tags, such as `<base/>`, `<link/>`, `<meta/>`, and `<style>`, can be easily omitted by only selecting the content within the body tag. Consequently, the total count of tags decreases from 40492 to 39095 (16453 to 15112 WYT), resulting in a 3.5% (8% WYT) reduction. Second, the content within the body tag also needs to be filtered, as it often contains tags that enhance the content but do not contain the content itself. For instance, script tags are commonly found within the body tag. By filtering the body content to exclude tags that do not display content, the content tag count reduces further from 39,095 to 16657 (15112 to 10914 WYT), accounting for a 57% (28% WYT) reduction. Cumulatively, this results in a 59% (27% WYT) reduction from the initial 40492 (16453 WYT) tags.

Any further advanced reduction would be impractical without first conducting these basic tag eliminations. Identifying the boundaries of content blocks, rather than all potential content tags, enables the achievement of the aforementioned 80% reduction (71% WYT). Of the filtered 16657 (10914 WYT) blocks, only 8091 (4633 WYT) remained after implementing the suggested method of web page division into content sections. This was accomplished by detecting variations of different block boundaries and retaining only one of the multiple identified sibling segments.

Table 4.2. Summary of data for automated web page division to content blocks

Web Page No.	No. of Web Page Tags	No. of Potential Content Blocks	Filtered Content Blocks	Reduced Content Blocks	Boundary range		No. of Blocks with Siblings	Siblings	
					Single Length	Max-Min Range		Main Sibling	Related Sibling
1	521	502	469	238	166	72	114	42	72
2	531	518	494	192	88	104	91	24	67
3	393	379	305	136	87	49	72	23	49
4	24039	23983	5743	3458	1772	2816	621	226	395
5	981	899	511	256	136	242	128	48	80
6	1087	919	718	281	216	239	218	59	159
7	826	763	612	285	185	262	148	42	106
8	1523	1433	975	472	264	444	144	32	112
9	111	87	82	45	38	27	21	4	17
10	2474	2410	2174	778	425	797	661	188	473
11	768	702	364	168	116	137	111	29	82
12	4990	4375	2866	1346	888	1091	909	251	658
13	2248	2125	1344	436	184	557	223	55	168
Total	40492	39095	16657	8091	4565	6837	3461	1023	2438
WYT	16453	15112	10914	4633	2793	4021	2840	797	2043

Consolidating content block boundaries and identifying sibling block variations led to a reduction in the filtered content blocks. Analysis of these methods reveals that approximately 51% (58% WYT) of the tags can be grouped into boundary ranges. A typical boundary block amalgamates around 5 (3 WYT) tags into a single block, specifying the minimum and maximum boundaries for that block.

Another strategy for reducing content blocks involves identifying pertinent blocks and tallying the path of a single content block instead of all. In the examined web pages, 3461 (2840 WYT) blocks had a sibling block. Of the condensed set of content blocks, 1023 (797 WYT) were chosen to represent sibling blocks, whereas 2438 (2043 WYT) were associated with them but discarded. This implies

that, on average, sibling blocks have three instances, but only a third of them need to be retained to represent the block pattern.

4.2.3. Results of Comparison between Web Page Division to Content Blocks and Manual Labelling

Storing the division of web pages into content blocks for both manual and automated labelling was executed similarly, except in the automated segmentation, where labels were not assigned, and these were stored in identical database structures but in distinct instances. This similarity in the database structure makes it easier to compare manual and automated divisions. However, it is a complex task, as manual labelling allows users to recognise specific tags and content blocks but possibly overlook others. Also, the users were not instructed to categorise tags into maximum and minimum boundary blocks. Therefore, additional strategies were devised to align a manually labelled tag with an automated division to a content block and then estimate whether the manually labelled tag is within the boundaries of the content block (minimum bounds $\leq$ labelled tag $\leq$ maximum bounds). This technique enables the estimation of label matches even when the boundaries are labelled in a similar but not identical manner. Another strategy assesses if a labelled tag aligns with a different content block based on its structure. This feature compiles information about sibling blocks, which were initially ignored for simplicity but corresponded to some previously identified blocks with different content.

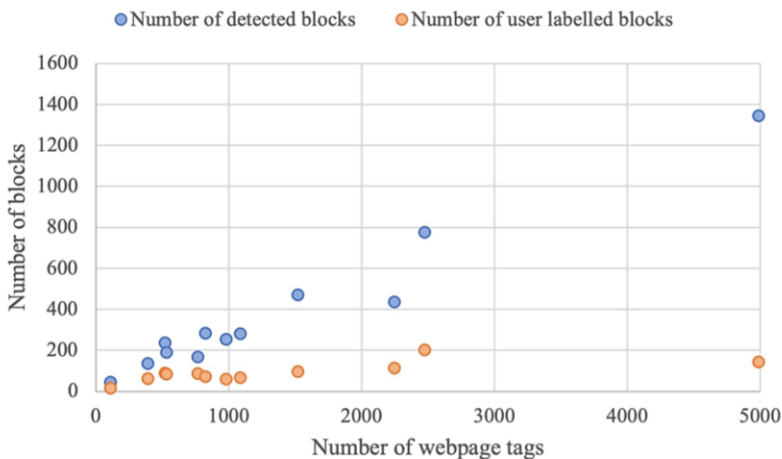


Fig. 4.6. Dependency between the human-labelled number of blocks and tags in the appropriate web page (Griazev & Ramanauskaitė, 2023)

This alignment between the automated division of web pages into content blocks and manual labelling made it possible to evaluate whether the solution could accurately identify all content blocks labelled by humans. In total, 1179 content blocks were labelled by human experts (1104 WYT). The relationship between the number of tags on the web page and the blocks identified by the proposed method is depicted in Figure 4.6, demonstrating a linear dependency.

The labelled blocks were correlated with those identified using the proposed method of content block identification. Standard classification metrics were employed to analyse the performance comprehensively. True positives (TP) were considered as the number of blocks proposed method indicated and matched with segments defined in the dataset or user-labelled blocks. False positives (FP) were the blocks detected by the proposed method but were not labelled in the dataset. False negatives (FN) were the blocks labelled in the dataset but not detected by the proposed method, and true negatives (TN) were calculated by subtracting TP, FP, and FN from the total number of tags on the web page. A summary of these metrics is provided in Table 4.3.

Table 4.3. Summary of data for automated web page division to content blocks

Web Page No.	Number of Blocks	Number of				Precision	Recall	Accuracy	F-Score
		True Positive	True Negative	False Positive	False Neg-				
1	238	89	0	149	0	37%	100%	71%	54%
2	192	84	0	108	0	44%	100%	80%	61%
3	136	64	0	72	0	47%	100%	82%	64%
4	3458	75	0	3383	0	2%	100%	86%	4%
5	256	61	0	195	0	24%	100%	80%	38%
6	281	69	0	212	0	25%	100%	80%	39%
7	285	74	0	211	0	26%	100%	74%	41%
8	472	98	0	374	0	21%	100%	75%	34%
9	45	17	0	28	0	38%	100%	75%	55%
10	778	203	0	575	0	26%	100%	77%	41%
11	168	87	0	81	0	52%	100%	89%	68%
12	1346	144	0	1202	0	11%	100%	76%	19%
13	436	114	0	322	0	26%	100%	86%	41%
Overall	8091	1179	0	6912	0	15%	100%	83%	25%
WYT	4633	1104	0	3529	0	24%	100%	79%	38%

The findings indicate that the proposed solution for dividing web pages into content blocks can accurately identify all content blocks that would be manually labelled. Additionally, it identifies extra content blocks that were overlooked during manual labelling, primarily due to their repetitive nature. This repetitiveness appears in two ways. First, in cases where a block can have multiple boundaries while containing identical content, the block's max-min boundary detection method is employed to eliminate the need for labelling all possible combinations of the same block. Second, repetitiveness occurs when multiple content blocks serve identical purposes but contain different content, such as a navigation menu. Each menu element has structures identical to other elements of the same menu, so users typically mark only one menu element. The structural sibling relationships between blocks are utilised to detect other menu elements, enabling the identification of all menu items regardless of which one the user labelled. Occasionally, both forms of repetitiveness co-occur; e.g., menu items may have multiple boundaries within the max-min range, necessitating both techniques to determine all potential labelled block variations.

The results from comparing automated web page division into content blocks with manual labelling indicate that the solution can identify all manually labelled blocks, directly or indirectly, by utilising related sibling records, resulting in 100% precision. Expanding the testing data is challenging because existing web page segmentation or labelling datasets are only partially compatible with the extended model of web page labelling.

Regarding the accuracy of the proposed method, it was approximately 83% (79% WYT), considering the number of blocks labelled by a person and present in the dataset of automatically detected blocks after excluding relevant siblings. These conditions align with those established for manual labelling, e.g., labelling all components serves the same purpose. Under these conditions, the F-score was approximately 25% (38% WYT).

4.2.4. Comparison of the Proposed Method with Existing Segmentation Methods

Comparing the suggested approach with other available methods for dividing web pages into content blocks is challenging since there are no precise counterparts. Nonetheless, solutions for web page segmentation are inherently quite alike. Most of these methods are usually validated using a widely recognised dataset by Kreuzer et al. (2015). However, the labelling of this dataset is less extensive than the proposed method intends to achieve. The difference is noticeable in both the quantity of blocks (with the dataset containing an average of 13 blocks compared to the previously evaluated web pages, which averaged 79 blocks) and the specifics provided for each block (the dataset outlines exact block boundaries in contrast to

the approach and previously used web page analysis data which offer minimum and maximum ranges for each block, along with information on the relationships between similar blocks, or siblings). Although this dataset comprises a significantly larger number of records, the performance of web pages is typically assessed using this dataset.

This dataset was chosen because several existing web page segmentation methods have already utilised it. Hence, accuracy metrics for those methods are available. The precision, recall, accuracy, and F-score were computed for each record in the dataset, and their average values were then calculated to summarise the findings. Using this dataset, the proposed method yielded a precision of 11%, a recall of 100%, an accuracy of 77%, and an F-score of 19%. These results suggest that the proposed method needs more precision (11%), and the F-score (19%) is the lowest among all other methods. This, however, is attributed to the fact that the dataset only includes a limited subset of labelled blocks and segments.

In contrast, the proposed method is designed to identify all potential content blocks. Furthermore, these figures are not directly comparable because other studies estimated the accuracy of utilised segments, not content blocks. Nevertheless, even with the proposed method of grouping similar elements into categories, it exhibits high accuracy (77%) comparable to other web page segmentation methods while maintaining a constant recall rate (100%). This is due to the fact that no blocks are omitted from the web page; instead, blocks are assigned to different variations or groups.

4.2.5. Evaluation of Validation Results

In the suggested approach, the DOM tree and web page similarity estimation are used instead of comparing web pages visually. This makes the application process more straightforward as it does not require the use of complex data clustering models. Although it is not possible to directly compare the outcomes of the proposed method for dividing web pages into content blocks with other research outcomes due to differing objectives, data, and dataset labelling specifics, the outcomes of the proposed method resemble the outcomes obtained by other methods of web page labelling or segmentation, e.g., the ones presented by Kiesel et al. (2021). Since outcomes may vary based on the dataset and labelling, the proposed approach was juxtaposed with a dataset by Kreuzer et al. (2015) and methods used to segment that same dataset. The outcomes for the proposed method are comparatively lower, which can be partially attributed to the dataset having an extremely limited number of labels and being designed to present segments, not all content blocks.

The precision value is considerably low compared to existing research and experimental outcomes, which is influenced by the dataset's limited number of

labelled blocks. The dataset only includes the main blocks; not all minor elements were labelled internally. On the other hand, recall is at the opposite end of the spectrum. Proposed methods achieve 100% recall, surpassing all other methods. This is because no blocks are discarded; instead, blocks are categorised into variations or sibling groups.

To sum up the outcomes and limitations of the method, it has excellent potential and could be utilised to assist in manual web page labelling. The method could extract all potential content blocks from the web page and present them to the individual performing the labelling, reducing the need for tag/block revision by 90% (70% excluding the YouTube case). Simultaneously, the labelling data will be more comprehensive in terms of relations between blocks. This could be utilised for more interactive labelling, where assigning a label to one block automatically generates suggestions for the labels of other related blocks.

As more detailed datasets with extended labelling data become accessible, the method could be refined to identify or suggest block labels. This would lead to a complete comprehension of the web page structure, enabling automated integration and transformation of web page content.

4.3. Establishing Connections Between Content Blocks from Different Web Page Regions

Segmentation of web pages and content extraction have been pivotal research areas for several years, given the exponential increase in information that necessitates processing. Numerous tactics have been devised to augment the efficiency and efficacy of data extraction and web segmentation tasks (Eldirdiery et al., 2015). This section elaborates on some of the most pertinent studies related to the proposed technique, emphasising block detection, content extraction, and the implementation of machine learning methodologies.

Groundbreaking works in this field include the Vision-Based Page Segmentation (VIPS) algorithm (Cai et al., 2003), which was developed to extract the semantic structure of a web page based on its visual presentation. Proponents of another methodology leveraging computer vision implemented a framework based on computer vision to discern semantic data showcased on websites, regardless of the web framework in use (Cormier et al., 2015). Another significant contribution is the RoadRunner methodology, which concentrates on extracting the content structure and data from web pages utilising wrappers (Crescenzi et al., 2001). This system adopts a semi-automated approach, merging human oversight with automated algorithms. This underscores the cruciality of amalgamating manual labelling and automated methods in the processes of web page segmentation and content extraction. The proposed method was contingent on automatically

generating wrappers for the data extraction process. To fabricate these wrappers, HTML pages were juxtaposed to discern their commonalities and discrepancies, which were subsequently utilised to craft a wrapper.

A fundamentally divergent approach was proposed, employing text database queries for proficient information extraction (Agichtein et al., 2003). This tactic harnesses a blend of text mining methods and web page analysis to bolster the quality of content extraction. This research accentuates the advantages of integrating natural language processing methodologies into the proposed technique to enhance the identification of related blocks and their associated functions.

Machine learning methodologies are exceedingly popular owing to the perpetually expanding volume of data necessitating processing. To diminish the effort requisite for processing this perpetually growing data volume, an MLOSS framework was suggested (Silpa et al., 2022). It amalgamates genetic algorithms with supervised and unsupervised methods to pinpoint optimal data segments within web documents.

Multiple clustering approaches are also under investigation. In clustering methodologies, diverse data types can be harnessed to gauge similarity; some techniques leverage structural information, and others process textual data and visual or stylistic elements. Researchers frequently employ a fusion of multiple data sources, for instance, to cluster web pages grounded in their structural and stylistic similarity scores (Gowda et al., 2016). In this instance, structural similarity was ascertained using tree edit distance, whereas stylistic similarity was gauged using the Jaccard similarity measure.

A paramount obstacle during web segmentation is the imperative to identify and correlate similar content that may be dispersed across disparate segments of the same page (Liu et al., 2004). This is vital to lessen the workload necessary for content labelling and diminish computational requirements during content filtering and analysis, as some segments of similar content can be omitted from the analysis.

4.3.1. Recognising Sibling Blocks on a Website

Websites typically conform to a specified design language, replicating various design elements and content layouts across the website. Consequently, the same component may be replicated to construct diverse composite content blocks. Undertaking content analysis of such a page would necessitate processing identical types of components multiple times per page, which is suboptimal in terms of processing expense and speed. The proposition introduces a technique that identifies these recurrent blocks across the page, excluding them from subsequent processing while preserving all information about them, their content, and their associations with other content blocks.

In the process of pinpointing repeated content blocks, two unique situations can occur. The first situation is observed when all duplicated blocks originate from the same immediate parent block or container. In such instances, these blocks are referred to as direct siblings. As depicted in Figure 4.7, there are two separate sections on a website, one with a dark background and another with a light one. Both sections incorporate the same kind of content block, indicated by a red outline. In Figure 4.7, the direct sibling blocks are linked by a yellow arrow. These linkages are presented in a simplified manner to ensure ease of understanding. Even though it might seem that a direct sibling block only contains information about the adjacent direct sibling, it keeps track of relational information about all its sibling blocks. A remote sibling connection between blocks of the same kind but originating from different parent blocks is marked by a green arrow. Like direct siblings, remote siblings also hold information about every other remote sibling within the same connection. Figure 4.7 is a streamlined representation, displaying only a single remote sibling connection to aid comprehension.

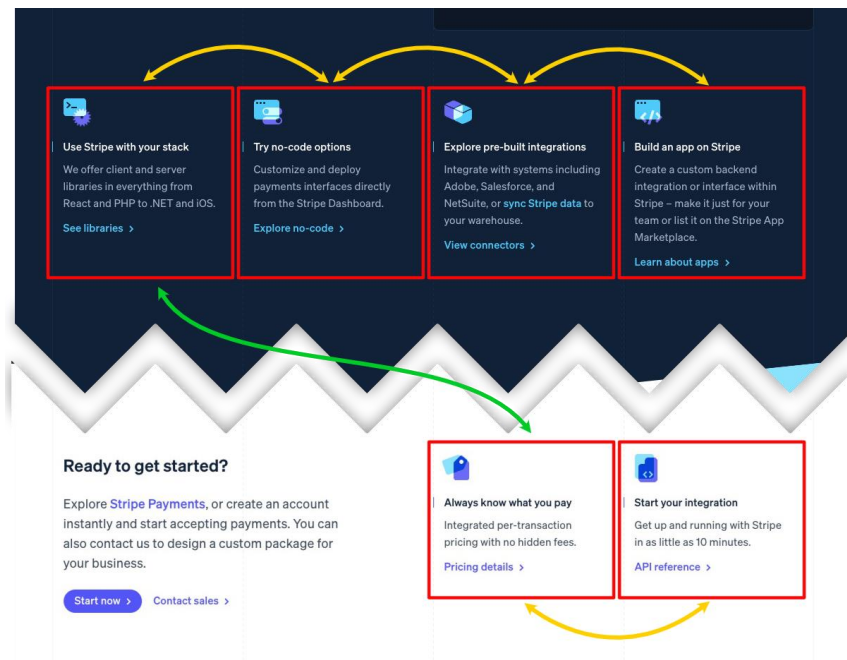


Fig. 4.7. Depiction of direct (indicated by a yellow arrow) and remote (indicated by a green arrow) sibling blocks in different sections of the same website

The practical application of remote sibling detection is more clearly grasped with a demonstrative example. Figure 4.8 illustrates an instance of content block minimisation utilising direct sibling identification. Each depicted section includes

a header, subtitle, a container holding two varieties of card blocks, and a footer. Despite the considerable similarity between these sections, the sections would not be categorised as analogous blocks due to the varying number of card blocks in each section. Moreover, even if the card block count were identical, the sections would not register a high similarity score because of the distinct combinations of card layouts. In this example, focus is placed on the card blocks because those blocks encompass both direct and remote siblings. Additionally, this highlights the intricate nature of web content analysis, as seemingly similar blocks may have underlying differences that affect their categorisation. This underscores the need for advanced techniques, such as remote sibling detection, to accurately identify and process repeatable content blocks.

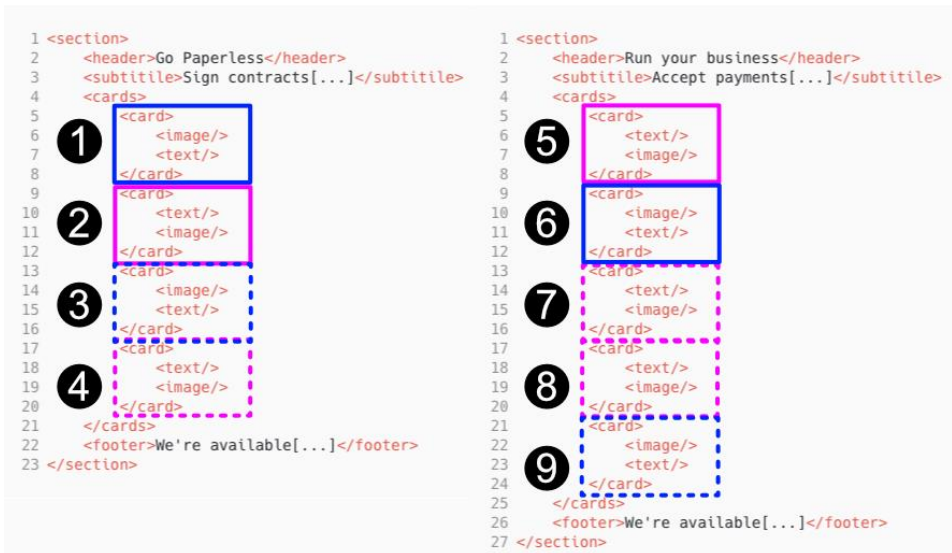


Fig. 4.8. Reduction of content blocks using direct sibling detection

In the early phases of web page analysis, the identification of direct siblings, that is, similar blocks that share the same parent, was already conducted. In Figure 4.8, the blocks recognised as direct siblings would be organised into four groups, two for each section. For the left section, the first group of direct siblings includes blocks 1 and 3, while the other group consists of blocks 2 and 4. On the right side, the first group contains blocks 5, 7, and 8, and the second group comprises blocks 6 and 9. The detection of direct siblings allows for the reduction of blocks requiring manual labelling from nine to four, specifically, those outlined with solid borders. This detection process is relatively uncomplicated since all the blocks in this category share the same parent, the `<cards>` block, in both sections

illustrated in Figure 4.8. Nevertheless, the advantages of this detection are restricted because only one group of blocks can be removed using this method. In the section on the left, blocks 3 and 4 can be excluded; on the right, blocks 7, 8, and 9 can be eliminated. This permits the omission of five blocks from further analysis or labelling, allowing the process to continue with four blocks (1, 2, 5, and 6).

In scenarios where the same kind of blocks appears in various locations on a single website, this detection method can be amplified to identify not only blocks with a common parent but also those with a distinct parent but identical structure. As previously mentioned, similar blocks with a common parent block are identified as direct siblings, while similar blocks with different ancestral blocks are termed remote siblings. This distinction is crucial for a more comprehensive analysis of web pages as it allows for identifying and processing repeatable content blocks that may not be immediately apparent due to their location or parentage. This, in turn, facilitates a more efficient and thorough labelling and analysis process, contributing to the overall effectiveness of web content management and transformation.

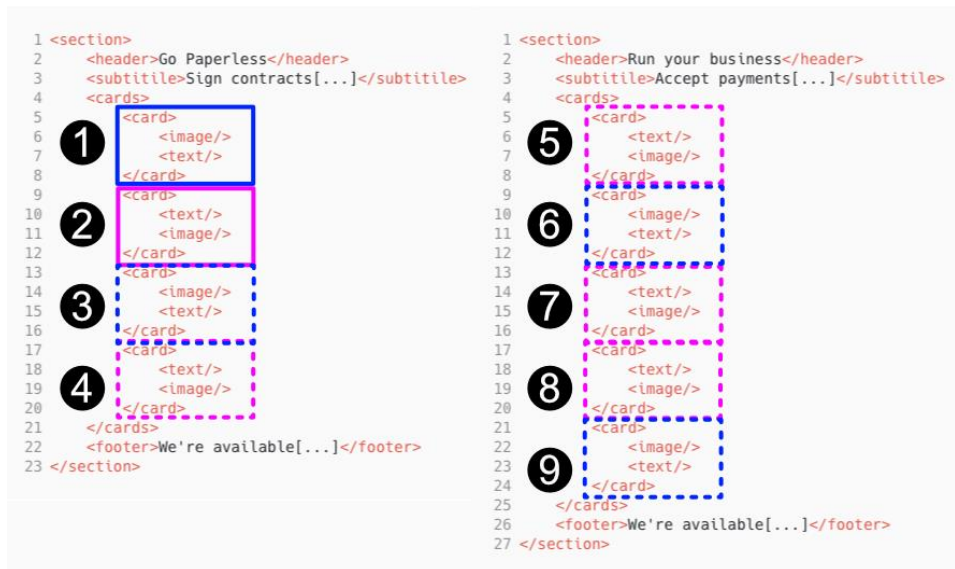


Fig. 4.9. Reduction of content blocks using remote sibling detection.

The advantages of remote sibling detection become more evident when it is applied to the marked blocks in Figure 4.8. It further reduces the number of blocks that need to be analysed. Figure 4.9 illustrates a scenario where blocks from both sections are evaluated simultaneously, in contrast to the earlier approach of separate analysis for each section. By implementing remote sibling detection, blocks

5 and 6 can be excluded from further examination, leaving just two blocks (1 and 2) for cards. Meanwhile, the relationships established using direct and remote sibling detection facilitate the extraction of the remaining seven blocks (numbered 3, 4, 5, 6, 7, 8, 9). The relationships between the blocks are illustrated in Figure 4.10, where the blue and purple colours represent different structures of child blocks (Figures 4.8 and 4.9). Nodes with solid borders are the primary parent blocks within a container that hold multiple blocks, whereas blocks with dotted borders signify direct children within the same container.

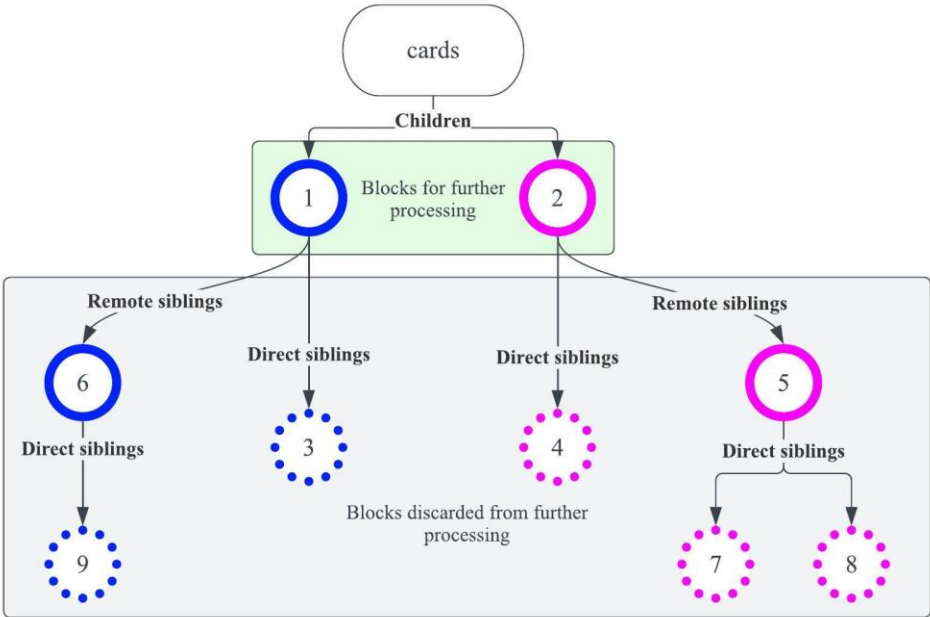


Fig. 4.10. Relationship connections for card blocks and the elimination of blocks from further processing

This illustration underscores the powerful efficiency of remote sibling detection in minimising the number of blocks that need to be manually analysed or labelled. It demonstrates a significant reduction in manual effort and computational resources, thereby enhancing the overall efficiency of the web page analysis process. This method is particularly useful for large websites with numerous pages that reuse similar content blocks, as it facilitates the identification and extraction of repeatable blocks across the entire website.

4.3.2. Implementation of Remote Sibling Block Detection

To identify remote siblings, it is imperative to examine the structure of a block and juxtapose it with other blocks that have already been processed. To streamline the analysis of block structure, the HTML tree is transformed into a bespoke tree format, similar to the Newick tree format. This format was adjusted to align with the specific requirements of the use case. In this adaptation, it is mandated that all nodes must have a designated name for which tag names are used, and the need for distance values is eliminated so blocks are neither incorporated nor retained. Adjacent to each node, supplementary data related to tags, such as class names, IDs, or any other attributes and their associated data, can be stored. In the current development phase, class names are actively employed while constructing content block trees.

Figure 4.11 showcases an example block for which a tree will be constructed, providing a visual depiction and the block's code. The process of generating a tree unfolds in two stages. Initially, a preliminary tree is constructed. Following the creation of this initial tree, it undergoes optimisation to simplify subsequent processing for detecting remote siblings. During this optimisation phase, all superfluous data placeholders are eradicated. In the context of this example, such placeholders are denoted as empty parentheses – ().

In essence, the process involves the transformation of HTML trees into a more specialised tree format, followed by optimisation to remove unnecessary elements. This streamlined structure facilitates the identification of remote siblings and enhances the overall efficiency of the web page analysis process. This process is especially crucial for managing the complexity of modern web pages that often contain intricate structures and repeated content blocks.

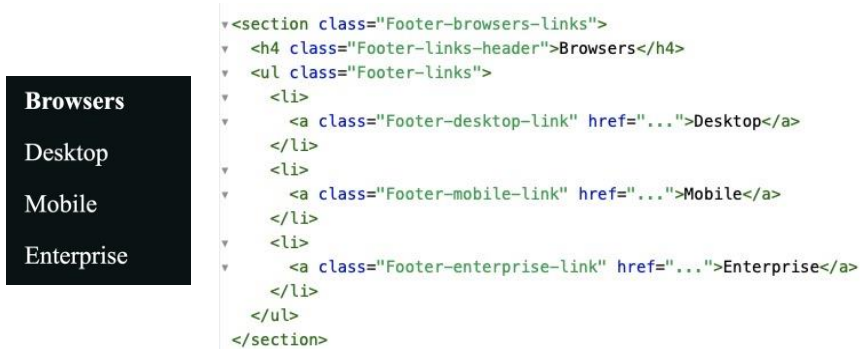


Fig. 4.11. Example block that will be used for tree generation

During the tree construction phase, two distinct tree variations are formulated. These variations are instrumental in identifying two different categories of remote sibling blocks. One variation exclusively retains the HTML structure and tag name information (Table 4.4). Siblings identified using this tree are designated as “loose remote siblings”, given that the comparison is relatively lenient, relying solely on the structure and tag names. Conversely, a second tree variation also incorporates class names (Table 4.5). This particular tree is leveraged to pinpoint “strict remote siblings” as the detection process considers the HTML class names data, making the detection more stringent.

This dual approach ensures a comprehensive analysis, catering to different levels of stringency in the detection process. This is especially crucial in handling the diverse nature of web page content, where some scenarios may require a broader detection approach while others necessitate a more meticulous analysis.

Table 4.4. Examples of loose trees generated from example HTML block

Initial tree	(section(h4(),ul(li(a()),li(a()),li(a()))))
Optimised tree	(section(h4,ul(li(a),li(a),li(a))))

Table 4.5. Examples of strict trees generated from the example HTML block

Initial tree	(section.Footer-browsers-links(h4.Footer-links-header(),ul.Footer-links(li(a.Footer-desktop-link()),li(a.Footer-mobile-link()),li(a.Footer-enterprise-link()))))
Optimised tree	(section.Footer-browsers-links(h4.Footer-links-header,ul.Footer-links(li(a.Footer-desktop-link),li(a.Footer-mobile-link),li(a.Footer-enterprise-link))))

Identifying remote siblings involves navigating through a collection of generated trees and comparing them with others in the set. This process is conducted individually for each tree type, resulting in two separate matching operations: one for the loose trees and another for the strict trees. The degree of similarity between the trees is evaluated using Gestalt pattern matching. At this point, only blocks that match completely are selected and marked as siblings, the similarity threshold can be reduced with further testing. The entire procedure for detecting and extracting remote siblings is visually represented in Figure 4.12.

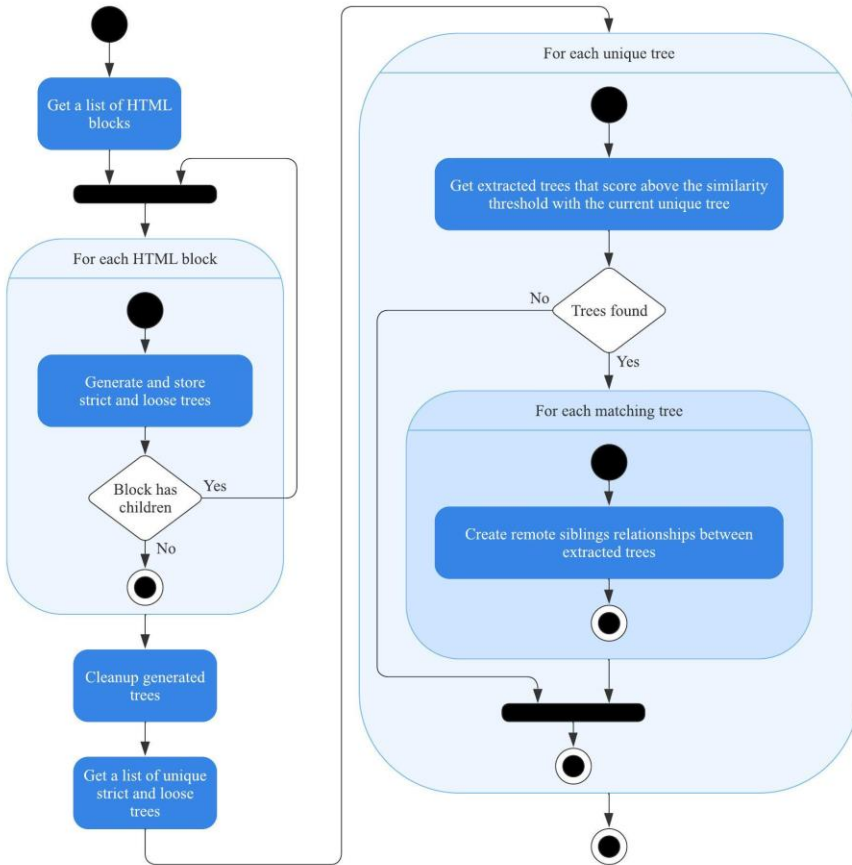


Fig. 4.12. Process of remote sibling block detection and extraction

Detailed descriptions of algorithms used in Figure 4.12 are presented below.

Algorithm 4 is responsible for fetching data from a specified URL (depending on the sourceId), executing a Python script to process this data, and then handling the output. It cleans up the output, removes duplicates, and processes the blocks for loose and classed siblings by calling another function with specific relationship types.

Algorithm 4. remoteSiblings

```

input: url to fetch website data from dataset
set json = call process_web_page and decode output as JSON
set siblingsLoose = call cleanupRemoteSiblings with
    json['loose']
  
```

```

set siblingsClassed = call cleanupRemoteSiblings with
json['classed']
set uniqueLoose = Remove duplicates from siblingsLoose
set uniqueClassed = Remove duplicates from siblingsClassed
call extractBlocksForSiblings for loose siblings using unique-
Loose, siblingsLoose, and relationship type 'remoteLooseStruc-
turalSiblings'
call extractBlocksForSiblings for classed siblings using
uniqueClassed, siblingsClassed, and relationship type 'remot-
eStructuralSiblings'

```

Algorithm 5 serves as the entry point of the Python script. It takes a URL as input, fetches the content, and processes it to extract information about remote siblings in the HTML structure. After invoking the `detect_remote_siblings` function on the fetched data, it outputs the structured information (in this case, the `block_trees` data structure) in JSON format. This function ties together the data fetching, processing, and output generation process, making it the core driver of the script's functionality.

Algorithm 5. `process_web_page`

```

input: url to fetch website data from dataset
try:
    open a connection to the URL and read the response
    parse the response as JSON into json_object
    set block_trees with 'loose' and 'classed' as empty diction-
aries
    for each block in json_object:
        call detect_remote_siblings with block and block_trees
    end
    return block_trees as a JSON string
catch URL, HTTP, timeout, and other errors:
    log the appropriate error message
end

```

Algorithm 6 is designed to process a block of HTML content to detect “remote siblings”, which are HTML elements that are not directly related but share specific structural characteristics. It parses the HTML content, utilises the `tree_gen` function to generate the structural trees, and then stores these trees in a provided data structure (`block_trees`). This function is vital in identifying potential relationships between different parts of a webpage based on their structural layout.

Algorithm 6. `detect_remote_siblings`

```

input: HTML page block and previously processed block_trees

```

```

set child_html = parse block['code'] as HTML
call tree_gen with child_html to generate its tree structure
update block_trees['loose'] and block_trees['classed'] with the
new tree structure for the current block
return block_trees

```

Algorithm 7 generates a tree structure from a given HTML content. It recursively traverses through the HTML elements, creating two types of tree representations: “classed” and “loose”. The “classed” tree includes the tag names and their class attributes, while the “loose” tree only contains the tag names. This function is instrumental in capturing the hierarchical structure of HTML content, which can be used for further analysis, such as comparing similarities between different HTML blocks.

Algorithm 7. tree_gen

```

input: parsed block
if html is a NavigableString or has no children:
    return an empty tree structure
end
set tree_classed and tree_loose as empty lists
for each child in html's children:
    if child has a name and is not an 'svg' element:
        call recursively tree_gen on the child to get its tree
structure
        concatenate class attributes of the child, if any
        append tree_classed with the child's tag (and class at-
tributes)
        append tree_loose with the child's tag
    end
return a dictionary with 'classed' and 'loose' keys mapping to
their respective tree strings

```

Algorithm 8 cleans up an array of sibling data. It removes empty parentheses from each item in the array and then filters the array to keep only items with more than three tags. This is used to refine the data set to more relevant and structured data.

Algorithm 8. cleanupRemoteSiblings

```

input: previously generated trees
map siblings array to remove empty parentheses
filter mapped array to keep only items with more than 3 tags
return the filtered array

```

Algorithm 9 is a utility function that calculates the number of tags in a given string representation of a tree. It counts the occurrences of “(“ and “,” characters, which represent the tags in the tree structure. This count is used as a criterion for filtering data in the `cleanupRemoteSiblings` method.

Algorithm 9. `countTagsInATree`

```
input: tree string
count number of '(' and ',' in treeString
return the total count
```

Algorithm 10 processes an array of unique siblings against all extracted siblings to determine relationships between data blocks. For each unique sibling, it identifies all matching blocks, checks for existing relationships, and then establishes new relationships as specified by the `siblingRelationshipType`. This function is critical in building the relational structure between data blocks based on the specified criteria.

Algorithm 10. `extractBlocksForSiblings`

```
input: unique trees
input: all extracted trees
input: relationship type
for each htmlTree in uniqueSiblings:
    find keys in allExtractedSiblings that match htmlTree
    if more than one key found:
        for each key:
            find block by key
            if block is not found, continue to next key end
            get IDs of existing structural and reverse structural siblings
            calculate difference between keys and existing sibling IDs
            remove the current key from the difference
            for each ID in the calculated difference:
                attach the ID to the block using siblingRelationshipType
            end
        end
    end
end
```

The custom tree notation proposed earlier is designed to be flexible and can be effortlessly expanded to include more data if necessary. This flexibility facilitates a more rigorous comparison of blocks. For example, integrating class names into the tree structure can be advantageous in differentiating between blocks with

similar structures but serving distinct functions. This is because the visual appearance of these blocks is likely to vary, which suggests a probable difference in their class names as well. At times, it may be necessary to incorporate other data into the tree for a more comprehensive analysis. This is particularly crucial when dealing with short trees, as the likelihood of encountering the same tree structure in other parts of the HTML document is significantly higher. A situation that may necessitate the inclusion of additional data is when evaluating menu items. To determine whether a menu is exclusively used for navigating the current page (using hashtags to link to IDs on the current page) or if it links to other pages, it is essential to assess the href attribute data and incorporate it into the tree. This assessment will clearly distinguish between menus that link to local content and those used for navigation to other pages.

The complete process from data collection to its processing is presented in Figure 4.13. The first step is website scraping. It consists of multiple steps that aim to collect all of the data and make it accessible independently from a live website at a future date. This process includes taking screenshots of each content block in mobile, tablet, and desktop views, extracting assets, and extracting both rendered website HTML code and HTML source code. During the scraping step, initial data processing is performed. Possible content blocks are extracted and prepared to be stored in the dataset. After scraping tasks are complete, the collected data is added to the dataset, and the main optimisation steps are performed. First, the min/max boundaries of the blocks are determined; this allows for the discarding of all the ranges except the maximum. Then, direct sibling detection is performed, and relationships are created. Finally, the same process is repeated to detect remote sibling blocks.

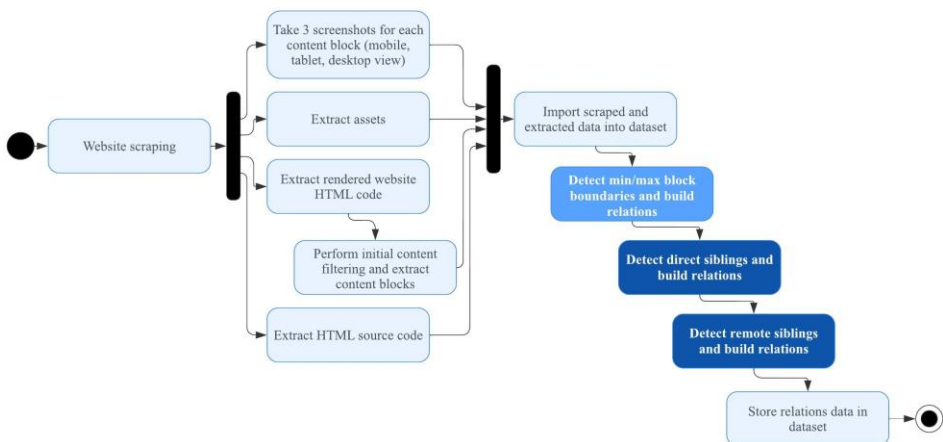


Fig. 4.13. The complete process of processing

4.4. Sibling Block Detection Validation

Contrary to previously performed validations, sibling block detection validation requires more significant amounts of data due to the nature of the feature being tested. Due to this, multiple datasets are used during validation to assess the proposed method's performance better. The goal is to validate whether the proposed method accurately detects sibling blocks across various types of websites, regardless of their design or age. Additionally, the weak points of the proposed method are evaluated for potential improvements.

4.4.1. Data and Principles of the Validation

For the initial validation of the results, two datasets were used: one that was created specifically for this purpose and another that is open source, referred to as "dataset-popular". The open-source dataset was chosen due to its application in related research conducted by other authors. A critical distinction between these datasets is the level of detail; the proposed dataset is considerably more comprehensive, with an average of 91 labelled blocks per website, whereas the open-source dataset averages only 16 blocks per website.

Key characteristics of each dataset include:

- Open source and most used in website segmentation tasks dataset of popular websites by Kreuzer (2023):
 - 50 websites;
 - Manually labelled content blocks (low number of labelled blocks);
 - An average of 16 labelled blocks per website;
 - Data scraped in 2013;
 - Consists of popular websites collected from the Yahoo Directory
- Self-developed dataset;
 - 13 websites;
 - Content blocks are manually labelled by participants during a labelling experiment;
 - Duplicate labelled blocks were removed;
 - An average of 91 labelled blocks per website;
 - Data scraped in 2022–2023.

Following the application of remote sibling detection to the pre-processed data, a notable decrease in the number of content blocks selected for labelling by the proposed technique was observed. This is mainly because the blocks recognised as remote siblings do not necessitate individual labelling. Only a single

initial block needs to be labelled, and the remote siblings automatically adopt this label due to the established relationship between the blocks.

Another aspect that contributes to this reduction is the proposed approach of comparing blocks as entire subtrees. This approach allows for the omission of all child blocks of remote siblings, as those blocks are expected to be included within the chosen block. Nevertheless, this method has certain limitations, which will be addressed later in the dissertation.

4.4.2. Results of the Proposed Method Testing

Table 4.6 summarises the relationship between user-labelled blocks and different phases of website data optimisation for the labelling process. Part of the data used in the table below originates from research described in the previous sections. Furthermore, given the considerable number of tags on YouTube, a separate summary row labelled “WYT” (Without YouTube) was added that excludes YouTube from the calculations.

Table 4.6. Performance comparison of optimisation steps for content blocks’ reduction in comparison with user-labelled blocks

Web Page No.	No. of web page tags	User-labelled blocks	User-labelled blocks ratio vs No. of tags	Reduced content blocks	User-labelled blocks ratio vs reduced content blocks	Content blocks after remote siblings’ optimisation	User-labelled blocks ratio vs remote siblings
1	521	89	17%	238	37.39%	147	60.54%
2	531	84	16%	192	43.75%	102	82.35%
3	393	64	16%	136	47.06%	114	56.14%
4	24039	75	0.31%	3458	2.17%	391	19.18%
5	981	61	6.21%	256	23.83%	91	67.03%
6	1087	69	6.34%	281	24.56%	126	54.76%
7	826	74	8.95%	285	25.96%	154	48.05%
8	1523	98	6.43%	472	20.76%	185	52.97%
9	111	17	15.31%	45	37.78%	24	70.83%
10	2474	203	8.2%	778	26.09%	394	51.52%

End of Table 4.6

Web Page No.	No. of web page tags	User-labelled blocks	User-labelled blocks ratio vs No. of tags	Reduced content blocks	User-labelled blocks ratio vs reduced content blocks	Content blocks after remote siblings' optimisation	User-labelled blocks ratio vs remote siblings
11	768	87	11.32%	168	51.79%	124	70.16%
12	4990	144	2.88%	1346	10.70%	764	18.85%
13	2248	114	5.07%	436	26.15%	142	80.28%
Total	40492	1179	2.9%	8091	14.57%	2758	42.75%
WYT	16453	1104	6.71%	4633	23.83%	2367	46.64%

The “Number of tags” column specifies the total blocks present on a webpage. This count is extracted from the webpage before performing any optimisations or validations. The “User labelled blocks” column indicates the users of the total block marked during the labelling experiment. This count, reviewed post-experiment, includes unique content blocks, excluding duplicates.

The “User labelled blocks versus a number of tags” ratio illustrates the proportion of the total page content that users would have marked if no content block optimisations were performed before the labelling step. This ratio uncovers a significant amount of content blocks that would be missed during the labelling process, resulting in a substantially prolonged labelling time to cover all content blocks on a webpage. For smaller web pages, manually labelled blocks constituted 15–17% of the total number of tags; however, for larger websites, this percentage notably decreased, plummeting to as low as 0.31% for YouTube.

Before embarking on this research, a series of optimisation strategies had already been devised to minimise the number of content blocks that needed to be labelled for extracting and processing a website’s content. The column labelled “Reduced content blocks” represents the number of remaining content blocks after implementing these optimisation strategies. Additionally, the column titled “Content blocks after remote siblings’ optimisation” signifies the number of content blocks that persisted after further refinement of the reduced content blocks. This refinement involved identifying remote sibling relationships and eliminating all but one remote sibling from the subsequent processing stages. This additional step was crucial in optimising the labelling process further, ensuring that only the most necessary blocks were carried forward for the final stages of analysis.

The inclusion of remote sibling detection capability significantly streamlined the number of content blocks, aligning more closely with the count of blocks that users manually tagged. As a result, the portion of user-labelled blocks surged to 42.75% (or 46.64% when YouTube is excluded) from a mere 14.57% (or 23.83%). This considerable increase underlines that the refined process requires notably less effort for labelling optimised content blocks than the effort needed in previous methodologies. This advancement is a significant milestone as it enhances the efficiency of the process, minimising the workload and, consequently, the time and resources required to complete the task.

Table 4.7 showcases a comparative analysis of the outcomes yielded at various content block minimisation and refinement phases.

Table 4.7. Performance of content blocks' reduction steps in comparison to the total number of blocks on a page

Web Page No.	No. of web page tags	No. of potential content blocks	Filtered content blocks	Reduced content blocks	Reduced %	Content blocks after remote siblings' optimisation	Total blocks reduction %
1	521	502	469	238	49.25%	147	71.79%
2	531	518	494	192	61.13%	102	80.79%
3	393	379	305	136	55.41%	114	70.99%
4	24039	23983	5743	3458	49.90%	391	98.37%
5	981	899	511	256	60.86%	91	90.72%
6	1087	919	718	281	53.43%	126	88.41%
7	826	763	612	285	51.59%	154	81.36%
8	1523	1433	975	472	45.12%	185	87.85%
9	111	87	82	45	64.21%	24	78.38%
10	2474	2410	2174	778	53.85%	394	84.07%
11	768	702	364	168	53.04%	124	83.85%
12	4990	4375	2866	1346	67.56%	764	84.69%
13	2248	2125	1344	436	39.79%	142	93.68%
Total	40492	39095	16657	8091	55.24%	2758	84.23%
WYT	16453	15112	10914	4633	55.45%	2367	83.05%

The prevailing algorithm can be divided into four principal phases. Initially, the algorithm distinguishes potential content blocks from the remainder. This initial phase is relatively uncomplicated, as it merely involves selecting the contents within the body tag and discarding all other elements. Subsequently, the algorithm filters the content tags obtained during the initial phase, eliminating tags that lack pertinent content. These irrelevant tags may encompass elements such as link, meta, style, and script, among others. This step is crucial to ensure that the algorithm is processing and analysing only the most relevant and essential parts of the webpage, thereby optimising its performance and the accuracy of the results it generates.

In the third phase, a more comprehensive examination is conducted, and the quantity of content blocks is further minimised by discarding variations in block length and direct sibling blocks. The primary block in a relationship tree is retained for both scenarios, while the rest are removed. In the final step, remote sibling detection is executed on the leftover blocks, and once more, all but the primary siblings from a remote sibling relationship tree are eliminated.

Incorporating the remote sibling optimisation phase allows for a drastic reduction in the overall quantity of content blocks, on average, by 84.23% (83.05% excluding YouTube), compared to the 55.24% (55.45% excluding YouTube) reduction achieved with the previous method. This enhancement underscores the effectiveness of the refined approach, showing that it significantly reduces the workload and optimises the entire process more effectively than earlier techniques.

Additionally, content blocks' reduction logic was applied to dataset-popular (Kreuzer, 2023), which consists of 50 websites. Results for this are provided in Table 4.8. For this comparison, the table referencing the number of labelled blocks in the dataset-popular is not provided because this dataset contains a small number of labels for each page and does not cover all the content blocks. Therefore, the metric of labelled blocks and detected blocks is excluded. Nevertheless, it is still possible to evaluate the number of blocks the algorithm suggests for labelling against the total number of blocks on the page. Observations indicate that similar to the previous experiment, there is an average overall reduction of 81.53% in content blocks. This further demonstrates the effectiveness of the proposed method in minimising the number of blocks requiring manual labelling.

Table 4.8. Performance of content blocks' reduction steps in comparison to the total number of blocks on a page when using dataset-popular data

Web Page No.	No. of web page tags	No. of potential content blocks	Filtered content blocks	Reduced content blocks	Reduced %	Content blocks after remote sibling optimisation	Total blocks reduction %
1	2	3	4	5	6	7	8
1	1500	1475	797	291	80.27%	202	86.31%
2	1337	1290	790	371	71.24%	247	80.85%
3	611	597	517	227	61.98%	140	76.55%
4	1050	1009	493	180	82.16%	125	87.61%
5	364	234	156	80	65.81%	59	74.79%
6	459	427	291	153	64.17%	90	78.92%
7	1197	1142	850	299	73.82%	231	79.77%
8	1420	1368	843	257	81.21%	212	84.50%
9	146	135	121	65	51.85%	63	53.33%
10	643	601	446	208	65.39%	148	75.37%
11	958	892	307	141	84.19%	112	87.44%
12	576	545	168	74	86.42%	66	87.89%
13	1262	1229	875	520	57.69%	376	69.41%
14	945	880	457	194	77.95%	140	84.09%
15	925	869	749	197	77.33%	146	83.20%
16	1034	932	390	132	85.84%	119	87.23%
17	225	187	157	66	64.71%	55	70.59%
18	1396	1328	1008	472	64.46%	268	79.82%
19	530	472	353	109	76.91%	91	80.72%
20	737	705	544	227	67.80%	150	78.72%
21	917	860	517	183	78.72%	145	83.14%
22	1958	1918	300	90	95.31%	79	95.88%
23	2248	2166	1019	405	81.30%	219	89.89%
24	501	463	170	85	81.64%	77	83.37%
25	1073	1011	748	297	70.62%	106	89.52%
26	280	262	252	128	51.15%	96	63.36%
27	221	203	91	49	75.86%	37	81.77%

End of Table 4.8

1	2	3	4	5	6	7	8
28	482	444	296	126	71.62%	94	78.83%
29	1245	1191	536	268	77.50%	184	84.55%
30	645	598	143	47	92.14%	47	92.14%
31	1405	1357	750	311	77.08%	197	85.48%
32	345	331	195	55	83.38%	38	88.52%
33	985	955	825	269	71.83%	174	81.78%
34	2709	2580	1582	457	82.29%	221	91.43%
35	3734	3720	1801	518	86.08%	227	93.90%
36	375	331	241	84	74.62%	61	81.57%
37	358	318	222	69	78.30%	68	78.62%
38	164	148	143	63	57.43%	45	69.59%
39	43	37	34	14	62.16%	14	62.16%
40	715	683	253	77	88.73%	55	91.95%
41	1422	1375	830	321	76.65%	207	84.95%
42	267	255	152	73	71.37%	62	75.69%
43	748	709	525	244	65.59%	161	77.29%
44	1090	1065	555	309	70.99%	211	80.19%
45	243	682	546	135	80.21%	115	83.14%
46	1199	1140	1004	435	61.84%	202	82.28%
47	284	267	170	71	73.41%	69	74.16%
48	1395	1352	358	139	89.72%	80	94.08%
49	573	515	399	180	65.05%	72	86.02%
50	1046	1023	676	215	78.98%	164	83.97%
Total	45985	44276	25645	9980	77.46%	6567	85.17%
Avg.	919.7	885.52	512.9	199.6	74.26%	131.34	81.53%

4.4.3. Validation Insights for Sibling Block Detection and its Limitations

Currently, during the ongoing development phase, only trees with more than three nodes are being processed. When the threshold was lower, the false-positive detection rate was excessively high with the existing detection criteria. To reduce this threshold, it would be necessary to implement more stringent detection criteria involving the stored trees containing additional information about the block

and possibly its parent to be considered during tree processing and remote sibling detection.

Another constraint is the block similarity threshold, set at 0.92. During the direct sibling detection phase (when all processed blocks have the same parent block), there might be instances where some blocks contain slightly different data. This necessitates a minor modification in the block's HTML tree to represent that additional data accurately. Given the current implementation of sibling detection, only the first block is retained and presented to the user during the labelling phase (although other sibling blocks can be extracted via reverse sibling relation). This implies that some supplementary data might not be labelled during the initial labelling phase. While this might only represent a minor portion of the block's content, it could still be significant. A potential solution could be to present the sibling with the most data during labelling. However, there might be situations where multiple siblings have additional data with varying structures. In such scenarios, the optimal approach is to extract the differences from the sibling blocks and present them during the labelling phase.

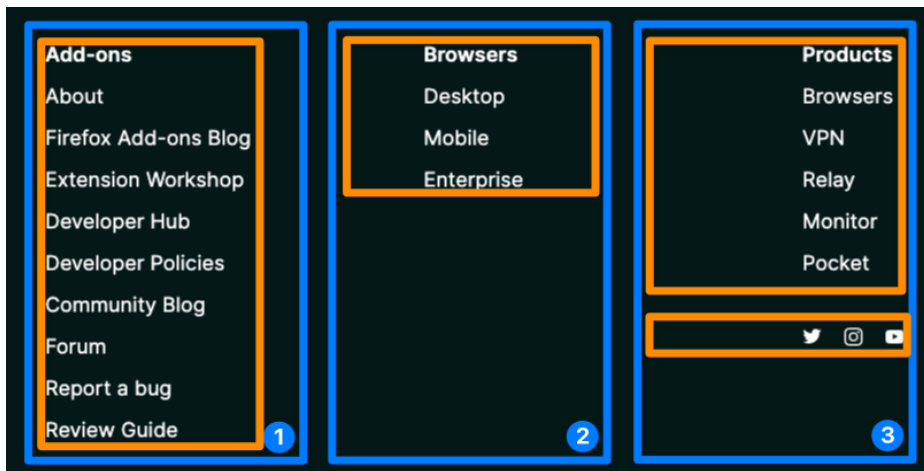


Fig. 4.14. Example HTML block where siblings are not matched due to limitations of detection algorithm

In certain instances, related sibling blocks are not detected due to variations in the number or repeated inner content blocks. This issue resembles the one discussed in the preceding paragraph, although, in this case, the difference exceeds the similarity threshold score. This commonly occurs when processing navigation menus. Multiple navigation menu categories are frequently located in the page footer, displaying links to primary content categories. Often, these menus have varying numbers of items, meaning that the similarity score will fall below the

threshold required for the blocks to be deemed similar during the block comparison.

The example depicted in Figure 4.14 highlights the footer menus of one of the scrutinised websites. Post-processing, the footer menus failed to be recognised as siblings for multiple reasons. The mismatch between the first and second columns was due to the varying lengths of the menus. The current sibling detection algorithm evaluates blocks as entities without additional content processing. Since there is a disparity in the number of links between the menus of the first and second columns, the resulting similarity score needs to be more significant to exceed the established threshold.

Resolving this issue could involve comprehensively examining the contents within these menu blocks. As both menus reside on an identical level in the HTML tree, supplementary tests could be carried out on the block contents to ascertain if those blocks are repetitive and whether only the repetition count varies while the structure remains unaltered. This technique would pair siblings on the same level, even when content lengths differ.

The inability to match the third column arose from its inclusion of two distinct menus: a conventional one and a social media menu. The conventional menu could potentially be paired with other menus utilising a method similar to the one outlined above, which was proposed for aligning the first two columns. Nonetheless, this scenario would necessitate minor adjustments as the menu blocks are enclosed within columns. This results in differing parent blocks, preventing their classification as direct siblings. It would be viable to process them by identifying a shared ancestor and verifying the structural similarity of the processed blocks. Regarding the social media menu, pairing it with other footer menus is improbable, given its unique structure, which incorporates social media icons instead of text links.

4.5. Conclusions of the Fourth Chapter

1. The innovative approach of enhancing the efficacy of content block labelling on web pages by incorporating remote sibling detection and several optimisation steps significantly reduced the number of content blocks necessitating user labelling, optimising and expediting the process. Optimisations are suggestive and can be disregarded by the user performing the labelling in case of an incorrect relationship detection.
2. Utilising remote sibling relationships between content blocks allows for eliminating such blocks from further analysis and assigning the labels of originating siblings to them. This approach reduces the total number of content blocks that require labelling. Based on experiment results, user-

labelled blocks constituted 42.75% (46.64% excluding YouTube) of the total number of blocks, a notable improvement from the 14.57% (23.83% excluding YouTube) attained without remote sibling detection.

3. The proposed method enabled a reduction in the overall content block count by an average of 84.23% (83.05% when excluding YouTube), which surpasses the 55.24% (55.45% without YouTube) reduction achieved with the preceding technique. These results validate the effectiveness of the remote sibling detection approach in substantially enhancing the web page labelling procedure.
 - Certain constraints were observed, such as the failure to identify sibling blocks with varying numbers of repeated inner content block lengths;
 - The established block similarity threshold can sometimes lead to temporary loss of content, but it can be recovered with additional processing since nothing is being discarded;
 - The minimum node count required for tree processing is currently set at three. This results in some blocks not being processed and thus not being eliminated from the total content block count.

To tackle these challenges, further research is recommended, including implementing stricter detection criteria, adjusting the block similarity threshold, and developing advanced techniques to handle structural differences in sibling blocks.

In summary, the research demonstrated that web page labelling efficiency can be significantly improved through remote sibling detection, consequently reducing the content blocks requiring user labelling and streamlining the entire process. The proposed method provides suggestions to the user on which blocks can be combined into sibling groups. In case of incorrect relationship detection, the user can choose to disregard these suggestions and dissolve the groups without losing any additional data. The insights gleaned from this research could inform the development and deployment of analogous systems across various sectors necessitating web content extraction and processing. As web structures and complexities continue to evolve, there exists an encouraging opportunity for further refinement and enhancement of these methodologies.

General Conclusions

1. The analysis of web mining taxonomies revealed that the existing ones are oriented on some specific areas. However, existing taxonomies need to summarise the overall landscape of the area's properties. This does not allow for a complete understanding of the relations between separate areas of the web mining area. The newly proposed web mining taxonomy has five main areas (mining purpose, mining object, web structure, data object, method) with its internal and shared structure. Combining a broader range of areas and adding new components in the proposed taxonomy contributes to a better highlighting of the web mining area.
2. The analysis of existing methods and initial experiments on website content block similarity estimation revealed that HTML block similarity estimation could be more accurate when comparing multiple similarity factors. On average, the developed HTML block similarity method is about 12% more accurate in estimating the same content blocks implemented with different HTML codes. Based on the Wilcoxon Signed-Rank test, the difference in similarity scores is significant when comparing both of the proposed methods to the existing ones. The W-value for proposed methods is 1 and 0, while the critical value for W is 8 or 3 with a significance level of 0.05 or 0.01, respectively. Additionally, when comparing

different pairs, the proposed method is more stable (standard deviation was reduced by 22%).

3. The newly developed website data set structure incorporates all data from existing data sets and adds additional features (screenshots, similarity scores, block relationships, block type hierarchy) oriented toward increased usability of the same dataset. Executed data mining and data set record insertion experiments revealed that the structure is suitable to cover data of the existing datasets. Additionally, it leads to a better understanding of the website's hierarchical structure.
4. Proposed and refined methods to detect website content blocks and present them in a hierarchical structure with aggregated ranges. The executed experiments revealed that in manual labelling, humans are capable of labelling just 3–7% of tags. The proposed method reduces the number of blocks presented for labelling by 83% on average while ensuring that all 100% of blocks receive labelling by utilising block relationships. This indicates that the proposed methods are suitable as a tool for the reduction of website content block labelling effort, keeping the needed accuracy and providing a base for labelling tool development.

References

- Abbasi, B. U. D., Fatima, I., Mukhtar, H., Khan, S., Alhumam, A., & Ahmad, H. F. (2022). Autonomous schema markups based on intelligent computing for search engine optimization. *PeerJ Computer Science*, 8, e1163. <https://doi.org/10.7717/peerj-cs.1163>
- Agichtein, E., & Gravano, L. (2003, March). Querying text databases for efficient information extraction. In *Proceedings 19th International Conference on Data Engineering (Cat. No. 03CH37405)* (pp. 113–124). IEEE.
- Akpınar, M. E., & Yesilada, Y. (2013). Vision based page segmentation algorithm: Extended and perceived success. In *Current Trends in Web Engineering: ICWE 2013 International Workshops ComposableWeb, QWE, MDWE, DMSSW, EMotions, CSE, SSN, and PhD Symposium, Aalborg, Denmark, July 8-12, 2013. Revised Selected Papers 13* (pp. 238–252). Springer International Publishing. https://doi.org/10.1007/978-3-319-04244-2_22
- Alcic, S., & Conrad, S. (2011, May). Page segmentation by web content clustering. In *Proceedings of the International Conference on Web Intelligence, Mining and Semantics* (pp. 1–9). <https://doi.org/10.1145/1988688.1988717>
- Algergawy, A., Nayak, R., & Saake, G. (2010). Element similarity measures in XML schema matching. *Information Sciences*, 180(24), 4975–4998. <https://doi.org/10.1016/j.ins.2010.08.022>
- Al-Ghuribi, S. M., & Alshomrani, S. (2013, June). A comprehensive survey on web content extraction algorithms and techniques. In *2013 International Conference on*

- Information Science and Applications (ICISA)* (pp. 1–5). IEEE. <https://doi.org/10.1109/ICISA.2013.6579445>
- Aljawarneh, S. A., Elkobaisi, M. R., & Maatuk, A. M. (2017). A new agent approach for recognizing research trends in wearable systems. *Computers & Electrical Engineering*, 61, 275–286. <https://doi.org/10.1016/j.compeleceng.2016.12.003>
- Amarnadh, S., Reddy, P. P., & Murthy, N. V. E. S. (2019). FreeHand Sketch-based Authenticated Security System using Sequence Matcher. *Int. J. Adv. Sci. Technol*, 29, 6663–6679.
- Anami, B. S., Wadawadagi, R. S., & Pagi, V. B. (2014). Machine learning techniques in Web content mining: a comparative analysis. *Journal of Information & Knowledge Management*, 13(01), 1450005. <https://doi.org/10.1142/S0219649214500051>
- Andrew, J., Ferrari, S., Maurel, F., Dias, G., & Giguët, E. (2019, September). Web page segmentation for non visual skimming. In *The 33rd Pacific Asia Conference on Language, Information and Computation (PACLIC 33)*, (pp. 423–431).
- Arotaritei, D., & Mitra, S. (2004). Web mining: a survey in the fuzzy framework. *Fuzzy Sets and Systems*, 148(1), 5–19. <https://doi.org/10.1016/j.fss.2004.03.003>
- Bartík, V. (2010, August). Text-based web page classification with use of visual information. In *2010 International Conference on Advances in Social Networks Analysis and Mining* (pp. 416–420). IEEE. <https://doi.org/10.1109/ASONAM.2010.34>
- Besbes, G., & Baazaoui-Zghal, H. (2017). Personalized and context-aware retrieval based on fuzzy ontology profiling. *Integrated Computer-Aided Engineering*, 24(1), 87–103. <https://doi.org/10.3233/ICA-160525>
- Bing, L., Guo, R., Lam, W., Niu, Z. Y., & Wang, H. (2014, July). 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* (pp. 767–776). <https://doi.org/10.1145/2600428.2609630>
- Bookstein, A., Kulyukin, V. A., & Raita, T. (2002). Generalized hamming distance. *Information Retrieval*, 5, 353–375. <https://doi.org/10.1023/A:1020499411651>
- Burget, R., & Rudolfova, I. (2009, April). Web page element classification based on visual features. In *2009 First Asian Conference on Intelligent Information and Database Systems* (pp. 67–72). IEEE. <https://doi.org/10.1109/ACIIDS.2009.71>
- Buttler, D. (2004). *A short survey of document structure similarity algorithms* (No. UCRL-CONF-202728). Lawrence Livermore National Lab. (LLNL), Livermore, CA (United States).
- Cai, D., Yu, S., Wen, J. R., & Ma, W. Y. (2003). *Vips: a vision-based page segmentation algorithm*, Tech. Rep. MSR-TR-2003-79, Microsoft Research, Redmond, WA (United States).
- Chang, C. H., Kayed, M., Girgis, M. R., & Shaalan, K. F. (2006). A survey of web information extraction systems. *IEEE transactions on knowledge and data engineering*, 18(10), 1411–1428. <https://doi.org/10.1109/TKDE.2006.152>

- Chbeir, R., & Ma, Y. (2005, September). Content and structure based approach for XML similarity. In *The Fifth International Conference on Computer and Information Technology (CIT'05)* (pp. 136–140). IEEE.
- Cheng, S. C., & Lu, C. A. (2019, July). Retrieving Articles and Image Labeling Based on Relevance of Keywords. In *2019 International Conference on Machine Learning and Cybernetics (ICMLC)* (pp. 1–6). IEEE. <https://doi.org/10.1109/ICMLC48188.2019.8949205>
- Cooley, R., Mobasher, B., & Srivastava, J. (1997, November). Web mining: Information and pattern discovery on the world wide web. In *Proceedings ninth IEEE international conference on tools with artificial intelligence* (pp. 558–567). IEEE. <https://doi.org/10.1109/TAI.1997.632303>
- Cormier, M., Cohen, R., Mann, R., Rahim, K., & Wang, D. (2015). A robust vision-based framework for screen readers. In *Computer Vision-ECCV 2014 Workshops: Zurich, Switzerland, September 6-7 and 12, 2014, Proceedings, Part III 13* (pp. 555–569). Springer International Publishing. https://doi.org/10.1007/978-3-319-16199-0_39
- Cormier, M., Moffatt, K., Cohen, R., & Mann, R. (2016). Purely vision-based segmentation of web pages for assistive technology. *Computer Vision and Image Understanding*, 148, 46–66. <https://doi.org/10.1016/j.cviu.2016.02.007>
- Crescenzi, V., Mecca, G., & Merialdo, P. (2001, September). Roadrunner: Towards automatic data extraction from large web sites. In *SIGMOD '02: Proceedings of the 2002 ACM SIGMOD international conference on Management of data* (Vol. 1, pp. 109–118). <https://doi.org/10.1145/564691.564778>
- Debnath, S., Mitra, P., & Giles, C. L. (2005a). Automatic extraction of informative blocks from webpages. In *Proceedings of the 2005 ACM symposium on Applied computing* (pp. 1722–1726).
- Debnath, S., Mitra, P., & Giles, C. L. (2005b). Identifying content blocks from web documents. In *Foundations of Intelligent Systems: 15th International Symposium, ISMIS 2005, Saratoga Springs, NY, USA, May 25-28, 2005. Proceedings 15* (pp. 285–293). Springer Berlin Heidelberg. https://doi.org/10.1007/11425274_30
- del Pozo, A., Pazos, F., & Valencia, A. (2008). Defining functional distances over gene ontology. *BMC bioinformatics*, 9, 1–15. <https://doi.org/10.1186/1471-2105-9-50>
- Demaine, E. D., Mozes, S., Rossman, B., & Weimann, O. (2009). An optimal decomposition algorithm for tree edit distance. *ACM Transactions on Algorithms (TALG)*, 6(1), 1–19. <https://doi.org/10.1145/1644015.1644017>
- Dias, S., & Gadge, J. (2014). Identifying informative web content blocks using web page segmentation. *entropy*, 1, 2. <https://doi.org/10.5120/ijais14-451129>
- Dong, H., Hussain, F. K., & Chang, E. (2009). A hybrid concept similarity measure model for ontology environment. In *On the Move to Meaningful Internet Systems: OTM 2009 Workshops: Confederated International Workshops and Posters, ADI, CAMS, EI2N, ISDE, IWSSA, MONET, OnToContent, ODIS, ORM, OTM Academy, SWWS, SEMELS, Beyond SAWSDL, and COMBEK 2009, Vilamoura, Portugal, November 1-6, 2009. Proceedings* (pp. 848–857). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-05290-3_103

- Eldirdiery, H. F., & Ahmed, A. H. (2015). *Web document segmentation for better extraction of information: a review*. *International Journal of Computer Applications* (0975–8887), Vol. 110(3), (pp. 24–28). <https://doi.org/10.5120/19297-0734>
- Escalona, Y. E. (2019). *Algorithms for Table Structure Recognition* [Master's thesis, Pontifical Catholic University of Rio de Janeiro]. Academia.edu Repository. <https://www.academia.edu/download/86509918/48789.pdf>
- Fauzi, F., Hong, J. L., & Belkhatir, M. (2009, October). Webpage segmentation for extracting images and their surrounding contextual information. In *Proceedings of the 17th ACM international conference on Multimedia* (pp. 649–652). <https://doi.org/10.1145/1631272.1631379>
- Feng, H., Zhang, W., Wu, H., & Wang, C. J. (2016, November). Web page segmentation and its application for web information crawling. In *2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI)* (pp. 598–605). IEEE. <https://doi.org/10.1109/ICTAI.2016.0097>
- Ferrara, E., De Meo, P., Fiumara, G., & Baumgartner, R. (2014). Web data extraction, applications and techniques: A survey. *Knowledge-based systems*, 70, 301–323. <https://doi.org/10.1016/j.knosys.2014.07.007>
- Flesca, S., Manco, G., Masciari, E., Pontieri, L., & Pugliese, A. (2005). Fast detection of XML structural similarity. *IEEE Transactions on Knowledge and Data Engineering*, 17(2), 160–175. <https://doi.org/10.1109/TKDE.2005.27>
- Ghaemmaghani, S. S. S., & Miller, J. (2022). Integrated-Block: A New Combination Model to Improve Web Page Segmentation. *Journal of Web Engineering*, 1103–1144. <https://doi.org/10.13052/jwe1540-9589.2146>
- Gowda, T., & Mattmann, C. A. (2016, July). Clustering web pages based on structure and style similarity (application paper). In *2016 IEEE 17th International conference on information reuse and integration (IRI)* (pp. 175–180). IEEE. <https://doi.org/10.1109/IRI.2016.30>
- Grigalis, T., & Čenys, A. (2014). Using XPath of inbound links to cluster template-generated web pages. *Computer Science and Information Systems*, 11(1), 111–131. <https://doi.org/10.2298/CSIS130416020G>
- Grigera, J., Gardey, J. C., Rossi, G., & Garrido, A. (2020, November). Flexible Detection of Similar DOM Elements. In *International Conference on Web Information Systems and Technologies* (pp. 174–195). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-031-24197-0_10
- Haase, M. (2023, March 28). Simple heuristic for measuring web page similarity (& data set). *GitHub*. <https://github.com/TeamHG-Memex/page-compare>
- Hashemi, M. (2020). Web page classification: a survey of perspectives, gaps, and future directions. *Multimedia Tools and Applications*, 79(17–18), 11921–11945. <https://doi.org/10.1007/s11042-019-08373-8>
- Jain, S., Rawat, R., & Bhandari, B. (2017, November). A survey paper on techniques and applications of web usage mining. In *2017 International Conference on Emerging Trends*

- in *Computing and Communication Technologies (ICETCCT)* (pp. 1–6). IEEE. <https://doi.org/10.1109/ICETCCT.2017.8280343>
- Jayamalini, K., & Ponnavaikko, M. (2017, February). Research on web data mining concepts, techniques and applications. In *2017 International Conference on Algorithms, Methodology, Models and Applications in Emerging Technologies (ICAMMAET)* (pp. 1–5). IEEE. <https://doi.org/10.1109/ICAMMAET.2017.8186676>
- Jayashree, S. R., Dias, G., Andrew, J. J., Saha, S., Maurel, F., & Ferrari, S. (2022). Multimodal Web Page Segmentation Using Self-organized Multi-objective Clustering. *ACM Transactions on Information Systems (TOIS)*, 40(3), 1–49. <https://doi.org/10.1145/3480966>
- Jiang, Z., Yin, H., Wu, Y., Lyu, Y., Min, G., & Zhang, X. (2019). Constructing novel block layouts for webpage analysis. *ACM Transactions on Internet Technology (TOIT)*, 19(3), 1–18. <https://doi.org/10.1145/3326457>
- Kiesel, J., Kneist, F., Meyer, L., Komlossy, K., Stein, B., & Potthast, M. (2020, October). Web page segmentation revisited: Evaluation framework and dataset. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management* (pp. 3047–3054). <https://doi.org/10.1145/3340531.3412782>
- Kiesel, J., Meyer, L., Kneist, F., Stein, B., & Potthast, M. (2021, March). An empirical comparison of web page segmentation algorithms. In *European Conference on Information Retrieval* (pp. 62–74). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-030-72240-1_5
- Kohlschütter, C., & Nejdl, W. (2008, October). A densitometric approach to web page segmentation. In *Proceedings of the 17th ACM conference on Information and knowledge management* (pp. 1173–1182). <https://doi.org/10.1145/1458082.1458237>
- Kosala, R., & Blockeel, H. (2000). Web mining research: A survey. *ACM Sigkdd Explorations Newsletter*, 2(1), 1–15. <https://doi.org/10.1145/360402.360406>
- Kreuzer, R., Hage, J., & Feelders, A. (2015). A quantitative comparison of semantic web page segmentation approaches. In *Engineering the Web in the Big Data Era: 15th International Conference, ICWE 2015, Rotterdam, The Netherlands, June 23-26, 2015, Proceedings 15* (pp. 374–391). Springer International Publishing. https://doi.org/10.1007/978-3-319-19890-3_24
- Kreuzer, R. (2023, March 28) *A dataset of popular pages (taken from dir.yahoo.com) with manually marked up semantic blocks*. <https://github.com/rkrzr/dataset-popular/>
- Kumar, M., & Meenu, M. (2017). A survey on pattern discovery of web usage mining. *International Journal of Advance Research, Ideas and Innovations in Technology*, 3(1), 379–385.
- Kuppusamy, K. S., & Aghila, G. (2012). A model for Web page usage mining based on segmentation. *International Journal of Computer Science and Information Technologies*, Vol. 2(2), 2011, (pp. 1144–1148). ISSN: 0975–9646 <https://doi.org/10.48550/arXiv.1202.2622>

- Laender, A. H., Ribeiro-Neto, B. A., Da Silva, A. S., & Teixeira, J. S. (2002). A brief survey of web data extraction tools. *ACM Sigmod Record*, 31(2), 84–93. <https://doi.org/10.1145/565117.565137>
- Li, Y., & Yang, J. (2009, April). A novel method to extract informative blocks from web pages. In *2009 International Joint Conference on Artificial Intelligence* (pp. 536–539). IEEE. <https://doi.org/10.1109/JCAI.2009.156>
- Lin, S. H., & Ho, J. M. (2002, July). Discovering informative content blocks from web documents. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 588–593). <https://doi.org/10.1145/775047.775134>
- Lin, Y., (2023, March 28) *HTML::Similarity– Calculate the structural similarity between two HTML documents*. <https://metacpan.org/pod/HTML::Similarity/>
- Liu, B., & Chen-Chuan-Chang, K. (2004). Special issue on web content mining. *Acm Sigkdd explorations newsletter*, 6(2), 1–4. <https://doi.org/10.1145/1046456.1046457>
- Lovaraju, D., & Devi, G. L. (2017). An Ontology Like Model for gathering Personalized Web Information. *International Journal of Advanced Research in Computer Science*, 8(3), (pp. 401–403).
- Manickam, Y., Rajalakshmi, S. U., & Uma, S. (2022). Extraction of information from web page using content mining approach. *International Journal of Scientific Research in Engineering & Technology*, 2(2), 13–15.
- Manugunta, R. K., Maskeliūnas, R., & Damaševičius, R. (2022). Deep Learning Based Semantic Image Segmentation Methods for Classification of Web Page Imagery. *Future Internet*, 14(10), 277. <https://doi.org/10.3390/fi14100277>
- Marca, E., (2023, March 28) *Compare html similarity using structural and style metrics*, <https://github.com/matiskay/html-similarity>
- Martinez-Rodriguez, J. L., Hogan, A., & Lopez-Arevalo, I. (2020). Information extraction meets the semantic web: a survey. *Semantic Web*, 11(2), 255–335. <https://doi.org/10.3233/SW-180333>
- Nguyen, C. K., Likforman-Sulem, L., Moissinac, J. C., Faure, C., & Lardon, J. (2012, March). Web document analysis based on visual segmentation and page rendering. In *2012 10th IAPR International Workshop on Document Analysis Systems* (pp. 354–358). IEEE. <https://doi.org/10.1109/DAS.2012.95>
- Niwattanakul, S., Singthongchai, J., Naenudorn, E., & Wanapu, S. (2013, March). Using of Jaccard coefficient for keywords similarity. In *Proceedings of the international multi-conference of engineers and computer scientists*, 1 (6), 380–384.
- Norouzi, M., Fleet, D. J., & Salakhutdinov, R. R. (2012). Hamming distance metric learning. *Proceedings of the 25th International Conference on Neural Information Processing Systems*, Vol. 1 (pp. 1061–1069).
- Nyein, S. S. (2011, March). Mining contents in Web page using cosine similarity. In *2011 3rd International Conference on Computer Research and Development* (Vol. 2, pp. 472–475). IEEE.

- Pekar, V., & Staab, S. (2002). Taxonomy learning-factoring the structure of a taxonomy into a semantic classification decision. In *COLING 2002: The 19th International Conference on Computational Linguistics* (pp. 1–7). <https://doi.org/10.3115/1072228.1072318>
- Roldán, J. C., Jiménez, P., & Corchuelo, R. (2020). On extracting data from tables that are encoded using HTML. *Knowledge-Based Systems*, 190, 105157. <https://doi.org/10.1016/j.knosys.2019.105157>
- Sanoja, A., & Gançarski, S. (2014, April). Block-o-matic: A web page segmentation framework. In *2014 international conference on multimedia computing and systems (IC-MCS)* (pp. 595–600). IEEE. <https://doi.org/10.1109/ICMCS.2014.6911249>
- Sanoja, A., & Gançarski, S. (2015). Web page segmentation evaluation. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing (SAC '15)* (pp. 753–760). Association for Computing Machinery. <https://doi.org/10.1145/2695664.2695786>
- Sanoja, A., & Gançarski, S. (2017). Migrating web archives from html4 to html5: A block-based approach and its evaluation. In *Advances in Databases and Information Systems: 21st European Conference, ADBIS 2017, Nicosia, Cyprus, September 24-27, 2017, Proceedings 21* (pp. 375–393). Springer International Publishing. https://doi.org/10.1007/978-3-319-66917-5_25
- Sharma, K., Shrivastava, G., & Kumar, V. (2011, April). Web mining: Today and tomorrow. In *2011 3rd International Conference on Electronics Computer Technology* (Vol. 1, pp. 399–403). IEEE. <https://doi.org/10.1109/ICECTECH.2011.5941631>
- Shasha, D., & Zhang, K. (1990). Fast algorithms for the unit cost editing distance between trees. *Journal of algorithms*, 11(4), 581–621. [https://doi.org/10.1016/0196-6774\(90\)90011-3](https://doi.org/10.1016/0196-6774(90)90011-3)
- Shu, Z., & Li, X. (2022). Automatic extraction of web page text information based on network topology coincidence degree. *Wireless Communications and Mobile Computing*, 2022. <https://doi.org/10.1155/2022/9220661>
- Silpa, N., & Rao, V. M. (2022). Machine learning-based optimal segmentation system for web data using Genetic approach. *Journal of Theoretical and Applied Information Technology*, 100(11), 3552–3561.
- Siteefy (2023, August 25) *How Many Websites Are There in the World?* <https://siteefy.com/how-many-websites-are-there/>
- Sleiman, H. A., & Corchuelo, R. (2013). Tex: An efficient and effective unsupervised web information extractor. *Knowledge-Based Systems*, 39, 109–123. <https://doi.org/10.1016/j.knosys.2012.10.009>
- Sun, G., Zhang, Z., Cheng, Y., & Chai, T. (2022). Adaptive segmented webpage text based malicious website detection. *Computer Networks*, 216, 109236. <https://doi.org/10.1016/j.comnet.2022.109236>
- Tang, Y., Cai, Y., & Mamoulis, N. (2015). Scaling similarity joins over tree-structured data. *Proceedings of the VLDB Endowment*, 8(11), 1130–1141. <https://doi.org/10.14778/2809974.2809976>

- Tekli, J., Chbeir, R., & Yetongnon, K. (2009). An overview on XML similarity: Background, current trends and future directions. *Computer science review*, 3(3), 151–173. <https://doi.org/10.1016/j.cosrev.2009.03.001>
- Utiu, N., & Ionescu, V. S. (2018, September). Learning web content extraction with DOM features. In *2018 IEEE 14th International Conference on Intelligent Computer Communication and Processing (ICCP)* (pp. 5–11). IEEE. <https://doi.org/10.1109/ICCP.2018.8516632>
- Vandenbussche, P. Y., Atemezeng, G. A., Poveda-Villalón, M., & Vatan, B. (2017). Linked Open Vocabularies (LOV): a gateway to reusable semantic vocabularies on the Web. *Semantic Web*, 8(3), 437–452. <https://doi.org/10.3233/SW-160213>
- Vargas, A. S. (2015). *Web page segmentation, evaluation and applications* [Doctoral dissertation, Pierre and Marie Curie University] HAL Archive. <https://theses.hal.science/tel-01128002/document>
- Wang, Q., Fang, Y., Ravula, A., Feng, F., Quan, X., & Liu, D. (2022, April). Webformer: The web-page transformer for structure information extraction. In *Proceedings of the ACM Web Conference 2022* (pp. 3124–3133). <https://doi.org/10.1145/3485447.3512032>
- Win, C. S., & Thwin, M. M. S. (2014). Web page segmentation and informative content extraction for effective information retrieval. *Int. J. Comput. Commun. Eng. Res*, 2, 35–45.
- Wu, Y. C. (2016). Language independent web news extraction system based on text detection framework. *Information Sciences*, 342, 132–149. <https://doi.org/10.1016/j.ins.2015.12.025>
- Xiang, P., Yang, X., & Shi, Y. (2007, July). Web page segmentation based on gestalt theory. In *2007 IEEE International Conference on Multimedia and Expo* (pp. 2253–2256). IEEE. <https://doi.org/10.1109/ICME.2007.4285135>
- Xie, W., Zheng, W., Tang, P., & Ting, Y. (2022, April). Design and Implementation of Web Information Extraction System Based on Crawler. In *EMIE 2022; The 2nd International Conference on Electronic Materials and Information Engineering* (pp. 1–6). VDE.
- Xu, Z., & Miller, J. (2016). Identifying semantic blocks in Web pages using Gestalt laws of grouping. *World Wide Web*, 19, 957–978. <https://doi.org/10.1007/s11280-015-0370-0>
- Yang, Y., & Zhang, H. (2001, September). HTML page analysis based on visual cues. In *Proceedings of Sixth International Conference on Document Analysis and Recognition* (pp. 859–864). IEEE. <https://doi.org/10.1109/ICDAR.2001.953909>
- Yao, J., & Zuo, X. (2023, March 28). CS229: Machine Learning: A machine learning approach to webpage content extraction. *Stanford*. <https://cs229.stanford.edu/proj2013/YaoZuo-AMachineLearningApproachToWebpageContentExtraction.pdf>
- Yujian, L., & Bo, L. (2007). A normalized Levenshtein distance metric. *IEEE transactions on pattern analysis and machine intelligence*, 29(6), 1091–1095. <https://doi.org/10.1109/TPAMI.2007.1078>

Youssefi, A. H., Duke, D. J., & Zaki, M. J. (2004, May). Visual web mining. In *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters* (pp. 394–395). <https://doi.org/10.1145/1013367.1013492>

Zeleny, J., Burget, R., & Zendulka, J. (2017). Box clustering segmentation: A new method for vision-based web page preprocessing. *Information Processing & Management*, 53(3), 735–750. <https://doi.org/10.1016/j.ipm.2017.02.002>

Zhang, X., Zhang, Y., He, J., & Cobia, F. (2013, November). Vision-Based Web Page Block Segmentation and Informative Block Detection. In *2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)* (Vol. 3, pp. 265–269). IEEE. <https://doi.org/10.1109/WI-IAT.2013.194>

List of Scientific Publications by the Author on the Topic of the Dissertation

Papers in the Reviewed Scientific Journals

Griazev, K.; & Ramanauskaitė, S. (2023). Web page content block identification with extended block properties // *Applied sciences: Special issue: New horizons in web search, web data mining, and web-based applications*. Basel: MDPI. eISSN 2076-3417. 13 (9), art. no. 5680, p. 1–16. <https://doi.org/10.3390/app13095680>

Griazev, K.; & Ramanauskaitė, S. (2021). Multi-purpose dataset of webpages and Its content blocks: Design and structure validation // *Applied Sciences*. Basel: MDPI. ISSN 2076-3417., vol. 11, iss. 8, art. no. 3319, p. 1-13. <https://doi.org/10.3390/app11083319>

Papers in Other Editions

Griazev, K.; & Ramanauskaitė, S. (2018). HTML block similarity estimation. In D. Navakas, A. Romanovs, & D. Plonis (Eds.), *IEEE. 6th workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE)*, November 8–10, 2018 Vilnius, Lithuania: proceedings. New York: IEEE, 2018. ISBN 9781728120003. eISBN 9781728119991. (pp. 1–4). <https://doi.org/10.1109/AIEEE.2018.8592241>

Griazev, K.; & Ramanauskaitė, S. (2018). Web mining taxonomy. In *2018 Open Conference of Electrical, Electronic and Information Sciences (eStream)* (pp. 1–4). 26 April 2018, Vilnius, Lithuania. New York: IEEE, 2018. ISBN 9781538667385. eISBN 9781538667378. <https://doi.org/10.1109/eStream.2018.8394124>

Ramanauskaitė, S., & Griazev, K. (2018). Similarity estimation for HTML code blocks = HTML kodo blokų panašumo vertinimas. *Jaunųjų mokslininkų darbai = Journal of young scientists*, 48 (1), 30–36. Šiaulių universiteto leidykla. ISSN 1648-8776. eISSN 2424-3345. <https://doi.org/10.21277/jmd.v48i1.219>

Summary in Lithuanian

Įvadas

Problemos formulavimas

Dirbtinio intelekto sprendimai vis dažniau taikomi įvairiems sprendimams priimti ir veiksmams automatizuoti. Daugeliu atvejų šie metodai paremti duomenų rinkiniais, pagal kuriuos modeliai yra apmokomi. Todėl sukurtų modelių tikslumas labai priklauso nuo turimų duomenų rinkinio, jo dydžio ir detalumo. Tačiau šiuo metu kokybiškų ir prieinamų duomenų kiekis yra ribotas. Organizacijos vis dar nenoriai dalinasi atvirais duomenimis, o dalis duomenų turi būti žmogaus pažymėta, todėl turimų pažymėtų duomenų kiekis yra nepakankamas.

Viena iš sričių, kurioje naudojami labai dideli duomenų kiekiai, yra internetiniai tinklalapiai. Jų skaičius pasauliniame žiniatinklyje yra milžiniškas, o turinys nuolat atnaujinamas. Tačiau internetiniai tinklalapiai pagrįsti technologija, kuri yra skirta duomenims pateikti žmogui, o ne kompiuteriui. Tai reiškia, kad visi duomenys yra pateikiami kaip tekstas, nepateikiant jokios informacijos apie duomenų metaduomenis, paskirtį ir pan. Tačiau kompiuterinis naudojimas reikalauja suprasti duomenų kontekstą. Todėl, kol internetinių tinklalapių kūrėjai ir savininkai nepereis prie naujos kartos interneto technologijų ar duomenų pateikimo dviem formatais (vartotojams HTML formatu, o kompiuteriams – kaip API), būtina turėti sprendimus, skirtus internetinių tinklalapių turinio blokų tipui ir paskirčiai identifikuoti.

Nors teksto analizės ir generavimo technologijos pastaraisiais metais labai tobulėjo, jos vis dar nėra pakankamai tikslios automatiniam tinklalapio turinio segmentavimui ir

kiekvieno jo turinio bloko paskirties nustatymui. Todėl norint tobulinti internetinių tinklalapių duomenų kompiuterinį suvokimą, reikia plėsti esamus šios srities duomenų rinkinius ir naudoti juos kuriant naujus tinklalapių turinio nuskaitymo sprendimus.

Darbo aktualumas

Esant detaliam internetinių tinklalapių turinio blokų duomenų rinkiniui, galima atlikti tyrimus, leidžiančius išbandyti įvairius duomenų analizės klasifikavimo sprendimus, kurie leistų turinio blokų tipo identifikavimą automatizuoti, nebereikalausti žmogaus darbo ar atskiros sąsajos, skirtos kompiuteriniam duomenų nuskaitymui ir interpretavimui. Toks turinio blokų identifikavimas leistų lanksčiau nustatyti ir nuskaityti tinklalapių pagrindinį arba naują turinį, ignoruojant dizaino ar navigacijos elementus.

Taip pat aiškus tinklalapio turinio blokų identifikavimas leistų realizuoti tinklalapių kokybės nustatymo paramos sistemas, leidžiančias tinklalapio kūrimo metu identifikuoti trūkstamus blokus, įvertinti jų tinkamiausią padėtį tinklalapyje ir pan. Tai galėtų būti naudojama kaip tinklalapio turinio ir dizaino automatinio atnaujinimo ar (ir) generavimo pagrindas. Senos, technologiškai ar dizaino požiūriu pasenusios svetainės galėtų būti automatiškai atnaujintos, pritaikytos prie naujų technologijų ir dizaino standartų, dėl to būtų pagerinta tokiomis svetainėmis besinaudojančių vartotojų patirtis.

Tyrimų objektas

Internetinių tinklalapių turinio blokai, metodai jų išskyrimui, susiejimui ir dokumentavimui duomenų rinkinyje.

Darbo tikslas

Padidinti duomenų rinkinių, sudarytų iš tinklalapių turinio blokų, detalumą ir supaprastinti rankinio žymėjimo procesą, sumažinant žymėjimą reikalaujančių blokų skaičių.

Darbo uždaviniai

Darbo tikslui pasiekti darbe reikia spręsti šiuos uždavinius:

1. Atlikti naujausių mokslinių tyrimų internetinių tinklalapių duomenų gavybos tematika sisteminę analizę.
2. Sukurti interneto svetainių duomenų rinkinio struktūrą, kuri apimtų įvairias duomenų savybes, apibrėžtų skirtingo lygio turinio blokus ir būtų suderinama su įvairiais duomenų gavybos metodais.
3. Sukurti sprendimą, skirtą daliniam automatizuotam interneto svetainės turinio blokų žymėjimui, kuris automatiškai nustatytų ryšius tarp turinio blokų ir juos grupuotų, taip mažinant rankiniu būdu peržiūrimų ir žymimų blokų skaičių.
4. Įvertinti sukurto duomenų rinkinio efektyvumą nustatant ryšius, grupuojant ir sudarant sąsajas tarp turinio blokų interneto svetainėje. Taip pat įvertinti iš dalies automatizuoto duomenų žymėjimo metodo veikimą.

Tyrimų metodika

Darbe taikomi mokslinės literatūros sisteminės analizės, idėjų generavimo, duomenų analizės, duomenų surinkimo ir apdorojimo metodai. Atlikti eksperimentiniai tyrimai.

Darbo mokslinis naujumas

Darbe pasiekti tokie mokslinių požiūriu nauji rezultatai:

1. Sudaryta nauja tinklalapių duomenų gavybos taksonomija, kuri atskleidžia platesnį galimų savybių ir jų reikšmių įvairovės spektrą, nei tai daro esami sprendimai.
2. Pasiūlytas metodas ir jo parametrai leidžia tiksliau aptikti panašių internetinių tinklalapių turinio blokų tarpusavio panašumą. Bloko panašumą nustato kompleksinės bloko struktūros ir turinio analizės kombinacija, atsižvelgiant į bloko variacijas ir tėvinius blokus.
3. Sudaryta internetinių tinklalapių turinio blokų duomenų rinkinio struktūra, kuri leidžia padidinti to pačio duomenų rinkinio taikymą įvairių tipų klasifikavimo sprendimams apmokyti.
4. Sukurtas metodas, kuris leidžia identifikuoti visus tinklalapio turinio blokus ir nustatyti jų tarpusavio ryšius, taip sumažinant rankinio duomenų žymėjimo reikalaujančių blokų skaičių.

Darbo rezultatų praktinė reikšmė

Pasiūlyti metodai yra aktualūs, kai kuriami platesnės paskirties ir didesnio kiekio internetinių tinklalapių turinio blokų duomenų rinkiniai. Šie sprendimai leidžia vartotojui, atliekant rankinį duomenų žymėjimą, iš karto pateikti žymėtinus duomenų blokus. Dabartiniai metodai reikalauja ne tik nurodyti duomenų bloko tipą, bet ir nustatyti paties bloko ribas. Be to, šis sprendimas ne tik akcentuoja pagrindinių turinio blokų identifikavimą, bet ir vidinių blokų, sudarančių tuos blokus, identifikavimą.

Sumažinus duomenų žymėjimo sudėtingumą, žymėtojų darbą galima panaudoti norint nustatyti išsamesnę informaciją apie turinio blokus arba tiesiog mažinti žymėjimo darbui reikalingą laiką. Tai supaprastintų duomenų rinkinių kūrimą ir savo ruožtu padidintų pažymėtų duomenų kiekį.

Turint išsamų internetinių tinklalapių turinio blokų duomenų rinkinį, būtų galima atlikti papildomus, įvairius mokslinius tyrimus, skirtus internetinių tinklalapių turinio analizei. Be to, turint tokius sprendimus, būtų galima pereiti prie greitesnių ir efektyvesnių tinklalapių duomenų apdorojimo sprendimų.

Ginamieji teiginiai

1. Taikant pasiūlytą reliacinės duomenų bazės pagrindu veikiančią internetinių tinklalapių turinio blokų duomenų rinkinio struktūrą, tie patys duomenys gali būti adaptuojami skirtingo tipo duomenis naudojančių tinklalapių duomenų blokų analizės sprendimuose.

2. Turinio blokų panašumui nustatyti nepakanka remtis tik teksto panašumu ar HTML kodo struktūra. Kombinuotas šių sprendimų taikymas leidžia padidinti tos pačios paskirties turinio blokų identifikavimo tikslumą.
3. Automatizuotas tinklalapio turinio blokų panašumo nustatymas, tarpusavio grupavimas ir susiejimas leidžia vidutiniškai apie 80 % sumažinti duomenų žymėjimo poreikį, kartu užtikrinant, kad žymėti nebus praleidžiamas nei vienas žymėti tinkamas blokas.

Darbo rezultatų aprobavimas

Disertacijos tema yra publikuoti 2 moksliniai straipsniai recenzuojamuose žurnaluose, turinčiuose citavimo rodiklį *Clarivate Analytics Web of Science* duomenų bazėje. Tyrimų rezultatai pristatyti trijose tarptautinėse konferencijose:

- Tarptautinė konferencija: “2018 Open Conference of Electrical, Electronic and Information Sciences” (eStream), 2018, Vilnius, Lietuva.
- Tarptautinė konferencija: “6th IEEE Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE’2018)”, 2018, Vilnius, Lietuva.
- Tarptautinė konferencija: “International Conference on Science & Technology” (STRA), 2023, Praha, Čekijos Respublika.

Disertacijos struktūra

Disertaciją sudaro įvadas, keturi skyriai ir bendrosios išvados. Ji apima 161 puslapį su 38 paveikslais ir 14 lentelių. Rašant disertaciją panaudoti 103 literatūros šaltiniai.

1. Segmentavimas ir turinio blokų identifikavimas interneto duomenų gavyboje

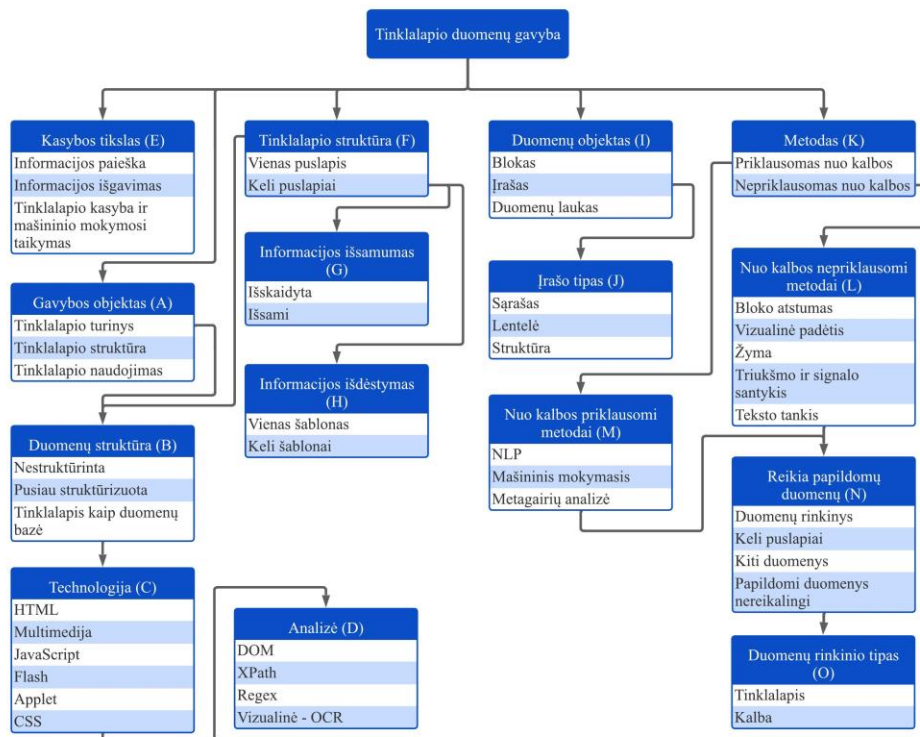
Internetinių puslapių, pasiekusių 1,1 milijardą (Siteefy, 2023), gausa atspindi informacijos amžiaus plėtrą ir iššūkius, susijusius su informacijos pertekliumi ir jos patikimumu. Lovaraju et al. (2017) pabrėžia perėjimą nuo informacijos trūkumo iki iššūkių, susijusių su jos tikslumu ir aktualumu. Aljawarneh et al. (2017) ir Besbes et al. (2017) atkreipia dėmesį į duomenų rinkimo ir analizės svarbą, siekiant atfiltruoti ar identifikuoti netinkamą turinį.

Vandenbussche et al. (2017) nagrinėjo semantinio kodo plėtrą, skirtą duomenų interpretacijai supaprastinti, tačiau internetinių puslapių duomenų gavybos sritis vis dar susiduria su iššūkiais. Nors Cooley et al. (1997) ir Kosala et al. (2000) pristatė ankstyvas interneto duomenų gavybos taksonomijas, visapusiška ir naujausius aspektus apimanti taksonomija tebėra neišvystyta.

Arotaritei et al. (2004) ir Laender et al. (2002) atskleidė naujus duomenų klasifikavimo ir rinkimo būdus, o Chang et al. (2006) pabrėžė interneto puslapių struktūros ir gavybos sudėtingumo sąsają. Naujesni tyrimai, tokie kaip Jain et al. (2017) ir Kumar et al. (2017), išryškina interneto naudojimo gavybos srities specifiką.

Nors buvo daug pastangų klasifikuoti interneto duomenų gavybos technikas ir metodus, vis dar trūksta išsamios taksonomijos, kuri apimtų visus svarbius aspektus, ypač naujausius. Tai reiškia, kad šios srities visapusiškas supratimas tebėra sudėtingas uždavinys.

Vertinant esamas taksonomijas, buvo pastebėta spraga visapusiškame interneto duomenų gavybos domeno klasifikavimo procese. Siūloma nauja interneto duomenų gavybos taksonomija, vaizduojama S1.4 paveiksle, yra paremta Kosala et al. (2000) bei kitų atliktais tyrimais, atkreipia dėmesį į turinio pateikimo technologijas, analizės metodus, internetinio puslapio struktūros įtaką gavybos strategijoms ir duomenų vaizdavimo būdus.



S1.4 pav. Pasiūlyta interneto duomenų gavybos taksonomija

Interneto duomenų gavybos automatizavimas, remiantis Dias et al. (2014) ir Win et al. (2014), yra svarbus dėl augančios automatizuotų įrankių reikšmės duomenų gavybai. Svetainių segmentavimo technikos, kurios atskiria turinio blokus, yra esminės. Naujausios tendencijos, kaip nurodo Cormier et al. (2016) ir Zeleny et al. (2017), linksta į vizualinį segmentavimą, nors jis reikalauja daugiau išteklių nei DOM metodai (Sanoja et al., 2014). Vizualiai grindžiamas segmentavimas (VIPS), aprašytas Kiesel et al. (2021), derina DOM ir vizualinį segmentavimą, tačiau DOM struktūra (Xiang et al., 2007; Alcić et al., 2011) ne visada iki galo išnaudojama.

S1.1 lentelė. Duomenų, saugomų duomenų rinkiniuose, palyginimas

Duomenų rinkiniai	Duomenys, saugomi duomenų rinkiniuose							
	HTML	DOM	CSS	JavaScript	Paveikslėliai	Ekranu nuotraukos	Švarus tek- stas	Pažymėti bloškai
Cleaneval, Utlu et al. (2018)	+	–	–	–	–	–	+	–
Dragnet, Peters et al. (2013)	+	–	–	–	–	–	+	–
Gosh, Vargas (2015)	–	+	+	–	+	–	–	–
Dataset-random, Kreuzer (2023)	+	+	+ ¹	+ ¹	+ ¹	–	+	+
Dataset-popular, Kreuzer (2023)	+	+	+ ¹	+ ¹	+ ¹	–	+	+
wt10 g, Sleiman et al. (2013)	+	–	–	–	–	–	–	–
Web-KB, Zhang et al. (2013)	+	–	–	–	–	–	+ ²	+ ²
L3S, Kohlschutter et al. (2010)	+	–	–	–	–	–	–	+ ³

¹ Iš viso 15 blokų etikečių, keturios 1-ojo lygio blokams ir 11 2-ojo lygio blokams.

² Tekstas išvalomas pašalinant nuorodas ir pažymint jas etiketėmis šaltinio kode.

³ Duomenys kategorizuojami į penkias etiketes: antraštė, išsamus tekstas, papildomas tekstas, vartotojų komentarai ir susijęs turinys.

Šiuolaikinėje tinklalapių duomenų gavyboje daug dėmesio skiriama mašininiam mokymuisi, kuris reikalauja testuoti algoritmus su tam tikrais duomenų rinkiniais. Šiuos duomenų rinkinius, apimančius įvairias sritis, pvz., naujienas ir forumus, sukūrė tokie autoriai kaip Burget et al. (2009) ir Vargas (2015), taip pat yra turizmo ir naujienų, specializuoti duomenų rinkiniai (Kuppasamy et al., 2019). Tačiau dauguma šių rinkinių yra pasenę ir dažnai nepateikia išsamių tinklalapių kopijų, reikalingų išsamesnei analizei, pabrėžiant būtinybę turėti naujesnius ir išsamesnius duomenis.

2. HTML turinio blokų panašumo nustatymas

Pusiaus struktūrinių ir nesustruktūrinių duomenų gavyba vis dar dažnai remiasi rankiniu apdorojimu dėl priemonių, galinčių efektyviai lyginti HTML blokus, trūkumo ir viešai neprieinamų duomenų rinkinių su aiškiai pažymėtais tinklalapio blokais.

Panašumo įvertinimas yra svarbus aspektas, žmonės panašumą suvokia subjektyviai, o skaičiavimo sistemos – skalėje nuo 0 iki 1. Teksto panašumui įvertinti dažnai taikomi Levenshtein atstumas (Yujian et al., 2007), Hamming atstumas (Norouzi et al., 2012) ir Jaccard koeficientas (Ni Wattanakul et al., 2013). Nepaisant šių algoritmų skirtumų, teksto palyginimas dažniausiai įvertinamas taikant redagavimo atstumą (S2.3 formulė):

$$\text{sim}(V_1, V_2) = 1 - \frac{\text{dist}(V_1, V_2)}{\text{len}(V_1) + \text{len}(V_2)}. \quad (\text{S2.3})$$

Nors anksčiau išvardinti metodai padeda įvertinti teksto panašumą, juos taikant negalima įvertinti semantikos. Semantinio panašumo nustatymas, įvertinant turinio prasminę vertę, reikalauja žodynų ar ontologijų, skirtų terminų ryšių analizei. HTML blokų panašumo įvertinimas taip pat apima semantinius ir struktūrinius aspektus, nes svarbu ne tik turinio prasmė, bet ir struktūra. XML dokumentų analizėje, kur žymės aprašo turinį,

taikomi metodai gali skirtis nuo HTML, nes HTML žymės daugiausia skirtos pateikimo formatui nurodyti, todėl ne visi metodai, tinkami XML panašumui nustatyti, tinka analizuojant HTML kodą (Algergawy et al., 2010).

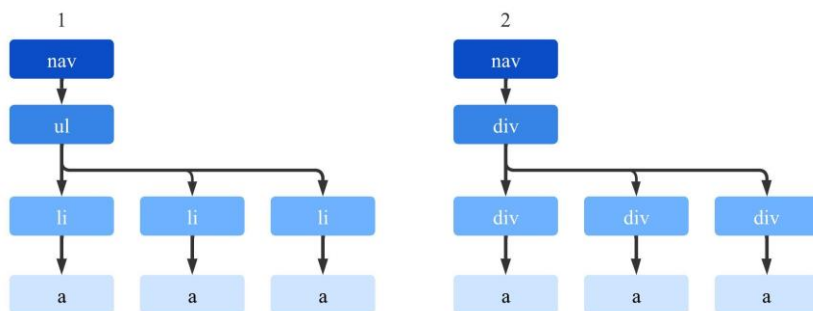
Kadangi HTML yra pritaikytas duomenims pateikti žmogui peržiūrėti, kai kurie panašumo vertinimo metodai naudoja vizualinę duomenų gavybą, kuri leidžia palyginti duomenis nenaudojant HTML kodo struktūros (Youssefi et al., 2004). Tačiau tokių metodų sudėtingumas, greitis ir tikslumas kelia iššūkių.

Pradinis siūlomas metodas HTML blokų panašumui vertinti apima tiek duomenų, tiek struktūros įvertinimą, remiantis XML panašumo vertinimo metodais. Visas procesas yra padalintas į tris etapus: duomenų panašumo vertinimas, struktūros panašumo vertinimas ir galutinis HTML bloko panašumo skaičiavimas:

$$\text{sim}(V_1, V_2) = \text{sim}_{data}(V_1, V_2) \cdot \text{sim}_{structure}(V_1, V_2). \quad (\text{S2.4})$$

Analizuojant HTML bloko panašumo vertinimo efektyvumą, pastebėta, kad standartiniai metodai yra tinkami tik paprastiems duomenims vertinti, o sudėtingų duomenų įvertinimas reikalauja vertinimo pagal kontekstą.

HTML blokams palyginti dažniausiai taikomas jų DOM medis (Buttler, 2004; Nyein, 2011; Shasha et al., 1990), tačiau šis metodas nepakankamai veiksmingas, kai struktūros skiriasi. Pavyzdžiui, blokai su identiška funkcija, bet skirtingomis struktūromis gali būti neteisingai įvertinti kaip visiškai nesutampantys. Nors kai kurie algoritmai bandė įtraukti žymų atributus į palyginimą (Yang et al., 2001), šie patobulinimai dažnai suteikia ribotą pranašumą dėl didelės atributų įvairovės.



S2.1 pav. Skirtingų žymų panaudojimo pavyzdžiai blokams, kurie gali būti stilizuoti taip, kad jie atrodytų vienodai naršyklėje

Įvairios HTML žymos gali sukurti identiškas struktūras (S2.1 paveikslas), tokie blokai gali atrodyti identišškai naršyklėse. Papildomų išvalgų galima gauti lyginant elementų klases, ypač jei svetainėje pasitelkiamas vienas iš plačiai naudojamų sąsajos su žmogumi karkasų. Tačiau klases gali lengvai sukurti ir programuotojai, todėl tikslingiau gali būti lyginti pritaikomas klasių CSS savybes, taip padidinant panašumo nustatymo tikslumą (Gowda et al., 2016).

Atsižvelgiant į tai, kad HTML kodas yra lankstus, nėra vieno panašumo palyginimo metodo, tinkančio visiems atvejams. Kompleksiški metodai tampa optimaliu sprendimu,

pavyzdžiui, Debnath et al. (2005a) pasiūlė du algoritmus – *ContentExtractor* ir *FeatureExtractor*. *FeatureExtractor* naudoja euristicą pagal tam tikras bloko savybes norint identifikuoti turinio blokus. *ContentExtractor* identifikuoja ne turinio blokus, remdamasis tuo, kad tas pats blokas randamas keliuose puslapiuose.

Siekiant įvertinti siūlomo algoritmo tinkamumą HTML blokų tarpusavio panašumui nustatyti, buvo atlikti eksperimentai su keliais algoritmo variantais. Kiekvienai operacijai buvo nustatyta operacijos kaina: viršūnės pridėjimui ir šalinimui – 0,5, modifikavimui – 0,75. Taip pat buvo įvertinamas ir gylys, kuriame atliekamos operacijos, gylio koeficientas – 0,9 vienam lygiui.

Toliau pateiktas blokų panašumo apskaičiavimas įvertinant blokų struktūrą ir naudojamą žymą (S2.5 formulė):

$$B_{sim} = \frac{\sum_1^{T_{count}} L_{score}}{T_{count}}. \quad (S2.5)$$

Lygio panašumo apskaičiavimas (S2.6 formulė):

$$L_{score} = \begin{cases} 1, & \text{žymos sutampa} \\ \frac{O_{cost} \times L}{L_{cost}}, & \text{žymos skiriasi} \end{cases}. \quad (S2.6)$$

Pakeitimo kainos apskaičiavimas lygiui (S2.7 formulė):

$$L_{cost} = \frac{L}{K^L \times O_{cost}}. \quad (S2.7)$$

Vėliau algoritmas buvo modifikuotas, atsižvelgiant į tai, kad žymos kartais gali būti keičiamos viena kita nepakeičiant rezultato (pvz., DIV gali būti pakeistas HEADER) (S2.8 formulė). Buvo įvesta fiksuota operacijos kaina 0,5, kuri buvo toliau koreguojama remiantis tikimybe, kad viena žyma galėtų pakeisti kitą. Taip žymų panašumo koeficientas gali turėti įtakos operacijos kainai.

$$O_{cost} = \begin{cases} O_c + \frac{T_{sim}}{2}, & \text{žymoms} \\ O_c + \frac{TEXT_{sim}}{2}, & \text{turiniui} \end{cases}. \quad (S2.8)$$

Kadangi panašios blokų struktūros gali turėti skirtingas paskirtis priklausomai nuo jų turinio, į vertinimą buvo įtraukta turinio analizė. Naudojantis *Cortical.io API*, galima buvo atskirti blokus su ta pačia struktūra, bet įvairiu turiniu.

Vertinant atvirojo kodo algoritmų gebėjimą įvertinti HTML struktūros panašumą, penki parinkti HTML navigacijos blokai buvo naudojami siekiant suprasti, kaip algoritmai vertino blokų panašumą, kai ta pati medžio struktūra, aprašyta naudojant skirtingas žymas. Visi blokai turėjo tuos pačius nuorodų elementus, o kiti bloko elementai skyrėsi. Papildomai dalis testų turėjo unikalų nuorodos tekstą, kad galima būtų įvertinti algoritmų jautrumą turinio pokyčiams.

Testavimo metu buvo pastebėta, kad kodo fragmentai, turintys identišką struktūrą, bet sudaryti iš visiškai skirtingų žymų, galėjo gauti 50 % panašumo rezultatą. Pasiūlytas algoritmas, kuris atsižvelgia į žymų panašumą, tokius blokus įvertino 47,61 %.

Atsižvelgiant į tai, kad egzistuoja galimybė sukurti dvi vienodas blokų struktūras naudojant skirtingas HTML žymas, šis rezultatas pabrėžia poreikį taikyti daugiaaspektę metrikų palyginimo metodiką, vertinant HTML blokų panašumą. Be to, palyginimai turėtų apimti ne tik struktūrą, žymas ar stilius, bet ir turinį dėl jo lemiamo vaidmens nustatant bloko funkciją.

Testuojant algoritmus pastebėta, kad, įtraukus gylį į siūlomo algoritmo skaičiavimus, analizuojamų blokų panašumo rodikliai šiek tiek pakilo. Žymų keitimo galimybės įtraukimas dar labiau pakoregavo rezultatus. Skirtingos žymos dažnai gali atlikti tą pačią funkciją, todėl svarbu tai atpažinti vertinant panašumą.

Atlikus turinio analizę buvo pateikti įvairūs rezultatai. Nors bendrai panašumo metrika sumažėjo 15–20 %, tam tikrose sudėtingesnėse situacijose buvo pastebėti geresni rezultatai nei tie, kurie buvo gauti be turinio panašumo vertinimo. Testų atveju minimalus turinys lėmė mažesnį turinio panašumą. Tačiau, kai turinio apimtis didėjo, panašumo įvertinimas stipriai augo, teigiamai veikdamas bendrą įvertinimą.

Įvertinus eksperimento rezultatus (S2.1 lentelė), siūlomas algoritmas parodė perspektyvius rezultatus, ypač taikant žymų panašumo įvertinimą.

S2.1 lentelė. Esamų ir naujai pasiūlyto HTML blokų panašumo palyginimo metodų rezultatai, naudojant parinktus menių blokų kodo pavyzdžius

	Vidurkis	Standartinis nuokrypis
Perl HTML::Similarity	0,65	0,18
Perl HTML::Similarity skirtingas turinys	0,65	0,18
Python HTML_Similarity	0,71	0,18
Python HTML_Similarit skirtingas turinys	0,71	0,18
Python Page_Compare	0,71	0,18
Python Page_Compare skirtingas turinys	0,71	0,18
<i>Siūlomas metodas</i>	0,79	0,13
<i>Siūlomas metodas su žymos panašumo įvertinimu</i>	0,81	0,14
<i>Siūlomas metodas su žymos panašumo įvertinimu ir skirtingu turiniu</i>	0,63	0,10

Algoritmų veikimo įvertinimas suteikė lemiamų įžvalgų, kurios reikalingos HTML turinio blokų panašumo nustatymo algoritmui tobulinti. Gylio įvertinimas, žymų pakeičiamumas ir turinys yra esminiai panašumo rodiklio nustatymo elementai. Norint optimaliai išnaudoti siūlomą algoritmą, reikalinga HTML blokų duomenų bazė, kurios sukūrimas reikalauja automatizavimo dėl didelio svetainių turinio blokų skaičiaus. Atlikus palyginimą, analizės duomenys turėtų būti saugomi, kad būtų prieinami tolesniam naudojimui. Galima teigti, kad tolesniam algoritmo naudojimui ir tobulinimui būtina sukurti svetainių informacinių blokų duomenų rinkinį.

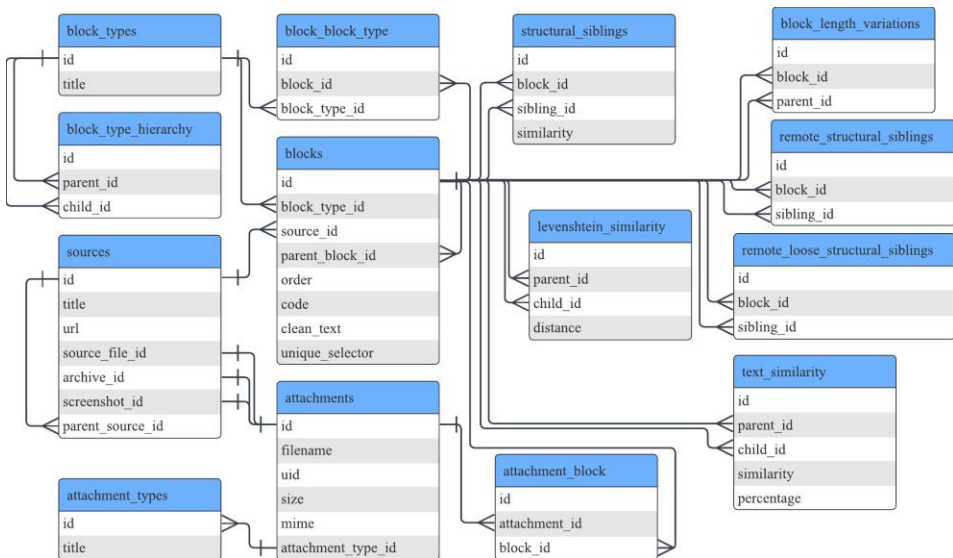
3. Daugiafunkcis tinklalapių ir jų turinio blokų duomenų rinkinys

Šiuolaikinėse svetainėse dažnai būna nereikalingas turinys, kuris apsunkina duomenų rinkimą. Siūlomas daugiafunkcis svetainių ir jų turinio blokų duomenų rinkinys gali būti naudojamas siekiant supaprastinti automatizuotą duomenų rinkimo procesą ir skatinti mašininį mokymusi paremtų metodų pritaikymą.

Egzistuoja įvairūs internetinio puslapio turinio ištraukimo metodai. Vieni iš jų analizuoja turinį, kiti svetainės struktūrą, įvertina vizualinį duomenų pateikimą arba naudoja hibridinį analizės modelį, kuris sujungia kelis analizės būdus (Ferrara et al., 2014; Laender et al., 2002). Nepaisant metodų įvairovės, nė vienas metodas nėra universalus ir tinkamas visiems atvejams (Al-Ghuribi et al., 2013), ypač atsižvelgiant į duomenų šaltinių įvairovę ir skirtingus testavimo aplinkų reikalavimus.

Didelis analizuojamų duomenų kiekis lemia tai, kad vis dažniau pasitelkiami prižiūrimo mokymo metodai internetinio puslapio turinio analizei atlikti, tam reikalingi detalai sužymėti duomenų rinkiniai (Al-Ghuribi et al., 2013; Utui et al., 2018). Siūlomas duomenų rinkinys atlieka dvigubą paskirtį – gali būti naudojamas kuriant naujus duomenų gavybos metodus arba naudojamas kaip bendras duomenų šaltinis esamų metodų tyrimams. Duomenų gavybos metodų įvairovė reikalauja universalios duomenų rinkinio, kuris būtų tinkamas įvairioms duomenų gavybos technikoms (Debnath et al., 2005b; Lin et al., 2002).

Siūlomo duomenų rinkinio architektūra yra orientuota į turinio blokus, puslapio šaltinio duomenų archyvavimą, blokų segmentus ir variacijas bei blokų ekrano nuotraukas. Tokiu būdu duomenų rinkinys gali būti naudojamas su įvairiais duomenų gavybos algoritmais.



S3.1 pav. Duomenų rinkinio duomenų struktūros diagrama

Atsižvelgiant į įvairius būdus, kaip HTML gali atvaizduoti informaciją, yra įgyvendinta galimybė saugoti blokų variacijas (S3.2 pav.). Duomenų rinkinio hierarchinė prigimtis leidžia atlikti blokų žymėjimą pagal jų poziciją HTML dokumente. Siekiant užtikrinti išsamų duomenų rinkimą, įtraukiamos kiekvieno HTML elemento ekrano nuotraukos, užfiksuotos įvairių tipų įrenginiuose (staliniuose kompiuteriuose, planšetiniuose kompiuteriuose, išmaniuosiuose telefonuose).

Įvairūs tinklalapių segmentavimo sprendimai, tokie kaip pasiūlyti Bing et al. (2014), Vargas (2015), Akpınar et al. (2013), bei kiti buvo įvertinti pagal įvesties duomenis, gaunamus rezultatus ir galimybę ištraukti šiuos duomenis iš siūlomos duomenų rinkinio struktūros. Duomenų rinkinio struktūra galėjo pateikti visus būtinus duomenis nagrinėtiems tinklalapių segmentavimo sprendimams.

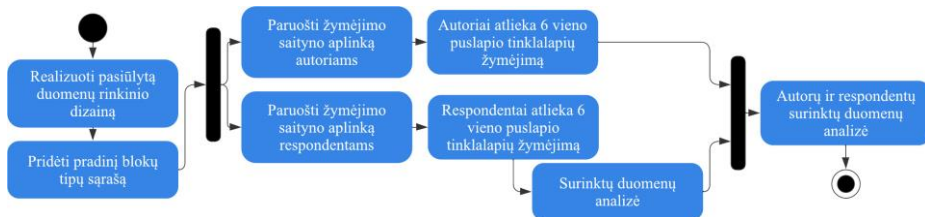
```

1 <body>
2   <a class="skippy sr-only sr-only-focusable" href="#content">...</a>
3   <header class="navbar">
4     <div class="navbar-nav-scroll">
5       <ul class="navbar-nav">
6         <li class="nav-item">...</li>
7         <li class="nav-item">...</li>
8         <li class="nav-item">...</li>
9         <li class="nav-item">...</li>
10        <li class="nav-item">...</li>
11        <li class="nav-item">...</li>
12      </ul>
13    </div>
14  </header>
15  <main class="bd-masthead" id="content" role="main">...</main>
16  <div class="masthead-followup row m-0 border border-white">...</div>

```

S3.2. pav. HTML bloko variacijų pavyzdys, kuriame raudonai apibraukti blokai, kurie gali būti laikomi tuo pačiu turinio bloku

Be duomenų apimties, svarbu suprasti, kaip duomenų rinkinyje telpa naujai atsiradę duomenys. Buvo atliktas tyrimas, siekiant išnagrinėti, kada atsiranda turinio blokų variacijų, ir įvertinti siūlomos struktūros gebėjimą užfiksuoti šias subtilybes. Metodologija apėmė rankinį segmentavimą ir žymėjimą, kad būtų užfiksuoti realaus pasaulio tinklalapių segmentavimo metu kylantys sunkumai (S3.3 paveikslas).



S3.3 pav. Pagrindinė duomenų rinkinio struktūros validavimo schema

Siekiant išlaikyti objektyvumą šeši respondentai, nuo pradedančiųjų iki patyrusių programuotojų, nepriklausomai atliko turinio blokų žymėjimą tinklalapių rinkiniui. Tai leido ne tik validuoti siūlomą struktūrą, bet ir palengvino iš anksto parinktų žymų, kurias sudarė tyrimo autoriai, palyginimą su respondentų sukurtomis žymomis. Buvo pasirinkti tinklalapiai, atspindintys įvairias sritis, kad būtų užtikrinti įvairūs blokų tipai ir dizaino variacijos. Dalyviai gavo užduotį pažymėti blokus pagal savo interpretacijas, be iš anksto nustatytų žymų, taip skatinant organišką ir įvairiapusišką žymėjimą. Tyrimo pabaigoje buvo atliktas palyginimas tarp dalyvių sukurtų žymų ir autorių pradinį žymų, toliau tobulinant siūlomo duomenų rinkinio žymėjimo metodologiją.

Eksperimento metu buvo pažymėti 905 unikalūs blokai, kiekvienas puslapis vidutiniškai turėjo 82 unikalius pažymėtus blokus. Žymėjimo procese sinoniminiai blokai buvo skaičiuojami kaip vienas blokas, pavyzdžiui, meniu punktai navigacijos bloke, kaip parodyta S3.4 paveiksle.



S3.4 pav. Besidubliuojančių blokų ignoravimo pavyzdys, kai bloko numeris 4 atstovauja bet kuriam meniu punktui, visų meniu punktų žymėjimas nėra būtinas

Vienas iš svarbių pastebėjimų buvo įvairi terminologija, kurią dalyviai vartojo identifiškiems blokams žymėti. Pavyzdžiui, blokas, kurio numeris 1, buvo vadinamas 'navbar', 'header' arba 'top navbar'. Duomenų rinkinio struktūros lankstumas buvo akivaizdus, nes ji galėjo apimti kelias sinonimines žymes vienam blokui (S3.4. paveikslas).

Eksperimento, kuriame buvo segmentuojami 6 skirtingų tipų tinklalapiai, rezultatai atskleidė didelį subjektyvumo laipsnį žymint blokus. Beveik 50 % unikalių blokų gavo bent dvi skirtingas etiketes, ir ši proporcija didėjo kartu su pažymėtų blokų skaičiumi. Tai pabrėžia duomenų rinkinio gebėjimą saugoti kelias blokų variacijas kaip svarbų veiksnių.

Dalyvių naudotos blokų etiketės daugiausia sutapo su autorių pradinio žymėjimu, sutapimo rodiklis buvo 75,41 %. Tačiau kai kurios etiketės buvo per daug orientuotos į bloko duomenis, tai prieštarauja abstrakčiai prigimčiai, kurią blokų etiketės turėtų turėti.

Eksperimento rezultatai patvirtino pradinės duomenų rinkinio struktūros ir žymėjimo schemas tinkamumą. Nepaisant savo stiprybių, siūloma duomenų rinkinio struktūra turi ir trūkumų. Rankinis žymėjimas, ypač blokams, turintiems daug panašių elementų, parodė, kad respondentai ne visada pasirenka tuos pačius blokus žymėti. Tokių, panašių pažymėtų blokų sujungimas reikalavo papildomo rankinio darbo, kuris užėmė gana nemažai laiko. Tai išryškina poreikį papildyti žymėjimo procesą įrankiais ar metodais, kurie supaprastintų žymėjimo procesą, sumažintų žymėti pateikiamų blokų skaičių ir gebėtų automatiškai nustatyti, kai žymimi pasikartojantys blokai.

4. Tinklalapio turinio blokų identifikavimas su išplėstinėmis bloko savybėmis

Tinklalapių turinio, dažniausiai pateikiamo per HTML, efektyvus žymėjimas yra būtinas turiniui išgauti, duomenims kaupti ir transformuoti, nepaisant iššūkių, susijusių su semantinių ir bendros paskirties žymų egzistavimu (Xie et al., 2022; Anami et al., 2014).

Automatizuotas duomenų išgavimas iš tinklalapių, remiantis HTML kodu, susiduria su iššūkiais, susijusiais su blokų hierarchinės prigimties atpažinimu, jų segmentavimu ir kategorizavimu (Cheng et al., 2019; Hashemi, 2020; Abbasi et al., 2022). Trūksta sprendimų, leidžiančių autonomiškai išgauti ir teisingai kategorizuoti duomenis iš nežinomų tinklalapio struktūrų, todėl siūlomas metodas, kurį taikant siekiama pagerinti tinklalapio segmentavimo sprendimus, apimant platesnį turinio blokų spektrą ir suteikiant įžvalgų apie jų vidinę struktūrą bei hierarchinius santykius.

HTML kodas yra hierarchiškai struktūrintas, kur didesnėse žymose yra mažesni elementai, atstovaujantys atskiroms turinio dalims. Pavyzdžiui, meniu etiketė gali apimti meniu punktus kaip giminingus elementus. Vienas vizualinis elementas, pavyzdžiui, meniu, gali būti sukurtas naudojant įvairias žymų kombinacijas, tokias kaip <header>, <div> ar <nav>. Bloko „maksimali riba“ yra daugiausiai apimanti žyma, o „minimali riba“ atstovauja mažiausiai apimančiai žymai, tarp šių ribų nėra jokio turinio (vaizduojama S3.2 paveiksle).

„Blokų variacijos“ yra panašaus pobūdžio blokai, esantys po tuo pačiu tėviniu bloku, tačiau ne visi vienodo turinio blokai laikomi vieni kitų variacijomis. Jų klasifikavimas kaip variacijų ar giminingų blokų priklauso nuo struktūrinių panašumų. Tikslus blokų diapazono ir variacijų atskyrimas yra esminis tinklalapio analizei ir sėkmingai segmentacijai. Blokų ribų ir variacijų nustatymas padidina turinio blokų identifikavimo efektyvumą, leidžiant mažinti žymėjimo darbą. Pavyzdžiui, S4.1 paveiksle yra aštuoni blokai, giminytės ryšių naudojimas leidžia pažymėti tik du pagrindinius blokus, o likę nustatomi per jų santykį su šiais pagrindiniais blokais.

S4.1 pav. HTML kodo pavyzdys, iliustruojantis mažesnę pažymėtų blokų skaičių, kai vietoje 8 blokų (pažymėtų raudonai) reikia priskirti etiketes tik dviem blokams (pažymėti mėlynai)

Siūlomas metodas paremtas HTML medžio analize, siekiant nustatyti gretimai esančius blokus su panašiomis struktūromis, bet skirtingu turiniu. Šis metodas yra svarbus, nes jis leidžia identifiкуoti turinio blokus, kurie atlieka identišką funkciją, remiantis jų struktūra, nepaisant turinio skirtumų.

Vienas iš pagrindinių šio metodo privalumų yra gebėjimas supaprastinti duomenų žymėjimą ir segmentavimą, leidžiant pažymėti tik vieną bloką, o likusius susijusius blokus žymėti automatiškai. Siūlomas metodas yra efektyvesnis už alternatyvas (Jayashree et al., 2022; Manickam et al., 2022; Amarnadh et al., 2019), nes jis apdoroja visus turinio blokus

be apribojimų dėl jų skaičiaus, skirtingai nei kiti metodai, kurie gali riboti segmentų skaičių.

Validavimas, atliktas naudojant 10 realių tinklalapių ir 3 vieno puslapio tinklalapius, sukurtus panaudojant įvairaus turinio sekcijas, parodė, kad išsamus rankinis segmentavimas yra labai svarbus, siekiant tiksliai pažymėti turinio blokus. Buvo sukurta specializuota internetinio puslapio pagrindu veikianti žymėjimo sistema, siekiant padidinti duomenų žymėjimo tikslumą, prašant dalyvių pažymėti kiekvieną svarbų turinio bloką, bet žymėti tik vieną bloką iš kelių panašių.

Analizė atskleidė, kad tik nedidelė dalis HTML blokų buvo pažymėta (2,9 % arba 6,7 % neįtraukiant *YouTube* duomenų), tai rodo didžiulį žymėjimo užduoties mastą. Išsamus rankinis segmentavimas yra sudėtingas ir daug laiko reikalaujantis procesas.

YouTube duomenų išskyrimas iš bendros analizės yra svarbus, nes ši svetainė naudoja didelį nestandartinių žymų kiekį, kurios nėra apibrėžtos HTML standarte, ir gali reikšmingai paveikti bendrą statistiką.

Šis eksperimentas ir jo rezultatai pabrėžia siūlomo metodo svarbą siekiant palengvinti duomenų žymėjimo ir padidinti šių duomenų analizės efektyvumą.

Automatizuoto tinklalapių segmentavimo metodų taikymas leido reikšmingai, iki 80 % (71 % neįtraukiant *YouTube*), sumažinti turinio blokų skaičių. Šis rezultatas buvo pasiektas per kelis etapus: pradžioje atmetus nereikalingas žymas, blokų skaičius sumažėjo 3,5 % (8 % neįtraukiant *YouTube*), tolesnis turinio filtravimas pagrindinėje 'body' žymoje sumažino blokų skaičių 57 % (28 % neįtraukiant *YouTube*), o turinio blokų ribų ir variacijų nustatymas leido blokų skaičių sumažinti dar 51 % (58 % neįtraukiant *YouTube*). Duomenų kaupimas segmentavimo eksperimentui buvo vienodas tiek rankiniam, tiek automatizuotam metodui, naudojant tą pačią duomenų bazės struktūrą.

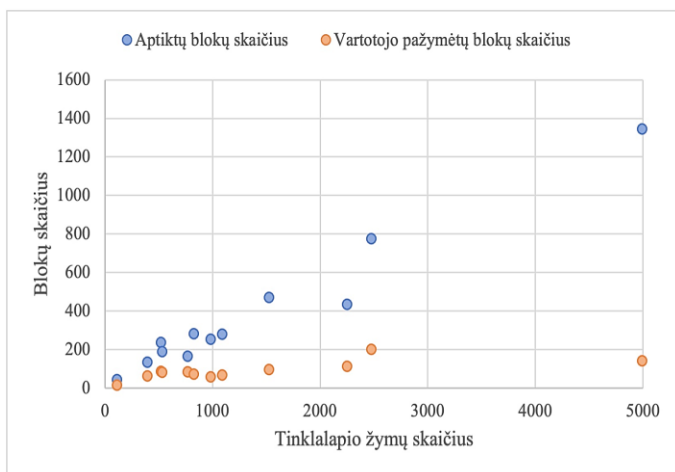
Taikant vertinimo strategiją, leidžiančią palyginti rankiniu būdu pažymėtus blokus su automatizuotu segmentavimu, buvo nustatyta, ar rankiniu būdu pažymėtas blokas patenka į automatiniu būdu nustatytas ribas, taip pat buvo įvertinti giminingi blokai su skirtingu turiniu. Tyrimo tikslas buvo įvertinti automatizuoto ir rankinio segmentavimo rezultatų sutapimą, siekiant nustatyti automatizuoto sprendimo tikslumą identifikuojant turinio blokus. S4.2 paveikslą grafike pavaizduota, kad automatiniu būdu aptinkamas blokų skaičius didėja proporcingai žymų skaičiui, o rankiniu būdu žymimų blokų skaičius tokios priklausomybės neparodė.

Metodas išsiskiria unikaliu požiūriu į turinio blokų identifikavimą, siekiant atpažinti visus galimus blokus, o ne tik ribotą jų rinkinį. Siūlomas metodas, taikomas turinio blokams aptikti internetiniuose puslapiuose, pasiekė 100 % tikslumą atpažįstant rankiniu būdu pažymėtus blokus ir identifikavo papildomus, rankiniu būdu nepažymėtus turinio blokus. Tai rodo metodo pranašumą palyginti su tradiciniais metodais, šis pranašumas leidžia atpažinti platesnį turinio blokų spektrą. Bendras tikslumas buvo apie 83 % (79 % neįtraukiant *YouTube*), o F reikšmė, rodanti tikslumo ir atpažinimo harmoninį vidurkį, buvo apie 25 % (38 % neįtraukiant *YouTube*), tai atspindi tam tikrus metodo pritaikymo iššūkius.

Lyginant su kitais puslapių segmentavimo metodais buvo pasiektas 100 % atpažinimo lygis, tikslumas yra 77 %, o F reikšmė – 19 %. Šie skaičiai rodo, kad metodas yra labai efektyvus atpažįstant turinio blokus, tačiau taip pat reikalauja tikslumo optimizavimo. Šis metodas yra svarbus sudėtingoje svetainių turinio analizėje, leidžia tiksliai

identifikuoti ir apdoroti pasikartojančius turinio blokus, atsižvelgiant į jų struktūrinius ir turinio skirtumus. Tai pabrėžia pažangių analizės metodų, kurie gali atpažinti ir klasifikuoti turinio blokus pagal sudėtingesnius kriterijus nei paviršutiniškas panašumas, svarbą.

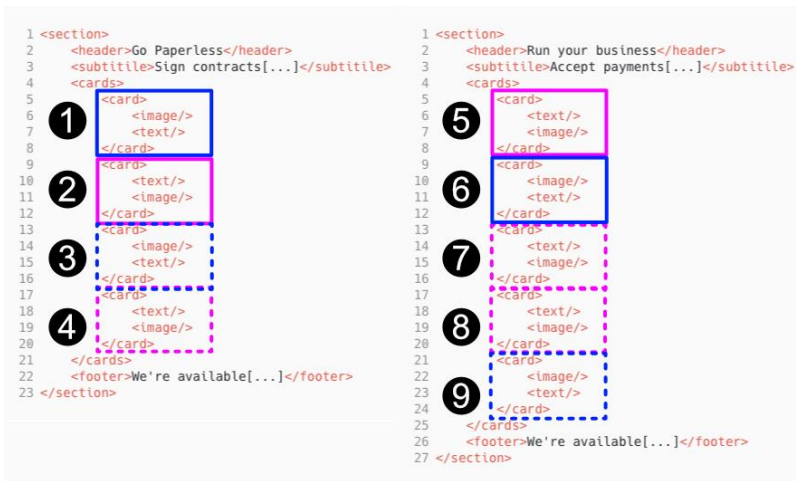
Pradinėje svetainės analizės fazėje identifikuoti tiesioginiai giminingi blokai, padalinti į keturias grupes: kairėje pirmoje grupėje yra 1 ir 3 blokai, antroje – 2 ir 4; dešinėje pirmoje grupėje yra 5, 7 ir 8 blokai, antroje – 6 ir 9. Dėl šio identifikavimo reikalingų rankiniam žymėjimui blokų skaičius sumažėjo nuo devynių iki keturių. Toliau, taikant nutolusių giminingų blokų nustatymą, iš analizės atmesti 5 ir 6 blokai, paliekant tik du blokus (1 ir 2) tolesnei analizei. Šis metodas leidžia efektyviau atlikti svetainės turinio analizę, identifikuojant ir eliminuojant pasikartojančius turinio blokus.



S4.2 pav. Priklausomybės tarp žmogaus sužymėtų blokų, aptiktų blokų ir žymių kiekio tinklalapyje

Pagrindinis metodologijos iššūkis yra tikslus panašaus, bet pasiskirsčiusio turinio identifikavimas ir susiejimas, kuris pasiekiamas taikant dvi strategijas: tiesioginių ir nutolusių giminingų blokų nustatymą. Tiesioginiai giminaičiai yra dubliuojantys blokai su bendra tėvine struktūra, o nutolę giminaičiai – tokio paties tipo blokai su skirtingais tėviniais blokais. S4.3 paveiksle matyti, kaip nutolusių giminingų blokų taikymas padeda atskirti turinio blokus su svarbiais struktūriniais skirtumais, net jei jie paviršutiniškai atrodo panašūs.

Nutolusių giminingų blokų nustatymas efektyviai mažina analizuotinų ar žymėtinų blokų skaičių, sumažinant rankinio darbo ir skaičiavimo išteklių poreikį, tai ypač naudinga didelėms svetainėms. Šis procesas prasideda nuo blokų struktūrų analizės ir HTML medžio konvertavimo į unikalią medžio struktūrą, paremtą *Newick* formatu. Procesas susideda iš dviejų etapų: medžio konstrukcijos ir optimizacijos (S4.1 lentelė), leidžiančių efektyviau analizuoti svetainės puslapius.

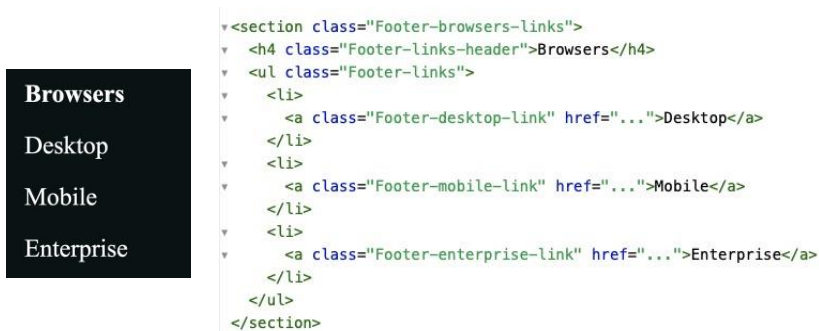


S4.3 pav. Turinio blokų sumažinimas, aptinkant gretimus giminingus blokus

Medžio kūrimo fazės metu išryškėja dvi medžio variacijos, kurios nustato dvi nutolusių giminingų blokų kategorijas. Negriežto medžio naudojimas orientuotas į HTML struktūrą ir žymų pavadinimus, o griežto medžio naudojimas įtraukia klasių pavadinimus detaliam palyginimui. Ši dvipusė strategija užtikrina išsamią analizę, tinkamą įvairiam internetiniam turiniui.

Nutolusių giminingų blokų identifikavimo procesas reikalauja palyginimo tarp sukuriamų medžių. Tai atliekama du kartus abiem medžio tipams, negriežtiems ir griežtiems, taikant gešalto šablonų atitikimo algoritmą kaip palyginimo metodą.

Individuali medžio notacija, sukurta lankstumui, leidžia lengvai įtraukti papildomus duomenis, kad būtų patobulintas blokų palyginimas. Šis išplečiamumas yra svarbus atskiriant struktūriškai panašius, bet skirtingos paskirties blokus, nes vizualūs skirtumai dažnai reiškia skirtingus atributus. Papildomų duomenų integravimas yra ypač reikšmingas analizuojant trumpus medžius, kurie gali turėti identiškas struktūras kitose dokumento vietose.



S4.4 pav. Bloko pavyzdys, naudojamas medžiui generuoti

S4.1 lentelė. Negriežto medžio generavimo iš HTML bloko pavyzdys

Initial tree	(section(h4(),ul(li(a()),li(a()),li(a()))))
Optimised tree	(section(h4,ul(li(a),li(a),li(a))))

Nutolusių giminingų blokų nustatymo procesas buvo išbandytas su dviem duomenų rinkiniais: atviru šaltiniu ('dataset-popular') ir savo sukurtu. Sukurtas duomenų rinkinys, surinktas per 2022–2023 m., yra detalesnis, su vidutiniškai 91 pažymėtu bloku kiekvienoje svetainėje, palyginti su viešai prieinamo duomenų rinkinio, kuris remiasi 2013 m. duomenimis ir turi vidutiniškai 16 pažymėtų blokų kiekvienoje svetainėje, duomenimis.

Optimizavimo procesas, skirtas turinio blokų analizei, susideda iš kelių fazių: pradžioje atskiriami potencialūs turinio blokai, o vėliau filtruojamos turinio žymos, pašalinant nereikšmingas ir paliekant tik svarbų turinį. Šis procesas vidutiniškai sumažina turinio blokų skaičių 84,23 %, išryškinant jo efektyvumą. Panašūs rezultatai gauti taikant šią logiką 'dataset-popular' rinkiniui, kuris apima 50 svetainių, parodytą vidutinį turinio blokų sumažėjimą 81,53 %.

Tyrimas buvo orientuotas į turinio blokų žymėjimo tobulinimą, įtraukiant nutolusių giminingų blokų aptikimo ir optimizavimo technikas, itin sumažinant rankiniu būdu žymėtinų blokų skaičių. Nutolusių giminingų blokų nustatymas ir pašalinimas leido sumažinti blokų skaičių 42,75 % (arba 46,64 % neįtraukiant *YouTube*), tai yra reikšmingas pagerėjimas lyginant su ankstesniais metodais.

Tačiau buvo nustatyti metodo apribojimai, pavyzdžiui, negebėjimas identifikuoti giminingus blokus su skirtingu pasikartojančių vidinių blokų skaičiumi ir ribinės panašumo rodiklio reikšmės apribojimai. Norint išspręsti šiuos apribojimus, reikia įdiegti papildomus blokų aptikimo kriterijus ir nustatyti aukštesnę ribinę rodiklio reikšmę. Tyrimas parodė nutolusių giminingų blokų aptikimo potencialą gerinant internetinių puslapių žymėjimo efektyvumą, atveriant galimybes tolesniam šių metodų tobulinimui.

Bendrosios išvados

1. Internetinių duomenų gavybos taksonomijų analizė atskleidė, kad esamos taksonomijos yra orientuotos į tam tikras specifines sritis, todėl nė viena iš jų nepateikia visapusiškos duomenų gavybos srities apžvalgos. Tai neleidžia visavertiškai suprasti atskirų internetinių duomenų gavybos sričių tarpusavio ryšių. Naujai pasiūlytoje internetinių duomenų gavybos taksonomijoje yra penkios pagrindinės sritys (gavybos tikslas, gavybos objektas, tinklaračio struktūra, duomenų objektas, metodas), kurios turi tiek vidinę, tiek bendrą struktūrą. Platesnis sričių diapazonas ir naujų komponentų pridėjimas į siūlomą taksonomiją padeda geriau išryškinti internetinių duomenų gavybos sritį.
2. Esamų metodų analizė ir pirminiai eksperimentai, atlikti vertinant svetainės turinio blokų panašumą, atskleidė, kad HTML blokų panašumo vertinimas gali būti tiksliau atliktas, jei palyginimo metu įvertinami keli panašumo veiksniai. Sukurta HTML blokų panašumo nustatymo metodika yra vidutiniškai apie 12 % tikslesnė vertinant blokus, kurie turi vienodą turinį, bet aprašyti skirtingu HTML kodu. Remiantis *Wilcoxon Signed-Rank* testu,

panašumo balų skirtumas yra reikšmingas lyginant abu siūlomus metodus su egzistuojančiais sprendimais, siūlomų metodų W reikšmės yra 1 ir 0, o kritinė W reikšmė yra 8 arba 3, kai reikšmingumo lygis yra atitinkamai 0,05 arba 0.01. Be to, lyginant skirtingas blokų poras, siūlomas metodas yra stabilus (standartinis nuokrypis sumažintas 22 %).

3. Naujai sukurta internetinių puslapių duomenų rinkinio struktūra leidžia saugoti visus esamų duomenų rinkinių duomenis ir pridėti papildomų duomenų ir įžvalgų saugojimo funkciją, orientuotą į didesnę to paties duomenų rinkinio pritaikomumą. Atlikti duomenų gavybos ir duomenų rinkinio įrašų įterpimo eksperimentai atskleidė, kad struktūra tinkama norint apimti esamus duomenų rinkinių duomenis, be to, leidžia geriau suprasti svetainės hierarchinę struktūrą.
4. Pasiūlytas ir vėliau patobulintas metodas skirtas svetainės turinio blokams aptikti, jiems pateikti hierarchinę struktūrą, su agreguotais blokų ilgių diapazonais. Atlikti eksperimentai atskleidė, kad žmonės gali rankiniu būdu pažymėti tik 3–7% žymių, esančių internetiniame puslapyje. Metodas leidžia sumažinti blokų, kuriems reikalingas žymėjimas, skaičių vidutiniškai 83 %, tuo pat metu palikdamas visus 100 % žymėjimui reikalingų blokų. Tai rodo, kad siūlomi metodai tinka kaip įrankis svetainės turinio blokų žymėjimo pastangoms sumažinti, neprarandant žymėjimui reikalingo tikslumo. Taip pat suteikiamas pagrindas žymėjimo įrankiams kurti.

Kiril GRIAZEV

INTERNETO TINKLALAPIŲ TURINIO BLOKŲ
DUOMENŲ RINKINYS IR JO DUOMENŲ ŽYMĖJIMĄ
SUPAPRASTINANTYS SPRENDIMAI

Daktaro disertacija

Technologijos mokslai,
Informatikos inžinerija (T 007)

INTERNET WEB PAGE CONTENT BLOCK
DATASET AND SOLUTIONS FOR ITS DATA
LABELLING SIMPLIFICATION

Doctoral Dissertation

Technological Sciences,
Informatics Engineering (T 007)

Lietuvių kalbos redaktorė Dalia Markevičiūtė
Anglų kalbos redaktorė Jūratė Griškėnaitė

2024 05 10. 12,7 sp. I. Tiražas 20 egz.
Leidinio el. versija <https://doi.org/10.20334/2024-032-M>
Vilniaus Gedimino technikos universitetas
Saulėtekio al. 11, 10223 Vilnius
Spausdino UAB „Ciklonas“,
Žirmūnų g. 68, 09124 Vilnius