Faculty of Technology and Society  
Department of Computer Science and Media Technology

Master Thesis Project 30p, Spring 2016

adxtractor – Automated and Adaptive Generation of Wrappers for Information Retrieval

Muhamet Ademi  
muhame@live.se

Supervisor:  
Daniel Spikol

Examiner:  
Enrico Johansson

March 17, 2017
adXtractor – Automated and Adaptive Generation of Wrappers for Information Retrieval
by Muhamet Ademi

The aim of this project is to investigate the feasibility of retrieving unstructured automotive listings from structured web pages on the Internet. The research has two major purposes: (1) to investigate whether it is feasible to pair information extraction algorithms and compute wrappers (2) demonstrate the results of pairing these techniques and evaluate the measurements. We merge two training sets available on the web to construct reference sets which is the basis for the information extraction. The wrappers are computed by using information extraction techniques to identify data properties with a variety of techniques such as fuzzy string matching, regular expressions and document tree analysis. The results demonstrate that it is possible to pair these techniques successfully and retrieve the majority of the listings. Additionally, the findings also suggest that many platforms utilise lazy loading to populate image resources which the algorithm is unable to capture. In conclusion, the study demonstrated that it is possible to use information extraction to compute wrappers dynamically by identifying data properties. Furthermore, the study demonstrates the ability to open non-queryable domain data through a unified service.
The study conducted found that computing extraction rules for specific web applications and paired with information extraction algorithms and content analysis makes it possible to retrieve unstructured information through structured documents.

Currently, the Web contains large amounts of non-queryable information as there lacks wide-use of standardized technologies to open data to the masses. Although, there has been a number of technologies developed to aid in structuring data published on the Internet such as the Semantic Web, it is still not widely used thus preventing us to collect this information. Our project presents a solution by aiming towards generating specific extraction rules tailored for the content presentation of specific web pages to make it possible to query this information through the means of information retrieval and processing. The application area for this type of study is vast, but primarily this would allow the system to fetch information within one domain, for instance automotive listings, from various resources and make the data available through a single unified service. Furthermore, the technology could also be used to construct accessible APIs to make other web applications access this data through the Internet.

The results from the findings suggest that the implementation of the information extraction algorithms paired with our wrapper generation method is successfully able to capture the majority of the data properties associated with one domain. The measurements derived from the results were conducted through accessible, live web applications that publish automotive listings. However, the measurements also demonstrate that the approach of using wrappers to retrieve information is unreliable at acquiring all data properties. Heavier resources, such as images may be loaded only when required by code that is executed dynamically on the client-side. This poses a number of challenges as the wrapper generation is not rendering the web pages, but only parsing the HTML documents so any lazy-loaded attributes may therefore be inaccessible for the information retrieval process.

The findings from the project support our claim to generate functional wrappers by using domain knowledge to generate wrappers. The results also suggest that the novel approach of identifying similar listings by using tag frequency, and pixel density and comparing their similarity is successful, although due to the relatively small sample size further research must be conducted to prove the extensibility of the algorithm.
Acknowledgements

To my family. Thank you!

I would like to express my deepest gratitude to my supervisor, Daniel Spikol for his words of encouragement, continued support and guidance throughout this process.

Furthermore, I would also like to extend my gratitude to my professors, friends and colleagues who have been with me throughout this journey.

December 21, 2016

Muhamet Ademi
Contents

Abstract                      iii

Popular Science Summary       v

Acknowledgements              vii

1 Introduction                1
   1.1 Project Idea and Objective ........................................... 3
   1.2 Motivation ............................................................... 3
   1.3 Research Questions ..................................................... 3
   1.4 Hypothesis and Project Goals ......................................... 4
   1.5 Outline of Thesis ....................................................... 4
   1.6 Limitations .............................................................. 5

2 Background                  7
   2.1 Extracting Information from Unstructured Text ...................... 7
   2.2 Wrapper Generation for Web Documents ................................ 9
   2.3 Semantic Web and Extensions ......................................... 13
   2.4 Research Opportunities ............................................... 13

3 Research Methodologies      15
   3.1 Research Methodology ................................................... 15
   3.2 Conducting the Literature Review .................................... 19
   3.3 Development and Evaluation of Artefact .............................. 19
      3.3.1 Research Tools and Techniques .................................. 20
      3.3.2 Construction of the Reference Sets .............................. 20
      3.3.3 Document Analysis and Content of Interest ..................... 22
      3.3.4 Construction of Wrappers ....................................... 23
      3.3.5 Artefact Evaluation Methods ................................... 26
   3.4 Suitability of other research methods ............................... 27

4 Artefact and Algorithms     29
   4.1 Algorithm Design and Workflow ....................................... 29
   4.2 Initial Document Analysis Algorithm .................................. 29
      4.2.1 Identifying Content of Interest ................................ 30
      4.2.2 Computing Node Similarity ...................................... 32
   4.3 Generation of Information Retrieval Wrappers ...................... 34
      4.3.1 Construction of the Reference Sets .............................. 34
      4.3.2 Utilising Text Extraction Techniques ............................ 35
      4.3.3 Traversing and Classifying the Listings ....................... 37
   4.4 Information Retrieval using Wrappers ................................ 41
      4.4.1 Implementation of Agent Architecture ........................... 41
      4.4.2 Extraction of Listings using Software Agents .................. 42
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Sample listing matched with records in the reference set.</td>
<td>7</td>
</tr>
<tr>
<td>2.2</td>
<td>Sample DOM tree for the wrapper generation.</td>
<td>9</td>
</tr>
<tr>
<td>2.3</td>
<td>Example of document-matching for wrapper generation as shown in [10].</td>
<td>10</td>
</tr>
<tr>
<td>3.1</td>
<td>Iterative development cycle adapted from Hevner [14].</td>
<td>15</td>
</tr>
<tr>
<td>3.2</td>
<td>Artefact design and development process.</td>
<td>19</td>
</tr>
<tr>
<td>3.3</td>
<td>Two sample data sets retrieved from independent reference sets.</td>
<td>21</td>
</tr>
<tr>
<td>3.4</td>
<td>Merged data sets generate the unified reference set.</td>
<td>21</td>
</tr>
<tr>
<td>3.5</td>
<td>Web page retrieved from eBay Motors that displays a set of listings.</td>
<td>22</td>
</tr>
<tr>
<td>3.6</td>
<td>Automotive listings retrieved from two independent sites.</td>
<td>23</td>
</tr>
<tr>
<td>3.7</td>
<td>String similarity scoring using various algorithms.</td>
<td>24</td>
</tr>
<tr>
<td>3.8</td>
<td>Sample HTML tree visualization with classes.</td>
<td>25</td>
</tr>
<tr>
<td>4.1</td>
<td>A state diagram that visualizes the flow of the artefact and the various sub-tasks associated with the IR process.</td>
<td>29</td>
</tr>
<tr>
<td>4.2</td>
<td>A state diagram that visualizes the tasks associated with the initial document analysis algorithm.</td>
<td>30</td>
</tr>
<tr>
<td>4.3</td>
<td>A small tree that is used to visualize the algorithm for the primary stage of detecting the content of interest.</td>
<td>30</td>
</tr>
<tr>
<td>4.4</td>
<td>A small tree that is used to visualize the recursive node similarity algorithm.</td>
<td>33</td>
</tr>
<tr>
<td>4.5</td>
<td>Graph representation of the reference set.</td>
<td>35</td>
</tr>
<tr>
<td>4.6</td>
<td>Text extraction mechanism.</td>
<td>35</td>
</tr>
<tr>
<td>4.7</td>
<td>The figure visualizes in which sequence the nodes were visited by utilizing the depth-first search algorithm.</td>
<td>37</td>
</tr>
<tr>
<td>4.8</td>
<td>Tree visualization with sample HTML nodes.</td>
<td>38</td>
</tr>
<tr>
<td>4.9</td>
<td>The tree on the left is the representation of the HTML elements, while the right tree is represented using the respective classes.</td>
<td>38</td>
</tr>
<tr>
<td>4.10</td>
<td>The tree is represented by combining the HTML tag and the respective classes.</td>
<td>39</td>
</tr>
<tr>
<td>4.11</td>
<td>High-level architecture for the implementation of the agent architecture.</td>
<td>41</td>
</tr>
<tr>
<td>5.1</td>
<td>Computed similarity scoring for vehicle makes with case insensitivity.</td>
<td>43</td>
</tr>
<tr>
<td>5.2</td>
<td>Computed similarity scoring for vehicle makes with case sensitivity.</td>
<td>44</td>
</tr>
<tr>
<td>5.3</td>
<td>Computed similarity scoring for vehicle models with case insensitivity.</td>
<td>45</td>
</tr>
</tbody>
</table>
5.4 Computed similarity scoring for vehicle models with case insensitivity ........................................ 45
5.5 Measurements of test 1 classes and their occupation on the page .............................................. 46
5.6 Measurements of test 1 classes and their computed similarity score ............................................. 47
5.7 Document analysis algorithm executed on test site 1 ................................................................. 47
5.8 Document analysis pre-and post execution of the filtering capabilities measured in pixels and elements for test 1 .......................................................... 48
5.9 Generated DOM tree from the selection generated by the document analysis ............................ 48
5.10 Measurements of test 2 classes and their occupation on the page .............................................. 49
5.11 Measurements of test 2 classes and their computed similarity score ............................................. 49
5.12 Document analysis algorithm executed on test site 2 ................................................................. 50
5.13 Document analysis pre-and post execution of the filtering capabilities measured in pixels and elements for test 2 .......................................................... 50
5.14 Generated DOM tree from the selection generated by the document analysis ............................ 51
5.15 Measurements of test 3 classes and their occupation on the page .............................................. 51
5.16 Measurements of test 2 classes and their computed similarity score ............................................. 52
5.17 Document analysis algorithm executed on test site 3 ................................................................. 52
5.18 Document analysis pre-and post execution of the filtering capabilities measured in pixels and elements for test 3 .......................................................... 53
5.19 Generated DOM tree from the selection generated by the document analysis ............................ 53
5.20 Measurements of test 4 classes and their occupation on the page .............................................. 54
5.21 Measurements of test 4 classes and their computed similarity score ............................................. 54
5.22 Document analysis algorithm executed on test site 4 ................................................................. 55
5.23 Document analysis pre-and post execution of the filtering capabilities measured in pixels and elements for test 4 .......................................................... 55
5.24 Generated DOM tree from the selection generated by the document analysis ............................ 56
5.25 Wrapper generation results from the generated extraction rules for test 1 ................................. 57
5.26 Evaluation results for the identification of the data fields for the listings in test 1 .......................... 58
5.27 Wrapper generation results from the generated extraction rules for test 2 ................................. 59
5.28 Evaluation results for the identification of the data fields for the listings in test 2 .......................... 60
5.29 Wrapper generation results from the generated extraction rules for test 3 ................................. 61
5.30 Evaluation results for the identification of the data fields for the listings in test 3 .......................... 62
5.31 Wrapper generation results from the generated extraction rules for test 4 ........................................... 63
5.32 Evaluation results for the identification of the data fields for the listings in test 4 ........................................... 63
5.33 Document object model tree illustrating the tree structure for the majority of the listings in the second test. .................. 66
6.1 The work that was conducted during this research study ........ 70
6.2 Wrapper generation using document-based matching/mismatching ......................................................... 71
6.3 Document pruning capabilities of the document analysis .......... 71
A.1 Visualized measurements of the document analysis .............. 79
A.2 Visualized measurements of the wrapper generation process 79
List of Tables

3.1 Adapted from Hevner et al [3] design-science guidelines . . 16
3.2 Adapted from Hevner et al [3] design-science evaluation methods . . . . . . . . . . . . . . . . . . . . . . . . . . . 17
3.3 Adapted from Hevner et al [3] design-science evaluation methods . . . . . . . . . . . . . . . . . . . . . . . . . . . 18
3.4 Selector methods examples to retrieve content . . . . . . . . 25
3.5 Artefact evaluation table . . . . . . . . . . . . . . . . . . . . . 26
4.1 Tables contain the structure of the reference sets and the details entailed with the make, models and associated years. . 34
4.2 Post-classification results . . . . . . . . . . . . . . . . . . . . . 35
4.3 The right rows contain the extraction techniques to retrieve the respective content types. . . . . . . . . . . . . . . 36
4.4 The table contains the formats of the content types, and the rules defined to aid in the information extraction phase . . 37
4.5 The table contains the extraction rules for the various elements and the retrieval method. . . . . . . . . . . . . . . 40
4.6 Complete selectors from the root node to the specified node 40
5.1 Wrapper generation rules for the content fields in test 1 . . . 57
5.2 Wrapper generation rules for the content fields in test 2 . . . 59
5.3 Wrapper generation rules for the content fields in test 3 . . . 61
5.4 Wrapper generation rules for the content fields in test 3 . . . 62
5.5 Information retrieval results for the first platform using the wrapper generation rulesets. . . . . . . . . . . . . . . 64
5.6 Information retrieval results for the second platform using the wrapper generation rulesets. . . . . . . . . . . . . . . 65
5.7 Information retrieval results for the third platform using the wrapper generation rulesets. . . . . . . . . . . . . . . 66
5.8 Information retrieval results for the fourth platform using the wrapper generation rulesets. . . . . . . . . . . . . . . 67
A.1 Measurements for the execution of the document analysis . 79
A.2 Measurements for the execution of the wrapper generation process . . . . . . . . . . . . . . . . . . . . . . . . . . . 79
**List of Abbreviations**

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRF</td>
<td>Conditional Random Fields</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>DOM</td>
<td>Document Object Model</td>
</tr>
<tr>
<td>JS</td>
<td>JavaScript</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>HTML</td>
<td>HyperText Markup Language</td>
</tr>
<tr>
<td>CSS</td>
<td>Cascading StyleSheet</td>
</tr>
<tr>
<td>XPath</td>
<td>XML Path Language</td>
</tr>
<tr>
<td>WWW</td>
<td>World Wide Web</td>
</tr>
<tr>
<td>RDF</td>
<td>Resource Description Framework</td>
</tr>
<tr>
<td>JSON</td>
<td>JavaScript Object Notation</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>NLP</td>
<td>Natural Language Processing</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

The information retrieval process from unstructured text is a highly complex area where certain methods may be more suitable depending on the type of content. The issue according to Michelson and Knoblock [21] with many documents published on the Internet is that the content is often characterised as poor due to irrelevant titles, lack of proper grammar and highly unstructured in the sentence structure. Furthermore, the quality of the content varies greatly depending on the author. The goal of this study is to improve, and automate the information retrieval process from unstructured text in automotive listings published on classified ad sites. The documents associated with this particular field are often characterised as short, poor grammar and contain spelling errors. The researchers involved in the studies [21, 22] claim that this renders many of the NLP methods ineffective for the information extraction process. Furthermore, Michelson and Knoblock [21] claim that the most suitable method for acquiring automotive listings is the use of reference sets and the results presented suggest that it yields satisfactory results.

The Internet does not have a standardized mechanism for retrieving data currently that is widely-used, and supported by the majority of the Web. The Semantic Web is an extension of the Web that aims to solve this particular problem although it is not available for a large number of web sites. There are various methods for leveraging information extraction, and some are empowered by the Semantic Web which is an extension to publish well-defined information. The main issue with the Semantic Web is that many ontologies have very poor coverage of their domain, for instance in our research we looked at the automotive ontology which did not contain nodes for all vehicle manufacturers and their associated models. Provided the coverage was significantly higher, and outperforming our supervised construction of the reference sets the outcome of this study would be different. As part of this study we pair the information extraction algorithm with our wrapper generation method to leverage precise extraction rules for documents that are retrieved from the same platform. Wrappers are documents that are tailored to the pages and are used retrieve the contents automatically. There is a number of techniques available to generate wrappers, such as tree analysis and document matching. The advantages of a wrapper is that the information retrieval is faster than the traditional text extraction techniques used to classify the contents and find a match.

The goal of the study is primarily to develop an artefact that computes
wrappers specifically for web documents, by using analysis and information extraction techniques on the web documents. The work is based on previous work within two fields, and is a fusion of wrapper generation and information extraction. We propose a novel contribution in which the two techniques are paired to generate content-based wrapper generation techniques. The information extraction techniques are based primarily on seeded background data which are referred to as reference sets. The primary goal of this thesis is to construct the artifact and evaluate whether combining these techniques allow us to compute relevant domain-driven wrappers. Thus, the primary research contribution of this study is a study which combines wrapper generation with domain-driven information extraction techniques (e.g., automotive data) to compute wrappers that specifically only generate rules for documents in relation to this field. The same results can be accomplished by only relying on information extraction techniques, but at a significantly higher cost in terms of performance and computing resources. Wrappers are significantly faster as the extraction rules are generated previously, which allows the system to fetch the data from the element nodes in the document.

The primary research contribution of our thesis is the investigation of rendering web documents, and using their tag frequency and pixel density and their computed tree similarities with their siblings. This is a pre-processing task, which allows the wrapper generation algorithm to operate on a significantly smaller scope, as this algorithm returns a small portion of the original document. The work in this study was concerned with the entire lifecycle of the artefact which means that we design a system with distributed architecture for distributed information retrieval and develop a unified data scheme to align information retrieved from documents with unique origins. In comparison to previous work within this field, we examine two research areas and attempt to bridge the gap between the research fields and use techniques within the respective fields to measure and evaluate an artefact that is based on techniques from two fields. Furthermore, research primarily within wrapper generation has been leveraged by various document to document matching techniques which compute wrapper extraction rules for entire documents.

The goal of the study was to investigate ways of generating wrappers for domain-related information so that depending on the reference sets (e.g., automotive) it would only compute rules for the relevant nodes in the document to retrieve automotive data. This enhances the performance of information retrieval, since information retrieval by wrappers significantly outperform information retrieval by text extraction techniques in terms of speed. The findings for the various components and sequences of the system aim to measure and validate the work on a component-level. The system works in a sequential order and each component is entirely dependent on the previous steps. Thus, if the document analysis algorithm computes an inaccurate identifier for the wrapper generation process it will most likely be unable to compute successful wrappers. Therefore, the measurements indicate whether the various system components were successful in accomplishing their respective task and aims to visualize the findings of the nodes, and respective data properties.
1.1 Project Idea and Objective

The objective of the research study was to investigate the feasibility of pairing domain-seeded knowledge and using this to generate wrappers. We use domain knowledge, in the form of reference sets which allows us to find approximate string matches for domain-related data. The findings from this process are used primarily to identify, and generate wrapper rules for those particular element nodes. The wrapper generation is reliant on the fuzzy string matching to compute wrapper rules. Wrappers consist of rules tailored for each document specifically, that describe how to retrieve the particular values of interest. The manual generation of wrappers can be tedious, time-consuming and inflexible which renders the manual generation less effective than automated wrapper constructions. We aim to investigate whether the wrapper generation can be automated by pre-seeding background information.

In addition to the processes described above, we aim to investigate the possibilities of computing the content of interest generically by comparing tag frequency, pixel density and sibling similarity scoring. The product of the research is an instantiation which is a working system that will (i) retrieve the web documents using distributed, agent architecture, (ii) identify the contents of interest, (iii) pair information extraction techniques to generate wrappers (iv) insertion of data into a unified data scheme and (v) evaluate the contents retrieved through the wrappers. Furthermore, to evaluate the applied research we will use a variety of evaluation methods to measure accuracy, performance, and optimisation. The evaluation will be done experimentally in a controlled environment with data from accessible, live web pages and the findings from these measurements will be used to answer our research questions.

1.2 Motivation

The motivating factor for undertaking this work is to primarily investigate the feasibility of pairing information extraction techniques coupled with wrapper generation to evaluate the findings. Traditionally, wrapper generation techniques have been primarily leveraged by using document to document matching techniques to identify string and tag mismatches, thus generating extraction rules for these mismatches. Although, these methods are successful our motivation for pairing these methods is to automatically detect the domain of the documents and generate extraction rules for a set of attributes, rather than rules for the entire documents. Paired with this, we also want to investigate methods for pruning the documents, effectively reducing the document to the content of interest.

1.3 Research Questions

The research questions were designed based on the research gaps that were defined as part of the literature review. We define a primary research question, paired with two sub-research questions. The research question is:
Chapter 1. Introduction

1. How can we generate information retrieval wrappers automatically by utilizing background knowledge to retrieve listings from unstructured, automotive listings on classified ad pages published on the Internet?

The main research question raised a couple of sub-research questions:

(a) In what way can we improve the performance and optimise the information retrieval process?

(b) How to integrate the information retrieved into a unified data scheme so that it can be exposed through a single service?

1.4 Hypothesis and Project Goals

The information retrieval process is improved primarily by (i) pruning the documents and generating a sub-tree that only contains the content of interest and (ii) pair information extraction techniques to generate wrappers automatically.

We can therefore break down the project goals to a number of steps required to conduct the research study:

1. Develop an instantiation to demonstrate the working system and the sub-items associated.

   (a) Pruning techniques to identify the content of interest and generate a sub-tree which only contains relevant information.

   (b) Pair information extraction techniques to generate information retrieval wrappers automatically.

   (c) Evaluate the various parts of the research to measure accuracy, performance and optimization.

1.5 Outline of Thesis

The remainder of the thesis is organized as follows. In Chapter 2, we introduce the background of the problem and the related works within the field of text extraction and wrapper generation strategies. The chapter is aimed to describe the problem and highlight why this study was undertaken. In Chapter 3, we introduce the research methodologies and the tools and techniques utilised for the systematic research process. Furthermore, the various steps involved with the prototype development were highlighted under this section. In Chapter 4, we introduce the underlying theory behind the algorithms and the implementation of the artefact. Furthermore, the theoretical background for the architectural implementation of the software agents is described. In Chapter 5, we present the measurements retrieved from the evaluation process described in the methodology. In Chapter 6, we discuss the results of artefact and compare the findings from this study to findings from other research studies within the field. In Chapter 7, we conclude the findings and advise on future work within the domain.
1.6 Limitations

The limitations of the study are code-based optimization, optimization, and performance. Code-level optimization are outside the scope of this thesis. Optimization strictly refers to pruning algorithms, specifically developed to optimize the search scope of the algorithms. Performance measurements were conducted and are available in the appendix but due to time constraints are outside the scope of this study.
Chapter 2

Background

The chapter serves as a literature view whose purpose was to identify the current state of the art research in this field but also highlight the problems, challenges and potential improvements to the current methodologies.

2.1 Extracting Information from Unstructured Text

In the research articles [21, 22] Michelson and Knoblock utilise reference sets (seed data) to extract text from unstructured text. Both studies argue that the use of information extraction techniques paired with reference sets has been proven to improve the precision and recall. The studies vary to some extent, as the previous study primarily studies how to extract information from text by using reference sets, as opposed to the other study which is primarily aimed at constructing reference sets automatically.

The researchers in [22] Michelson and Knoblock argue that the construction of reference sets is a tedious, demanding process and that primarily it is hard to complete a reference set database with high coverage. Furthermore, they argue that often reference set databases are hard to locate on the Web. Their solution to solve this problem is to construct reference sets from the listings themselves, paired with seed data in small amounts. Consider the example where you would like to construct a reference set for computers and notebooks, the authors describe, that this could for instance be the name of computer manufacturer, e.g., Lenovo, HP.

```
MAKE MODEL YEAR

Volvo V70 in excellent condition 07
```

**Figure 2.1:** Sample listing matched with records in the reference set.

The findings from the study [22] conducted by Michelson and Knoblock argue that their implementation is more accurate, has higher coverage as it primarily consists of seed data which enables them to build accurate reference sets from the listings themselves. As previously mentioned, the paper [21] is aimed at exploring possibilities of extracting information from web documents with the use of reference sets. Thus, this paper utilises existing
reference sets retrieved from third-party sources. The information extraction methods are based on computing the similarity given the input document and the reference set record, and selecting the reference set record which has the highest similarity to the input document. Furthermore, the evaluation conducted by the Michelson and Knoblock in [10] suggest that the measurements for the seed-based approach outperform two CRF implementations, even though the CRF-based approaches are significantly more time-demanding as the training data has to be labelled. CRFs is an abbreviation for Conditional Random Fields and it is a statistical modelling method that is used within the domain of machine learning and pattern recognition. We argue that there could be improvements for the research studies, provided the use case is information retrieval from unstructured texts in structured documents published on the Web. Primarily, in our literature review we were unable to locate research studies that touch on optimization aspects of the information retrieval processes that are based on reference sets, and information extraction.

Although, the studies [21, 22] conducted by Matthew and Knoblock indicate good evaluation measurements we argue that the methods are not suitable for information retrieval at large and that the methods are not scalable. Furthermore, we argue that depending on the application area there may be documents that only contain 20% of the content that is relevant to the reference sets. When compared with other methods [6], the findings from both studies [21, 22] suggest that outperforms the supervised and un-supervised algorithms to retrieve key and value pairs in [6]. In [6] Subramanian and Nyarko utilise two domains, cars and apartments and the data sets were systematically obtained from Craigslist. In the study, they used roughly 80% of the collection of data to construct the models, and the remaining data to test the models. As the study was aimed at comparing supervised, and un-supervised algorithms both methods were evaluated. The findings from the study suggest that the supervised method was significantly better in terms of accuracy.

We argue that the application of these studies in large-scale information retrieval is unsuitable, primarily as the algorithms are demanding and the search space unrestricted. We aim to present pruning algorithms to effectively reduce the size of the document to primarily preserve the content of interest. Furthermore, our aim is to explore possibilities of combining information extraction techniques presented by related research to automatically construct automated, and adaptive wrappers for information retrieval. We argue that the generation of wrappers tailored for specific platforms on the Web is significantly faster for processing large chunks of data, and paired with reference sets will allow our work to automatically group data. Traditional, document-based matching algorithms for wrapper generation infer rules by comparing two documents from the same origin. Additionally, we argue that the research conducted in this study will combine the best of two fields to improve the information retrieval process.
2.2 Wrapper Generation for Web Documents

The research study undertaken utilises wrapper generation for the information retrieval process, paired with the information extraction with reference sets. The two techniques allow us to create adaptive, robust and automatic information retrieval wrappers by utilising information extraction techniques to identify the data properties and the wrapper generation process to generate the rules. Wrappers are essentially rules to extract vital information from web sites, and these are tailored specifically for each web document.

Consider the problem where you want to generate a set of rules to extract the text elements 4, 5, 7 that are wrapped inside HTML elements (such as `<a>`, `<b>`). To do this, we must define a logical path from the root element at the top which is assigned the value 1. To extract element 4, we must define rules from 1 to 2 to 4. The path will remain the almost the same to retrieve the text element at position 5, although the last rule will be changed to 5 instead of 4. Thus, a wrapper is essentially a logical framework for navigating through a document object model and acquiring the content of interest which has been pre-defined by a number of methods. These rules can be in the form of Regular Expressions, CSS Selectors or XPaths. Collectively, Irmak and Suel et al. in [16, 7, 8, 10, 1] explore possibilities of developing automated and semi-supervised or unsupervised wrapper generation methods. The argument for this is primarily to reduce complexity, time to generate functional wrappers as opposed to manually annotating which is a tedious, and time-demanding task. Furthermore, we argue that the key advantage of utilising automated wrapper generation methods is that the adaptive aspects of adjusting wrappers to reflect changes in the web documents can not be overstated. Consider, the problem where you generate wrappers for thousands of pages and the majority of these documents contain large changes in the document schema, with the use of automated wrapper generation reflecting the changes of the pages in the extraction rules is done with minimal effort. In contrast with other approaches, we argue that this is by far the most effective method for scalability purposes.

Furthermore, manual generating of wrappers are more prone to errors as demonstrated in [16, 7, 8], which renders these semi-supervised or unsupervised methods more relevant to the field of the study. Current state
Chapter 2. Background of the art

Research within the wrapper generation field utilise to a large extent some sort of document-matching to infer rules by comparing documents from the same origin in order to construct extraction rules for fields in the document schema. In one of the studies, Crescenzi et al. [10] look at automatic data extraction from large web sites where the main goal is to discover, and extract fields from documents. Crescenzi et al. utilise a document matching method where the primary goal is to discover fields in the document schema. The computation of the extraction rules is done by matching documents from the same origin, and comparing the similarities and differences of two or more pages. In their implementation, they match strings and use string mismatches as a field discovery, whereas tag mismatches are primarily used to discover optional fields. Although, it is far more complex to handle tag mismatches as these can originate from lists, and other nested presentational elements. They identify whether the existing node, or the parent node are iterating types to discover whether the fields are optional. Our study looks at the possibilities of generating wrappers from reference sets by combining information extraction techniques to discover fields, as opposed to matching documents from the same origin to discover fields. Furthermore, current state of the art research processes entire documents which may extract rules for elements which contain nodes that contain no relevant information. For instance, assume you use a document matching technique to compute the wrappers and there is a dynamic element which presents the time on the page. In the majority of the studies, whose wrapper generation is built upon document matching it would compute rules for this particular node, although it may serve no purpose for the information retrieval process. Therefore, we argue that our study is relevant primarily as it will fetch the information which is contained within the reference sets. Furthermore, we study the optimisation aspects of the wrapper generation process so that only the content of interest is used during the wrapper generation process.

In [19] Liu et al. develop a semi-automatic wrapper generation method that takes advantage of semi-structured data on the web. The authors describe the development of an XML-enabled wrapper construction system...
which consists of six sequential phases. The information extraction techniques which generates valid XML files that contain the logic for the retrieval of the nodes, and data properties contained within that document. The paper is based on a semi-automated way, with an individual overseeing the wrapper generation process and defining the important regions on the page which is used for the basis of generating wrappers. In comparison with the research undertaken in this study, our research is based on an automated process and requires no user feedback. Additionally, the research undertaken in our study is based on automatically identifying the important regions using a novel approach of computing the collection of nested elements by parsing the document and investigating the tag frequency, and pixel density of the elements collectively. In a related study, [30] Yang et al. use domain knowledge to construct wrappers automatically for documents on the web. Both the wrappers, and the background knowledge is represented using XML. The domain knowledge describes the entities associated with the domain itself, such as price, city and furthermore defines a set of rules for each entity. In comparison, the research conducted in our study similarly uses pre-defined knowledge but in our research we use entire domain models with associated entities and use fuzzy string matching to match tokens automatically. Furthermore, the domain knowledge that is defined in our study can be extended for multiple pages that may originate from multiple sources. The knowledge representation in [30] is tailored for specific sites and the extension of this model is not suitable for documents that are from different sources. The main difference with the work undertaken in our study is that it is possible to re-use in a multi-domain environment with the same knowledge model.

Jedi [15] is a tool that was developed primarily to allow programmers of the tool to define, specify rules for specific web pages. The implementation of this tool is significantly more basic, and is primarily built to manually define rules. In comparison, however as it allows for manual input it is also flexible, and can be used to define very specific wrapper generation rules for even data properties, and parsing of text elements. However, other studies in relation to this field indicate that manual generation of wrappers are more error prone, and time consuming. Although, since this tool is primarily built for manual entry of guidelines and extraction rules it is better served for extremely complicated, complex subtasks associated with the wrapper generation method. Automated solutions are significantly better for easier tasks, but for extremely nested, complex DOM structures this tool is significantly more flexible.

Yang et al. [28] present a shopping agent that automatically constructs wrappers for semi-structured documents on the web. It is based on inductive learning, and is a fairly simple system that processes the data on a line-basis and categorizes it accordingly. The idea of their system is to recognize the position of the data property in relation to the line in the document, and to identify the most frequent patterns common within the nested listings to derive the price information. The overview of the system is built on a set of sequential steps, where (a) the system generates a set of wrappers for the document of that origin and (b) computes a query template for the page of origin. It is closely tied to other relevant research studies in this wrapper
Chapter 2. Background

generation field, although with one large distinction, it also analyzes input fields in forms to understand, learn how to construct a query template for that particular domain. In a more recent study [29] they present a more refined version of the original study in [28]. The study presents MORPHEUS, a more scalable comparison-shopping agent whose purpose is same as the original study, but goes more in-depth about the functionality of the system and the sequential steps of the shopping agent. The documents processed by the system are broken down into "logical lines" that are essentially identified data properties within the document. In their approach, they look for the form data (the search parameters) to match the input field. We argue that this approach is prone to generating false positives, and generating a number of irrelevant matches.

TSIMMIS [17] is a toolkit designed to quickly build wrappers, but it is primarily built on manual input. The authors argue that the wrappers built during the study were generalizable upon their observations, and could thus be extended to other sources although we argue that it is unlikely to be used for various pages currently. The flow of the system is that a wrapper was built to accept queries using a rule-based language that was built as part of this research study. The data retrieved from sources are represented using their human-readable format, OEM. The entire system was built with flexibility, thus enabling end-users to query documents from multiple sources using the same query structure. Additionally, [4] is also based on manual entry and the wrappers are generated by humans. The proposed work in [4] however accomplishes integration of information by acting as a mediator and facilitating user queries and mapping them to the information retrieved. There are a number of similarities in the studies [17] [4], most notably in how they attempt to integrate information and map standardized queries to documents retrieved from various sources.
2.3 Semantic Web and Extensions

The Semantic Web was primarily developed as an extension for the web to standardize formats and enable collaboration and sharing of accessible data published on the Web. Although, this particular method is not utilised in our study we mention it briefly since we argue that it is relevant to the study as it presents alternative possibilities of conducting our study. In a study, Braines et al. [5] enable the integration of multiple ontologies which is based on their own work. The study aims to enable users to query multiple ontologies through their own tool. The authors however identified a key disadvantage which is that it cannot trace back the origin of the RDP triples, and whether the data is stated or inferred. Furthermore, in a recent survey conducted by a number of researchers [20] they analysed specifically how intelligent semantic web technologies can be great aid for presenting highly relevant, and accurate information in specific domains. The paper is a summary of the existing state of the art research in the field, but also the challenges that the future poses within the field of the Semantic Web.

We argue that as the data is presented today, the Semantic Web has still not spread widely to make it reliable for sharing data through their standardized methods. Furthermore, we argue that it will take time for many web sites to adapt and make their sites compatible with the framework. Therefore, we argue that for the moment information extraction techniques are more relevant, especially when empowered by reference sets which contain valid and standardized information. In the previous sections, we touched on various studies whose purpose is to extract information on the Web by primarily comparing the records published on the pages with reference sets. Paired, with wrapper generation and services to expose the information you capture we believe that as of today, this method is more suitable as the Semantic Web is still in its early days. Additionally, our research is primarily concerned with automotive listings and to the best of our knowledge there is no platform which supports sharing of information through ontologies.

2.4 Research Opportunities

We identified a number of research gaps after conducting a literature review and comparing the measurements conducted by a number of studies. Primarily, the problem with existing methods is that they are either intensive and demand extensive computational resources for the information retrieval process. Consider the problem, where you have a number of dynamic documents that are leveraged by various platforms and the output of these documents are standardized and within one domain. Consider using information extraction methods in studies such as [21, 22] where you have a large number of documents that change every minute. We argue that this approach is suitable for processing smaller chunks of data, and that it is no match for wrappers that are designated to be performant, scalable. Although, traditionally wrapper generation methods have been leveraged primarily by comparing two documents sourced from the same origin, and
have had no context information of the content. Therefore, we argue that there is a gap of how to achieve scalable, and performant information retrieval. Additionally, the studies that were evaluated within our literature review either neglect optimisation entirely or use manually retrieved training data. We argue that pruning documents are an essential part of research within this domain, as it empowers the algorithms to process the information that is of interest.

We argue that the main issue with research within this domain is that there is a lack of cross-disciplinary research, and particularly combining methods to leverage scalable, performant information retrieval. Furthermore, we believe that the wrapper generation methods are most suitable for retrieving information from documents accessible on the Internet. Paired with information extraction techniques we argue that the method will be able to precisely identify key attributes and generate extraction rules based on our own wrapper generation implementation. As part of the literature review we analyzed a number of documents accessible on the page generated by a number of platforms and we found that most of the nodes remain intact, although a number of listings deviate from the regular structure. This is a key finding which allows us to successfully conduct the research study using wrapper-generation paired with information extraction techniques.
Chapter 3

Research Methodologies

3.1 Research Methodology

The research methodology selected for this research study is design science as defined by Hevner et al. [3]. The primary reason for selecting this research methodology is that it allows us to construct innovative artefacts and have the measurements derived from the artefact answer our research questions. The objective of the study is to use the research methodology and develop an instantiation to demonstrate a complete, working system. Additionally, the methodology selection allows us to solve intricate problems through the development of innovative artefacts. Furthermore, the work undertaken during this study is completed iteratively where minor parts are constructed iteratively and evaluated. It also allows for us to reflect and adjust the research study during the research process.

![Figure 3.1: Iterative development cycle adapted from Hevner [14]](image)

The measurements derived from the artefact are intended to answer the research questions. The measurements are generated by evaluating parts of the system, and the sequential steps throughout the entire system. As all steps are closely tied the output of the system depends on components working as designed. The measurements will generate a mix of quantitative and qualitative data which are visualized in tabular, graphical forms and collectively this evidence will be used to answer our research question. March and Smith [18] identified two processes as part of the design science cycle, (1) build and the (2) evaluation phase. Furthermore, they identified four artefacts which are constructs, models, methods and instantiations. As previously stated, the objective of the study is to develop an instantiation to demonstrate which is a complete, working system. Instantiations comprise
of a set of constructs, models and methods. As the work in this study is done iteratively, the output of the evaluation methods will help in improving the state of the algorithms. In a paper by Hevner et al [3] the authors propose a set of guidelines to undertake design-science research within the IS domain. The Table 3.1 presents the guidelines of the systematic research process.

<table>
<thead>
<tr>
<th>Guideline</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Design as an Artifact</td>
<td>Design-science research must produce a viable artifact in the form of a construct, a model, a method or an instantiation.</td>
</tr>
<tr>
<td>2. Problem Relevance</td>
<td>The objective of design-science research is to develop technology-based solutions to important and relevant business problems.</td>
</tr>
<tr>
<td>3. Design Evaluation</td>
<td>The utility, quality, and efficacy of a design artifact must be rigorously demonstrated via well-executed evaluation methods.</td>
</tr>
<tr>
<td>4. Research Contributions</td>
<td>Effective design-science research must provide clear and verifiable contributions in the areas of the design artifact, design foundations and/or design methodologies.</td>
</tr>
<tr>
<td>5. Research Rigor</td>
<td>Design-science research relies upon the application of rigorous methods in both the construction and evaluation of the design artifact.</td>
</tr>
<tr>
<td>6. Design as a Search Process</td>
<td>The search for an effective artifact requires utilizing available means to reach desired ends while satisfying laws in the problem environment</td>
</tr>
<tr>
<td>7. Communication of Research</td>
<td>Design-science research must be presented effectively both to technology-oriented as well as management-oriented audiences.</td>
</tr>
</tbody>
</table>

Table 3.1: Adapted from Hevner et al [3] design-science guidelines

The study will follow the adapted guidelines by Hevner et al [3] as it allows us to create innovative artefacts to solve intricate problems, supports iterative development and evaluation and in general the systematic process is highly relevant for research problems of this nature. Furthermore, undertaking the research study within the proposed guidelines will justify that the work undertaken follows proper and recognized research guidelines. The evaluation methods proposed by Hevner et al. [3] for design-science work within the field of IS is demonstrated in Table 3.2. The evaluation methods that were relevant for the research study and adapted from the
guidelines presented are briefly described in Table 3.3 and are the evaluation methods that were used for this research study.

<table>
<thead>
<tr>
<th>Observational Evaluation Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Case Study</strong></td>
</tr>
</tbody>
</table>
| Study artifact in depth in business envi-
  ronment                                  |
| **Field Study**                           |
| Monitor use of artifact in multiple ob-
  jects                                    |

<table>
<thead>
<tr>
<th>Analytical Evaluation Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Static Analysis</strong></td>
</tr>
</tbody>
</table>
| Examine structure of artifact for static
  qualities (e.g., complexity)             |
| **Architecture Analysis**                 |
| Study fit of artifact into technical IS
  architecture                            |
| **Optimization**                         |
| Demonstrate inherent optimal properties
  of artifact or provide optimality bound-
  s on artifact behavior                   |
| **Dynamic Analysis**                     |
| Study artifact in use for dynamic quali-
  ties (e.g., performance)                 |

<table>
<thead>
<tr>
<th>Experimental Evaluation Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Controlled Experiment</strong></td>
</tr>
</tbody>
</table>
| Study artifact in controlled environment
  for qualities (e.g., usability)          |
| **Simulation**                           |
| Execute artifact with artificial data    |

<table>
<thead>
<tr>
<th>Testing Evaluation Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Functional Testing</strong></td>
</tr>
</tbody>
</table>
| Execute artifact interfaces to discover
  failures and identify defects            |
| **Structural Testing**                   |
| Perform coverage testing of some metric
  (e.g., execution paths) in the artifact
  implementation                          |

<table>
<thead>
<tr>
<th>Descriptive Evaluation Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Informed Argument</strong></td>
</tr>
</tbody>
</table>
| Use information from the knowledge base
  (e.g., relevant research) to build a
  convincing argument for the artifact’s
  utility                                  |
| **Scenarios**                            |
| Construct detailed scenarios around the
  artifact to demonstrate its utility       |

TABLE 3.2: Adapted from Hevner et al [3] design-science evaluation methods

The optimization and dynamic analysis methods from the analytical evaluation aim to investigate, measure and present findings from the various parts of the system. The optimization evaluation methods aim to compare the original document to the post-processing document, and measure the reduction of tags and pixel density in the newly constructed document. The second part of the analysis evaluation method is one of the primary evaluation methods which aim to investigate, measure and present the findings for the entire system on a component-based level. The sequential steps are broken down into separate tasks, and evaluated respectively. For the document analysis and the wrapper generation, we measure the execution speed which is the time of completion for the entire analysis and present the time required to process the document, and the nodes on the page. The resource usage aims to profile the resource usage for the respective task and log the CPU and memory usage over time. Furthermore, the
### Analytical Evaluation Methods

<table>
<thead>
<tr>
<th>Optimization</th>
<th>Document Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic Analysis</td>
<td>Document Analysis</td>
</tr>
<tr>
<td></td>
<td>• Measuring pruning effects</td>
</tr>
<tr>
<td></td>
<td>• Measuring execution speed (time to completion)</td>
</tr>
<tr>
<td></td>
<td>• Measuring resource usage (CPU, memory)</td>
</tr>
<tr>
<td></td>
<td>• Measuring process execution (tag frequency, pixel density)</td>
</tr>
</tbody>
</table>

**Wrapper Generation**
- Measuring execution speed (time to completion)
- Measuring resource usage (CPU, memory)
- Measuring process execution (identified data properties, text extraction)

### Experimental Evaluation Methods

| Simulation | • Measure fuzzy string measurements |
|           | • Information retrieval measurements (retrieval of listings and data properties) |

### Testing Evaluation Methods

| Functional Testing | • Input and output validation for entire system |

---

**Table 3.3**: Adapted from Hevner et al [3] design-science evaluation methods

The main part of this evaluation is the process break-down which aims to break down the process and visualize the data generated at various stages of the document analysis and wrapper generation. The data generated by the algorithms are visualized with tables and figures in the results and evaluation of the artefact. This particular step is crucial for the research study and collectively the findings from these methods will help answer parts of the research questions. The findings from the optimization evaluation for the document analysis will help to validate whether the document analysis is able to identify the collection of listings by using tag frequency and pixel density on the documents, and whether the output is correct. Furthermore, as it constructs a new document we can compare the optimization aspects to the original document in terms of reduction of nodes on the page.

The simulation of the experimental evaluation methods attempts to measure the fuzzy string matching measurements which is the primary method paired with the reference sets that allows our system to detect automotive-related data properties on documents. Thus, the findings from this evaluation method are significant and producing the correct results is necessary for the wrapper generation to correctly be able to annotate data properties on the document. Therefore, we must evaluate whether the text extraction techniques are able to correctly map automotive makes, and models with reference sets in our database. It can not be understated how vital this
part is for the entire system, as not being able to identify the data fields correctly would significantly impact the results and evaluation for the wrapper generation. The information retrieval measurements is the last step of the experimental evaluation and it aims to use the rules generated by the wrapper generation to retrieve the listings and their associated data properties. This will answer our research question and is the last step of the sequential execution, so the output from this evaluation will be used to answer our research question. As the output from this step indicates whether the information retrieved from the documents was successful or not, our research question can be answered depending on this evaluation method. Thus, the output of this test is primarily what answers our research question, and in combination with the results from previous measurements we can also answer the sub-questions. The functional testing is a iterative process which is there to evaluate whether the components work as designed, provided input is given that the correct output is produced. The measurements from this test are presented in the section of every measurement.

3.2 Conducting the Literature Review

The literature review was conducted systematically by starting with a number of key queries, and identifying relevant literature within the field of wrapper generation. We used a number of databases such as Google Scholar, ACM, IEEE to aid in the search of the relevant literature. The starting key words used for conducting the literature review were "wrapper generation", "domain-based wrapper generation", "information extraction", "wrapper generation structured documents", "web wrapper generation", "wrapper generation semi-structured documents", "information extraction reference sets", "unsupervised extraction". The results of the queries were used to primarily assemble a starting list of relevant literature, and reviewing other works that the study themselves referenced to, or were cited by. The process was iterative and started in the early phases of the research study, and continued throughout the research project.

3.3 Development and Evaluation of Artefact

The development of the artefact starts by conducting a literature review to identify the current state of the art research, examine various text extraction techniques and wrapper generation methods. The literature review serves the purpose of formulating the problem statement, and ensuring that the research carried out in this study contributes in a number of ways. Although, the design and development phase of the research is carried out in the steps highlighted below.
1. The reference set phase of the development process includes searching the web for relevant reference sets. This phase was conducted in order to identify, review and utilise the available resources to populate our own local reference set.

2. The document analysis included manual document analysis, identifying patterns and formulating a hypothesis to see whether it is possible to identify the content of interest without the use of pre-defined background knowledge.

3. The wrapper generation process was completed iteratively to refine the development process and reflect on the algorithms. The knowledge required to undertake this phase was retrieved during the literature review that was conducted.

4. The evaluation methods to test the artefact are mentioned in the sections for the various parts. The evaluation methods generate quantitative data that is used to answer our research questions and the hypothesis.

### 3.3.1 Research Tools and Techniques

The development of the artefact requires a number of tools, techniques and methodologies. The list below contains briefly the tools and techniques that were utilised for the research process.

1. **Methodologies**

   (a) Agile methodologies [2] will be utilised to conduct the systematic research process as defined by Hevner et al [3]. The agile methodologies are defined by an iterative development process and is compatible with the proposed guidelines highlighted in Table 3.2.

2. **Tools**

   (a) Jsoup - "HTML parser"
   (b) Selenium - "Web Browser Automation"
   (c) JADE - "JAVA Agent Development Framework"

### 3.3.2 Construction of the Reference Sets

In our research we found that many reference sets were either incomplete, missing various makes or were tailored for specific markets. Furthermore, in our analysis we found that there were inconsistencies in the naming schemes for various makes and models. Consider the example *E-Class* and *E200* they refer to the same model although the engine displacement is missing entirely in the first example. We merged the two training set databases to a unified reference set, leveraged by Mobile.de [23] and Edmunds [12].

The **Figure 3.3** displays an example of differences between the regional data sets retrieved from the two sources, neither is incorrect although the regional differences indicate that there is different ways to name based on
the region, alphabet etc. Although, in order to unify the two data sets we integrate them upon insertion which is done automatically by comparing the string similarity with the existing record. For instance, Citroën is not even included in the Edmunds database since it is not sold in the US market. The first reference set insertion was the Mobile.de database with the makes and models. The second insertion (Edmunds) was dependent on computing the similarity for all the makes already inside the database.

The Figure 3.4 is the merged data sets from Figure 3.3 that were aligned after computing the similarity score for the second data set. If the similarity is high, above the pre-defined threshold the data sets are merged. The advantage position of this is that, all the models associated with that particular make only link to one manufacturer. This allows us to retrieve all the models, without having to compute similarities and parse several responses containing the same make. To compute the similarity we utilise the Jaro-Winkler [27] algorithm. Once the insertion of the remaining reference set was complete, a single make record in the database would generate all the models. This is the process of aligning multiple reference sets and integrating them into a unified data scheme. The advantage of utilising multiple reference sets and aligning them is that we gain higher coverage for computing string similarities correctly and this should generally generate much better accuracy for determining a match with the records in the reference set. Furthermore, in our analysis of the Edmunds reference set is that specifically the model matching process would fail against the majority of automotive listings published in european automotive listings.

---

1Jaro-Winkler algorithm measures the similarity between two strings.
3.3.3 Document Analysis and Content of Interest

The document analysis describes the break down of the presentational structure for the classified ad sites within the domain of automotive listings. The Figure 3.5 below is retrieved from eBay Motors and the general structure of the presentational elements is very similar to other pages within the same field. For instance, compare the figures in Figure 3.6 and the similarities of the presentational elements are very similar.

![Figure 3.5: Web page retrieved from eBay Motors that displays a set of listings](image)

From the Figure 3.5 it is clear that the listings occupy the most of the screen area as it is the main content of the web page. We can assume that the content of the listings collectively will occupy the most screen area on classified listing sites. Furthermore, as the figure suggests the structure of the presentational elements is identical for the collection of listings that are published on the page. Although, the listings may be wrapped within specific containers such as \(<\text{div}>\), \(<\text{section}>\) etc. . . . The problem with the assumption that the elements will occupy the most screen size is that other elements that may be used to wrap the content of the listings may occupy more pixels on the screen. The algorithm used to identify the content of interest for our study is to evaluate the frequency of the HTML elements, and measure the pixels occupied for each HTML element. Furthermore, our algorithm has a pre-defined filter that filters out non-presentational HTML elements such as input, section elements etc. Consider the example where the listings are wrapped within a list that has a parent node \(<\text{div}>\) with the class listing, in order to retrieve the right container we must filter out the DIV tag so that the document analysis algorithm returns the correct value.

Although, the assumption can be made and the algorithm can be developed accordingly we must add a validation layer to verify whether the child nodes within the parent container which was returned by the document analysis algorithm. To compute this, we must compare the similarity of the child nodes for the container of the listings. Consider the example where the previous step returns a unique identifier that can be used to retrieve a collection of automotive listings for that document. Assume that we retrieve the first listing \(L_1\) from the collection of listings, we must check that \(L_2\) to \(L_N\) contain the same children nodes as \(L_1\). The validation layer can
utilise recursive calls to traverse all the nodes of the listing and comparing them to another node, although the condition can be satisfied if the first-level children for the $L_1$ are a match when compared to the sibling elements.

The evaluation of the document analysis is to primarily investigate whether the algorithm will adapt and return the right value for a number of automotive listing platforms. Additionally, should the algorithm return satisfactory results we can evaluate the optimisation affects of the pruning algorithm as it generates a tree which is a subset of the original document. The evaluation is finished with dynamic analysis which will measure the algorithm in terms of performance and the various tools that were utilised to parse, render the document and in-depth analysis of the quantitative figures.

### 3.3.4 Construction of Wrappers

The construction of the wrappers is based on utilising text extraction techniques to determine the listings and then build logical paths for the DOM tree that will be utilised for future requests. The automotive listings contain fields such as the make, model, year, price, image and the destination page for the detailed automotive listing page. The retrieval of the fields is highly dependent on the content type so depending on the content type we utilise various extraction methods such as computing the string similarity or utilising pre-defined regular expressions. To compute the similarity we evaluated a number of string similarity algorithms and string distance algorithms. Although, for the string distance algorithms we modified the result to return normalized results (0...1). The measurements of the string similarity algorithms were conducted by utilising a fixed string and a list of strings that were compared with the fixed string. For this measurement we utilised Mercedes-Benz as the fixed string. The figure below presents the measurements collected from this particular test.

The string similarity algorithms that were utilised for this measurement are Levenstein distance ($lv$), Jaro-Winkler ($jw$), Jaccard ($ja$) index, Cosine similarity ($co$) and Sorensen-Dice ($so$) coefficient. The string distance algorithms were normalized to generate string similarity measures instead of returning the amount of character edits required to transform the
string. In our measurements we found that the Jaro-Winkler (jw) demonstrated the highest similarity scores as demonstrated by the measurements generated in Figure 3.7. Additionally, the Jaro-Winkler was found to be faster than the majority of the algorithms when computing the string similarity. To compute the similarity we utilise the Jaro-Winkler [27] but in order to do that we must compute the Jaro distance.

\[ d_j = \frac{1}{3} \left( \frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m} \right) \]

where \( m \) is the number of matching characters and \( t \) is the number of transposed characters. We compute the \( d_w \)

\[ d_w = d_j + (lp(1 - d_j)) \]

where \( d_j \) is the original Jaro distance, \( l \) is the length of the common prefix to a maximum of 4 characters, \( p \) is standard weight as defined by the original work so \( p = 0.1 \). The matching condition was met if the string similarity algorithm returned a value that was matched to a pre-defined threshold. The retrieval of certain content types such as the image source, destination URL are rather trivial to retrieve although we have additional regular expressions that are used more to validate whether the URLs are correct and well-shaped. For the other fields, such as price for instance whose format may vary depending on the region we pre-defined general regular expressions where the currency may occur before or after the price. The price may contain dots or commas.

The matching of the make, and the associated model is based on the Jaro-Winkler \(^2\) string similarity algorithm. In our algorithm proposal we assume that the listing is structured in the sense that the make occurs prior to the model. With the assumption we can query the words in the listing for a make match. If the string similarity algorithm finds a satisfactory result, that is higher than the pre-defined threshold, we can proceed to identify the models associated with that particular make. The reasoning for this particular set of steps is to reduce the search scope of the matching process.

\(^2\)Jaro-Winkler algorithm measures the similarity between two strings.
Although, if no make satisfied our condition we must query the database of models to see whether one of the models satisfies the condition. If a model is matched and satisfies the condition, we can infer the make from the reference set. This allows us to populate the listings that contain very little information about the vehicle.

The Figure 3.8 highlights two trees that are identical and in fact it is the same tree but represented the HTML tags, and their class association. To generate wrapper rules for the given tree we can either utilise Xpaths or CSS selectors. We conducted a series of performance analysis on the various approaches and found that CSS selectors provided significantly better performance. Furthermore, although subjective we believe that the readability of the selectors and the code is much improved.

<table>
<thead>
<tr>
<th>Method</th>
<th>Selector</th>
</tr>
</thead>
<tbody>
<tr>
<td>XPath</td>
<td>//li[@class='listing']/a</td>
</tr>
<tr>
<td>CSS</td>
<td>li[class=listting] &gt; a</td>
</tr>
</tbody>
</table>

Table 3.4: Selector methods examples to retrieve content

The selector examples in Table 3.3 display the various methods to retrieve the <a> tag under the <li> node. Evidence shows that CSS selectors are primarily easier to read, and the benchmarks conducted locally made our decision easy. For the implementation of the wrapper generation rules, we will utilise CSS selectors to generate the paths to the elements. In order to compute the paths for the elements, we utilise the text extraction techniques to match the elements when traversing through the sub-trees. Consider the example, where you want to generate a selector from the <span> element in Figure 3.8. To accomplish this you may backtrack to the parent node, all the way to the root node and generate the selectors for each level in the tree. For example, assume we start from the top element we can define the first selector as "li[class=listting] followed "p[class=desc]" and finally "span[class=italic]".

The evaluation of this process emphasises particularly on dynamic analysis, simulation and functional testing. We intend to utilise dynamic analysis to study the performance aspects of the system and measure with other approaches. Furthermore, various parts of the process will be separated and
analysed as separate components to provide more in-depth analysis for the final results. The simulation of the system is to utilise third-party tools to invoke the system and have it access a set of web pages stored locally and measure the various aspects of the simulation. This process is tied with the dynamic analysis and the functional testing. The functional testing entails the process of discovering whether the automated approach to generating wrappers with the use of reference sets actually works.

3.3.5 Artefact Evaluation Methods

The evaluation of the artefact is done by utilising the evaluation methods proposed by Hevner et al. [3]. The Table 3.4 highlights the evaluation methods and the various parts of the artefact creation. The table indicates which evaluation methods are relevant to the artefact creation and how the various parts will be evaluated.

<table>
<thead>
<tr>
<th></th>
<th>Reference Sets</th>
<th>Document Analysis</th>
<th>Wrapper Generation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Observational</td>
<td>/</td>
<td></td>
<td>/</td>
</tr>
<tr>
<td>Analytical</td>
<td>/</td>
<td>Optimisation, Dynamic Analysis</td>
<td>/</td>
</tr>
<tr>
<td>Experimental</td>
<td>/</td>
<td>Simulation</td>
<td>Simulation</td>
</tr>
<tr>
<td>Observational</td>
<td>Functional</td>
<td>/</td>
<td>Functional</td>
</tr>
<tr>
<td>Descriptive</td>
<td>/</td>
<td></td>
<td>/</td>
</tr>
</tbody>
</table>

Table 3.5: Artefact evaluation table

1. The reference sets will utilise functional testing to verify that the merging of the two data sets/reference sets has been done correctly. To validate this, we must identify a number of inconsistencies and verify upon insertion that the algorithm is capable of filtering, and merging the data sets automatically.

2. The document analysis will be utilising a couple of evaluation methods paired together to help answer our research questions. Primarily, we are interested in dynamic analysis to validate or refute our hypothesis. Additionally, we want to measure the optimisation of the document analysis and the generation of the sub-trees. This will be done programmatically and highlighted with a series of images displaying the algorithm working, and the pruning results. Furthermore, the quantitative data can be used to calculate the percentage of the DOM tree that was converted. Finally, the simulation is paired with the existing evaluation methods as we will simulate various tests with data retrieved from various web pages.

3. The wrapper generation is more concerned with the functional testing of the unit to verify whether the generation of the paths from the wrapper generation process retrieve the appropriate data. To accomplish this task we must simulate a number of tests using data that was retrieved from a variety of automotive listing sites that were stored
locally for research validity. Furthermore, this stage will also include the final evaluation of the artefact which attempts to measure the system as a whole. Consider the example, where we know for a fact a page contains 50 listings, we must measure whether the algorithm and the wrapper will pick up all of these listings, and in-depth analysis for each listing to verify whether it corresponds with the original listing.

3.4 Suitability of other research methods

The motivation for selecting design science research is that it is primarily a problem-solving paradigm that seeks to develop innovative artefacts to solve complex problems. In our evaluation of research methodologies we found that this is the research method that aligns the best with our new approach to the generation of information retrieval wrappers. To motivate our reason for selecting this particular research methodology we evaluate a number of relevant research methods which could potentially answer our research questions partially.

To provide contextual information that is relevant for a research study the literature review is a method which is done in two steps. The former step is to conduct a systematic search of the available research studies and filter the results to base the study on the most relevant studies for the topic of choice. The latter is conducted iteratively during the entire research process, refined up until the final documentation parts of the research study. The goal of this part is to critically evaluate and reflect on the research findings and to continuously relate, compare and assess the research and identify gaps within the research domain.

The main argument against using this as a primary research methodology is that it aims to critically evaluate, assess and reflect upon the existing work within the body of knowledge. In order to claim novelty within the body of knowledge we must prove our hypothesis, and this we argue can not be accomplished by only covering the relevant works. The closest research methodology that aligns well with the project outcome is the experimental methodology which aims to isolate the cause and effect by manipulating the independent variable and measuring the dependent variable(effect). It is important to isolate all other factors which could affect the potential results. We argue that this is less ideal, primarily since the evaluation of the artefact cannot be reproduced using the same data as we are using pages accessible on the Internet which may change at any given time.

Furthermore, it is our attempt to demonstrate the capability of the artefact adapting to different environments which renders the experimental method less suitable than the design-science approach which attempts to develop artefacts to solve complex problems. Additionally, to ensure internal validity one must be able to reproduce the same testing environment. Furthermore, in experiments one must be able to measure something before and after a new theory is implemented and without these initial measurements we can not assume that it has caused an effect, which is the equivalence of
an uncontrolled trial. Evidence shows that it is harder to establish a cause-effect relationship from these studies. Thus, our selection for the research methodology is primarily the design science methodology which encompasses both applied development and evaluation, and the applied research must be relevant for the body of knowledge within that research discipline. The idea is to conduct a literature review to strengthen, evaluate and relate the findings of our study with relevant research. Furthermore, the design science methodology can be represented using the systematic process of the three-cycle view by Hevner et al. [14] of design-science research, where part one is to identify the requirements and define the evaluation criterias, the second part is to highlight previous research to ensure that it is an innovation, and the last part is the iterative build and evaluation process. Additionally, the motivation for selecting this research methodology is that it is primarily a problem-solving paradigm and aligns well with the research questions and goals.
Chapter 4

Artefact and Algorithms

4.1 Algorithm Design and Workflow

The implementation of the artefact is based on a multi-step algorithm that is visualized in Figure 4.1. The main tasks involved with each incremental step are highlighted in the state diagram below.

![State Diagram](image.png)

**Figure 4.1:** A state diagram that visualizes the flow of the artefact and the various sub-tasks associated with the IR process.

The steps on the upper part of the state diagram are all part of the initial document analysis algorithm which aims to 1) identify the content of interest, and 2) reduce the search scope for the traditional information retrieval methods. The lower parts of the state diagram are associated with the information extraction methods to identify the listing properties by iterating through the DOM tree and validating the values of the HTML tags. The upper steps highlighted in Figure 4.1 are detailed in the section 4.1. The remaining tasks are detailed in the section 4.3.

4.2 Initial Document Analysis Algorithm

The initial document analysis algorithm section is a breakdown of the various algorithms utilized to analyze, prune and identify the content of interest. The steps associated with this process are highlighted in Figure 4.2.
This chapter explains the sequence of steps required to generate information retrieval wrappers. The Figure 4.2 is a simplified view of the systematic process to analyse the document and identify the content of interest. In Chapter 4.2.1 we describe how we iterate through the documents and identify the listings. In Chapter 4.2.2 we describe the process of computing the tree similarity between the sibling elements.

### 4.2.1 Identifying Content of Interest

The DOM is a complex document that includes a number of nodes and these can be represented as a tree structure, or what is commonly referred to as the DOM tree. The goal of this algorithm is to investigate any document within the scope of classified ads and automatically identify the content of interest based on iterating, and evaluating the DOM tree. The Figure 4.3 highlights the results of executing our algorithm on a small data tree.

![Algorithm started](image)

**Figure 4.2:** A state diagram that visualizes the tasks associated with the initial document analysis algorithm.

The content of interest is identified by utilizing a bruteforcing method to iterate through the DOM and store the frequency of the tags. In Algorithm 1, the initial part of the algorithm is highlighted which iterates through the page elements and filters out non-presentational HTML elements such as `<select>`, `<option> etc...). The collection of elements used with the pre-calculated tag with the highest frequency is used calculate the sum of pixels occupied for the collection of elements.

The next step is to utilize the tag with the highest frequency that was generated in Algorithm 1 and retrieve the collection of tags associated with the HTML tag that was pre-calculated in the previous step. Following this
the idea is to iterate through the collection of HTML elements and retrieve their class attributes, height and width. If the element is visible, the map containing the class is incremented by the sum of the pixels occupied on the page for the given element. When the iteration is complete, the algorithm will automatically select the most likely candidate based on the class that occupies the most screen surface on the page. The algorithm for pruning the search scope is highlighted in Algorithm 2.

Although, in order to evaluate the efficiency of detecting the content of interest we propose an alternative approach to identify the content of interest. The major difference between the approach highlighted earlier is that we assume that the highest frequency tag will most likely be the placeholder for the collection of the listings. Thus, this approach aims to iterate through the collection of tags with the highest frequency and rather than computing the node it simply uses the nodes to retrieve the unique class identifiers. The computation of the node similarity is similar, in that all the
child nodes of the parent node are matched with their siblings in order to evaluate whether it is a match.

**Algorithm 3** Content of Interest Algorithm - Alternative Approach

1: elements ← DOM elements  
2: nodes ← empty map  
3: classes ← empty map  
4: simclasses ← empty map  
5: for element ∈ elements do  
6:  if whitelist contains nodeName then  
7:  nodes(node.Name) ← increment by 1  
8: end if  
9: end for  
10: for node ∈ nodes do  
11:  if node is visible then  
12:    field ← class  
13:    height ← node height  
14:    width ← node width  
15:    classes(field) ← increment by (height * width)  
16:  end if  
17: end for  
18: for class ∈ classes do  
19:  selector ← formatted DOM selector  
20:  size ← selector elements size  
21:  if TREESIMILAR(a.nextSibling, b.nextSibling) is false then  
22:    similarity ← computed similarity score  
23:    simclasses(class) ← similarity  
24:  end if  
25: end for  
26: finalClass ← highest occupation class with similarity threshold achieved  
27: ⊿ This is a simplified version of the algorithm which does not entail all the operations.

The algorithm displayed in the code block **Algorithm 3** briefly presents the main logic for discovering the content of interest. Essentially, what differs is from the previous approach is that the computation is concerned with iterating through a number of classes with the highest frequency. The reason for this is to propose an alternative approach and measure whether the highest frequency tag can be used to discover the content of interest, and whether the evaluations will generate different results based on these two algorithms.

### 4.2.2 Computing Node Similarity

At this stage, the algorithm has computed the class that is most likely to be used by the listings on the document. The remaining step is to compute the similarity between the nodes, in particular the children of the parent node. In **Figure 4.4** there is a small visualization of a DOM tree that share a common parent node (which is a single element determined by the previous algorithms). The similarity of the nodes is based on the assumption that the listings published on classified ad sites share the identical presentational
structure. Thus, the structure of the presentational elements should utilize similar subtrees.

\[
\begin{align*}
&n_0 = \text{Level 0} \\
&n_1 = \text{Level } l \\
&n_2 = \text{Level 2} \\
&\vdots \\
&n_l = \text{Level } l
\end{align*}
\]

**Figure 4.4:** A small tree that is used to visualize the recursive node similarity algorithm

In the Figure 4.4 we can see that the tree displays a number of levels that are associated with the depth of the tree. From the previous algorithms we identify the parent node which is highlighted with \(n_0\), thus we are able to select all the children elements which are under Level 1.

**Algorithm 4 Tree similarity algorithm**

1: function **TREE SIMILAR**\((a, b)\) \(\triangleright\) First child element of trees \(a\) and \(b\)
2: \hspace{1em} if \(a\) equals \(b\) then
3: \hspace{2em} return false \(\triangleright\) Check for equality of references
4: \hspace{1em} end if
5: \hspace{1em} if \(a\) equals null then
6: \hspace{2em} return false \(\triangleright\) Check \(a\) for null reference
7: \hspace{1em} end if
8: \hspace{1em} if \(b\) equals null then
9: \hspace{2em} return false \(\triangleright\) Check \(b\) for null reference
10: \hspace{1em} end if
11: \hspace{1em} if \(a\).nodeName() \(\neq\) \(b\).nodeName() then
12: \hspace{2em} if \(a\).nodeName() == null || \(b\).nodeName() == null then
13: \hspace{3em} return false
14: \hspace{2em} end if
15: \hspace{2em} if \(a\).nodeName() \(\neq\) \(b\).nodeName() then
16: \hspace{3em} return false
17: \hspace{2em} end if
18: \hspace{1em} end if
19: \hspace{1em} if **TREE SIMILAR**\((a\).nextSibling, \(b\).nextSibling\) is false then
20: \hspace{2em} return false \(\triangleright\) Sibling mismatch, returns false
21: \hspace{1em} end if
22: \hspace{1em} return true \(\triangleright\) The default state is a match
23: end function

Furthermore, to reduce the scope of the similarity algorithm a subtree is generated that only contains the parent node from Level 1, and all its children from Level 2. This subtree is used to compare with other subtrees that
are derived from other siblings at the Level 1. The algorithm highlighted in Algorithm 3 contains the tree similarity function which is used to compare the subtrees generated from the Level 1 nodes as visualized in Figure 4.4. The method takes the first child of each tree \((a, b)\) and compares the equality of the references and ensure that the references contain data. The algorithm is based on a recursive approach where the method is called recursively until all the children of the subtree have been compared with the other subtree.

### 4.3 Generation of Information Retrieval Wrappers

The generation of wrappers is a multi step process which entails a number of different techniques for analyzing the data, matching and computing the similarity with the seeded background information. This process is computed once the initial document analysis algorithms have executed and the subtree for the content of interest has been identified. From the previous findings that originate from the document analysis, we can query the collection of listings so that the trees associated with the listings are stored in a data structure. The size of the data structure determines how many listings that were retrieved from the page.

#### 4.3.1 Construction of the Reference Sets

The reference sets are constructed to aid in identifying the automotive make, model and other important values published in the listings. In Table 4.1 the tables indicate the construction of the reference sets and the data stored locally.

<table>
<thead>
<tr>
<th>Make</th>
<th>Model</th>
<th>Model</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Volvo</td>
<td>240</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Volvo</td>
<td>244</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Volvo</td>
<td>245</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(A) (B)

Table 4.1: Tables contain the structure of the reference sets and the details entailed with the make, models and associated years.

The construction of the reference sets are stored in the database that include the make, and all associated models. For each model there is a collection of years that the vehicle was manufactured in. In Table 4.1 the make and model are stored in separate tables with the model referencing the make table. The production years are stored in an external table linking to the Model table. The reference set comparison is computed by (1) identifying the make, (2) identifying the model and finally identifying the production years. We make the assumption that the most common information is the make, followed by the model and the year. The tables highlighted in Table 4.1 can also be represented using the graph representation which is highlighted in Figure 4.5.

The reference sets were retrieved and processed from Edmunds and Mobile.de. To ensure that the reference sets retrieved from the sources were
4.3. Generation of Information Retrieval Wrappers

integrated, we ignored any duplicate insertions. Furthermore, for some cases we manually adjusted certain vehicle make names as we found inconsistencies in the naming scheme of the reference sets.

4.3.2 Utilising Text Extraction Techniques

The matching of the derived listings from the previous algorithms to the reference sets is done by utilizing a variety of methods. In Figure 4.6 and Table 4.2 the flow of the process is visualized and the matched results are contained in the table. The first process includes extracting the trees from the initial document analysis algorithms and store them as a collection of nodes. The next step is process the contents the text elements in each node in the tree and send it to the classifier to have it processed. When all the elements in the tree have been processed, the system automatically generates unique selectors for each node.

The methods used to retrieve the various fields from the listings are contained in Table 4.3 and the string similarity is based on the Jaro-Winkler algorithm to measure the string distance between two strings. There are a number of fields where the extraction techniques act simply as a validation tool to ensure that the content retrieved is in accordance with the HTML
standards. For instance, the \texttt{<a>} tag is used to define a hyperlink so for this particular instance retrieving the destination source of the hyperlink is simply to retrieve the value properties of the hyperlink. The destination source is contained within the \texttt{HREF} field in hyperlinks. Thus, retrieving the information required for a particular set of fields is trivial. The regular expressions in place are to validate the fetched attributes from the nodes.

\begin{table}[h]
\centering
\begin{tabular}{|l|l|}
\hline
\textbf{Content Type} & \textbf{Extraction Technique} \\
\hline
Make & String Similarity \\
Model & String Similarity \\
Year & Regular Expression \\
Price & Regular Expression \\
Image & Regular Expression \\
URL & Regular Expression \\
\hline
\end{tabular}
\caption{The right rows contain the extraction techniques to retrieve the respective content types.}
\end{table}

The string similarity is based on the Jaro-Winkler [27] distance and the mathematical equation for the algorithm is

\[ d_w = d_j + \left( lp(1 - d_j) \right) \]

where \( d_j \) is the original Jaro distance, \( l \) is the length of the common prefix to a maximum of 4 characters, \( p \) is standard weight as defined by the original work so \( p = 0.1 \). For each \texttt{listing} \( \in \texttt{listings} \) we compute the similarities using the Jaro-Winkler algorithm and if the classifier finds a match for both fields the title of the listing is locked. The classifier is configured to match any attribute that exceeds \( > 0.9 \) from the string distance computed by the Jaro-Winkler algorithm. The classifier is sent the tree as the input parameter so that it can access the node type, thus being able to infer what kind of attribute it should retrieve. For the title, which is most likely to be derived from the \texttt{a} tag it splits the title into an array. The computation for the string distance measurement is then made for each string in the string array.

The other fields are extracted by utilizing a variety of regular expressions, and text processing techniques. The classifier automatically purifies the input and removes any whitespace that may affect the comparison of the regular expressions with the pre-defined ruleset. In Table 4.4 the rules for the regular expressions are defined and there are various formats established for the respective fields as there are many different ways of formatting price depending on the region, and the site. The year for instance can either a valid year, defined as four digits or can be represented with a hyphen followed by two digits. The price should contain a formatted price that contains one to three digits followed by a dot or a comma, and the formats enlist the many formats that are recognized by the regular expression developed. This only looks at the digits and is not concerned with the currency, although it can easily be extended to include currency matching.

The last remaining fields are quite trivial, as the classifier is automatically
4.3. Generation of Information Retrieval Wrappers

<table>
<thead>
<tr>
<th>Field</th>
<th>Format(s)</th>
<th>Char Filter</th>
<th>Selector</th>
</tr>
</thead>
<tbody>
<tr>
<td>Year</td>
<td>-07, 2007</td>
<td>[0-9],[-]</td>
<td>4 digits or hyphen followed by 2 digits</td>
</tr>
<tr>
<td>Price</td>
<td>1,000, 1,000, 65,000.00, 65,000.00</td>
<td>[0-9][,]</td>
<td>1-3 digits followed by dot or comma</td>
</tr>
<tr>
<td>Image</td>
<td>http(s)://...</td>
<td>*</td>
<td>Starts with http(s) and ends with .jpg,.gif,.png etc</td>
</tr>
<tr>
<td>Link</td>
<td>http(s)://...</td>
<td>*</td>
<td>Starts with http(s) and ends with TLD or ccTLD</td>
</tr>
</tbody>
</table>

Table 4.4: The table contains the formats of the content types, and the rules defined to aid in the information extraction phase.

able to infer that the node for each of these fields is either `<img>` for images, or `<a>` for links. We pre-defined the rules for various tags, so that the classifier can automatically retrieve for instance, the destination URL for hyperlinks. Thus, the regular expressions are more for validation to ensure that the document contains a proper URI depending on the resource.

4.3.3 Traversing and Classifying the Listings

We assume that the graph rendered in Figure 4.7 is a simple graph that contains a listing. The graph representation is in digits and the digits represent the depth-first search traversal of the graph which would visit the nodes in that particular order. The depth-first implementation is tied with the `jsoup` library that is utilized for the DOM analysis. We utilise this implementation to iterate through the DOM and classify the content of each node inside the tree.

![Figure 4.7](image)

Figure 4.7: The figure visualizes in which sequence the nodes were visited by utilizing the depth-first search algorithm.

The graph in Figure 4.7 can be represented using a tree model with sample HTML tags. The idea is to illustrate how the DFS works in combination with the classifier to iterate through the nodes, and classify the node. In Figure 4.8 you can see the visualization of the tree, and the root node at Level
Chapter 4. Artefact and Algorithms

0 is a simple list item wrapped inside a `<li>` HTML tag. The elements highlighted with `#text` are simply text elements that are wrapped within their parent nodes at the upper level. We make the assumption that the traversal sent the classifier the first `<a>` element located in the top, left side of Figure 4.8.

![Figure 4.8: Tree visualization with sample HTML nodes.](image)

The classifier would automatically be able to infer that based on the node analysis of the current element, that it is a link and classify the `href` attribute of the link. Furthermore, one level deeper we can see that there is a `#text` element directly under the parent `<a>`, lets assume that the `#text` element contains the title. Post-classification findings would generate a selector for the element until the parent node, so a unique selector from the Level 2 would be generated until the root node `<li>` at Level 0.

The same logic can be applied for other HTML tags, such as images and their `src` attribute.

![Figure 4.9: The tree on the left is the representation of the HTML elements, while the right tree is represented using the respective classes.](image)

The Figure 4.9 represents the tree with the HTML elements and their respective classes. The trees are similar and represent the same data, although one can utilize both methods to navigate through the listings. In our generation of the selectors, we utilize both the HTML tag and the corresponding
class if one exists to ensure that the wrapper is precise and detailed. The reasoning for utilizing both approaches is to reduce the probability of having identical fields with similar structures, and to ensure that the information retrieval process can differentiate between the two nodes represented by the same HTML structure. For instance, on the left tree in Figure 4.9 we can see that there are two `<a>` elements that are used to wrap a text node within a hyperlink and the remaining is used to wrap the image.

![Figure 4.10](image)

**Figure 4.10:** The tree is represented by combining the HTML tag and the respective classes.

The Figure 4.10 is used to generate the selector for the HTML element and the corresponding classes. The selectors generated by the wrapper generation process are CSS selectors and the figure specifically highlights how a CSS selector can work for the attribute and the corresponding class. Furthermore, the classifier is also able to generate unique identifiers for children with no unique identifier such as `<li>` items. We assume that the element that we want to extract is the middle list item on the right side of the tree, under the `<ul>` element with the class `.listingdetails`. To generate a unique identifier, the classifier is able to determine the position of the index for the given element and utilize the `:nth-child` attribute to create a unique selector for the given index. For the particular tree highlighted in the figure the generated selector would be `li:nth-child(2)`.

**Algorithm 5** Selector generation

```plaintext
1: function RETRIEVESELECTOR(node) ▶ Node as input parameter
2:    if node has class then
3:       return node and class selector ▶ Return selector
4:    else
5:       return node ▶ Return selector
6:    end if
7: end function
```

The algorithm highlighted in **Algorithm 5** is a small extract for the selector generation. The algorithm works by generating a unique identifier given a `node` as input parameter, by utilizing the node name (HTML tag) and the corresponding classes. For multiple classes, the selector generation algorithm will automatically replace the whitespaces with dots to target multiple classes. We assume that element `<a>` is associated with class `.link`
the generated selector for this particular node would be `a[class=link]`. Suppose the element has multiple classes `.link,.bold,.italic`, the generated selector would be `a[class=.link.bold.italic]

<table>
<thead>
<tr>
<th>Rule</th>
<th>Tag</th>
<th>Classes</th>
<th>Selector</th>
</tr>
</thead>
<tbody>
<tr>
<td>attr(&quot;href&quot;)</td>
<td><code>&lt;a&gt;</code></td>
<td><code>link</code></td>
<td><code>a[class=link].attr(&quot;href&quot;)</code></td>
</tr>
<tr>
<td>attr(&quot;src&quot;)</td>
<td><code>&lt;img&gt;</code></td>
<td><code>listImg.padding</code></td>
<td><code>img[class=listImg.padding].attr(&quot;src&quot;)</code></td>
</tr>
<tr>
<td>text()</td>
<td><code>&lt;p&gt;</code></td>
<td><code>desc</code></td>
<td><code>p[class=desc].text()</code></td>
</tr>
</tbody>
</table>

Table 4.5: The table contains the extraction rules for the various elements and the retrieval method.

Continuing with the process we need to generate selectors for each node and in the table above Table 4.5 there is a demonstration of some sample elements with their associated attributes. The data retrieval for each of the entries is documented in the table, although this is for one particular node. The rule is the pre-defined knowledge that is utilised to extract the information from the element type (link, image or text element). This is dependent on the structure of the document object model and depending on the parent node, the data retrieval may differ from the table highlighted above. For instance, assume a text node is a child element to the parent `<a>` so the retrieval process must occur in the parent element `<a>` and calling the `text()` function to retrieve the text node.

<table>
<thead>
<tr>
<th>Rule</th>
<th>Tag</th>
<th>Classes</th>
<th>Selector</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;a&gt;</code> &gt; <code>href</code></td>
<td>li[class=page.listing]</td>
<td>a[class=link]</td>
<td>li[class=page.listing] &gt; a[class=link].attr(&quot;href&quot;)</td>
</tr>
<tr>
<td><code>&lt;a&gt;</code> &gt; <code>text</code></td>
<td>li[class=page.listing]</td>
<td>a[class=link]</td>
<td>li[class=page.listing] &gt; a[class=link].text()</td>
</tr>
<tr>
<td><code>&lt;img&gt;</code> &gt; <code>src</code></td>
<td>li[class=page.listing]</td>
<td>a[class=imgwrapper]</td>
<td>li[class=page.listing] &gt; a[class=link] &gt; img.attr(&quot;src&quot;)</td>
</tr>
</tbody>
</table>

Table 4.6: Complete selectors from the root node to the specified node

To demonstrate the use of the selectors from the root element (`<li>`) in Figure 4.10 the Table 4.6 was constructed and contains all the various nodes of interest. Although, the table presents the Level 0 the algorithm does not account for this particular field as the collection of listings is retrieved using
the Level 0 selector. Thus, the selector at the Level 0 field is not taken into account when generating the selectors as it is used primarily to extract all the listings and then the selectors highlighted in the table from Level 1 to $n$.

4.4 Information Retrieval using Wrappers

Provided that there exists a wrapper generated for the destination we can utilise the wrapper generation to extract the information. The previous steps will generate the selector to retrieve all listings given a document $D$. Furthermore, for each listing the process will be able to retrieve the attributes. The information retrieval process is empowered by agent architecture and for this particular retrieval we utilise the Jade framework.

4.4.1 Implementation of Agent Architecture

Our agent platform implementation was developed with a main container and other containers for wrapping the agents responsible for the document analysis, generation of wrappers and information retrieval agents using the wrappers. The advantage with using the architectural implementation proposed in this section is that the scalability aspects are improved as the containers can be placed on different machines. The architecture is suitable for the context of this project and other distributed problem-solving areas.

![High-level architecture for the implementation of the agent architecture.](image)

The Figure 4.11 is a high-level representation of the architectural implementation of the agent platform for the study. The main container has an addition of a database agent that is responsible for all the SQL operations to the database engine. Furthermore, the main container has a directory facilitator which is responsible for enabling agents to discover other agents in the platform. Agents must register with the DF service with their own area of responsibility. The agents that have successfully registered will be able to search for other agents in the platform.
The advantage with the regular containers is that they can be placed on different machines within a network and connect to the main container. The containers are layers that are concerned with (1) a single domain and (2) the multi-step information retrieval process. The containers wrap three agents which are used for the information retrieval process. The document analysis agent is responsible for parsing the rendered document, executing the computations required to generate a subtree and retrieve a unique identifier for the listings. The wrapper generation agent is responsible for utilising the text extraction techniques and generating the wrapper rules for retrieving the contents of the listings. The information retrieval agent is responsible for using the wrapper generation rules generated to retrieve the data from the destination page.

### 4.4.2 Extraction of Listings using Software Agents

The execution of the document analysis returns a class \( C \) that is used to retrieve the collection of listings \( L \), so querying the DOM using the class \( C \) populates the listings \( L \). For each \( \text{listing} \in L \) use the wrapper generation rules to extract the content.

#### Algorithm 6 Information Retrieval

1: \( \text{function CrawlPage} \)
2: \( \quad \text{doc} \leftarrow \text{page} \)
3: \( \quad \text{listings} \leftarrow \text{page listings} \)
4: \( \quad \text{for listing} \in \text{listings} \text{ do} \)
5: \( \quad \quad \text{title} \leftarrow \text{wrapper selector} \)
6: \( \quad \quad \text{url} \leftarrow \text{wrapper selector} \)
7: \( \quad \quad \text{img} \leftarrow \text{wrapper selector} \)
8: \( \quad \quad \text{price} \leftarrow \text{wrapper selector} \)
9: \( \quad \quad \text{year} \leftarrow \text{wrapper selector} \)
10: \( \quad \quad \text{ADDListing}(\text{title, url, img, price, year}) \)
11: \( \quad \text{end for} \)
12: \( \quad \text{SAVE}() \)
13: \( \text{end function} \)

The algorithm highlighted in the snippet above is a small extraction of the information retrieval process that is used to retrieve the listings through the use of the wrapper generation rules. The selectors are retrieved from a locally stored serialized object for use to extract the various content types. The file was generated by the document analysis and wrapper generation agents within the same container. The method \text{AddListing()} is used to populate a local list of listings for the crawling session. The contents of the items are purified by reducing unnecessary characters. The \text{save()} method is used to invoke a call to the database agent to store the list that was populated. The contents of the list are saved through the use of bulk database insertion to improve insertion performance.
Chapter 5

Evaluation and Analysis

5.1 Fuzzy String Matching Measurements

The fuzzy string matching measurements aim to visualize the string similarity scoring measurements that are retrieved by matching the listings with the reference sets in the database. It is important to note that we execute the same test two times with case sensitivity switched on and off to compare the findings. Additionally, the measurements visualized are the matches that the system was able to match with reference sets in the training data. The collective data was visualized in a tabular form, with green table cells determining approximate strings.

5.1.1 Similarity Measurements for Automotive Manufacturers

The results visualized in the figure below indicate the computed similarity scores for the matched values in the vertical columns, whereas the references that are stored in our training database are used to compute the similarity score. The values are sorted alphabetically to visualize the most approximate string results. Concluding, for this particular test we found

<table>
<thead>
<tr>
<th>Similarity Measurements for Automotive Manufacturers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mercedes-Benz</td>
</tr>
<tr>
<td>BMW</td>
</tr>
<tr>
<td>Audi</td>
</tr>
<tr>
<td>Volkswagen</td>
</tr>
<tr>
<td>Lexus</td>
</tr>
<tr>
<td>Toyota</td>
</tr>
</tbody>
</table>

FIGURE 5.1: Computed similarity scoring for vehicle makes with case insensitivity
that there was a 100% detection rate for all automotive manufacturers, primarily as there were no spelling errors. However, since this test was case-insensitive this affected the results. For instance there are numerous occurrences where the reference set and the matched value pair have a case-mismatch. The measurements are normalized and return values within the range of 0, to 1 where 1 indicates a perfect match. Executing the same

![Figure 5.2: Computed similarity scoring for vehicle makes with case sensitivity](image)

test with case sensitivity we find vastly different results for a number of matched values. Compared to the previous measurements with case insensitivity the algorithm was unable to locate the reference candidate due to string differences in caps. For instance, the String Mini only yields a 50% similarity when compared to MINI. The findings suggest that the algorithm is able to lock the matching reference more successfully when case insensitivity is toggled on and the string distance is strictly measuring the matching characters.

5.1.2 Similarity Measurements for Automotive Models

The findings from the similarity measurements for automotive models evidently show larger differences when compared to the automotive manufacturers. Primarily, investigating the results we see that despite executing the test with case-insensitivity we do not manage to find perfect matches for all matched values. Although, the algorithm is still able to locate the correct, most approximate string spaces, dashes or other characters may negatively impact the similarity scoring. Additionally, the most interesting result is the first matched value pair where the algorithm was able to capture two satisfying results (approximate neighbours), whereas one is a perfect match the other scores 0.92. Although, the algorithm is designed to pick the most approximate, or highest scoring reference it is still interesting as it highlights
5.1. Fuzzy String Matching Measurements

that the system could potentially match wrong pairs. Upon deeper inspection, we can see that the two matched references only differ by one digit.

Compared to the findings from the previous test with automotive manu-

FIGURE 5.3: Computed similarity scoring for vehicle models with case insensitivity

FIGURE 5.4: Computed similarity scoring for vehicle models with case insensitivity

facturers the matched values from the models test generated better values, although was unable to establish for one value. The string **fortwo** was...
Chapter 5. Evaluation and Analysis

unable to be matched successfully with ForTwo, thus generating worse results than with the case insensitivity toggled on. Primarily, the evidence generated from these measurements indicated that for this type of approximate string matching it is better to switch off the case sensitivity and have the string distance represent the number of matching characters.

5.2 Document Analysis Measures

5.2.1 Evaluation Criterias

The document analysis evaluation method aims to summarise the results from executing the document analysis algorithm whose primary goal is to identify the content of interest, compute the similarity of the children nodes and validate the identifier for the collection of the listings. The measurements presented are a summary of various HTML tags, classes and other relevant features such as pixel density on the screen surface. Additionally, the evaluation of the document analysis algorithm is used by executing it towards four platforms that publish automotive listings. The language of the platforms collectively were English, Swedish and German. The similarity scoring for sibling nodes, which are used to compute the final class that is used to identify the collection of listings is based on a similarity score above 1/3 (=66.6% rounded to 70). Collectively, the platforms publish 50 listings therefore 33 listings must match and contain the same tree structure. Thus, the tree similarity scoring which is a final step of the document analysis algorithm is satisfied for any container which generates nodes that share 70% of the same children nodes in first tree-layer. Collectively, for the measurements presented in this section a brief overview of the pages with the listing container is presented.

5.2.2 Measurements from Test 1

The initial part of the steps includes counting the HTML tags and computing the pixels occupied by the tags and their respective class. The tags

![Figure 5.5: Measurements of test 1 classes and their occupation on the page](image)

returned by the previous filtering of relevant elements are processed and
for each tag the respective classes associated are stored in a map along with the pixels occupied on the page. This particular step generates a total figure of 86 classes. The table below is restricted to the top 10 classes in terms of space occupation on the page. The figure 5.5 is used to retrieve all the elements that contain the respective classes on an individual basis. The iteration is to compute the tree similarity for all siblings and to measure the similarity rate by computing successful attempts where successful is the amount of siblings that share similar structure and the attempts is the total amount of children that the class generates. This step generates a table of 26 classes that share similar structure. The table figures highlighted

<table>
<thead>
<tr>
<th>Classes</th>
<th>Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>gh-eb-li.rt</td>
<td>86</td>
</tr>
<tr>
<td>rbx</td>
<td>69</td>
</tr>
<tr>
<td>lvsubtitle</td>
<td>100</td>
</tr>
<tr>
<td>pnl-h</td>
<td>100</td>
</tr>
<tr>
<td>gsp.right</td>
<td>100</td>
</tr>
<tr>
<td>hotness-signal.red</td>
<td>100</td>
</tr>
<tr>
<td>lvprice.prc</td>
<td>100</td>
</tr>
<tr>
<td>sresult.lvresult.clearfix.li</td>
<td>100</td>
</tr>
<tr>
<td>hotness.bold</td>
<td>100</td>
</tr>
</tbody>
</table>

FIGURE 5.6: Measurements of test 1 classes and their computed similarity score

in Table 5.6 display the similarity of the sibling nodes for the elements with the respective classes. To lock our content of interest, we retrieve the class that occupies the most pixels and has a similarity percentage over the pre-defined threshold. For this test, the identifier was dynamically set to sresult.lvresult.clearfix.li. The execution results from Figure

FIGURE 5.7: Document analysis algorithm executed on test site 1
5.7 display the rendered page with the dynamically generated identifier from the previous steps. It is clear from the results that our algorithm was successfully able to identify the listings on the page. Furthermore, analysing the page as a whole there is no additional elements that contain listings or additional attributes. Additionally the figure shows why it is not possible to assume that the listings will share identical structure. For instance, listing 2 and 5 on the figure have additional attributes such as people currently watching the listing. Furthermore, 1, 3 and 5 have an additional tag at the listing title but this is an element that is deeper than the direct children of the listing container. Although, by studying Table 5.8 we can see that the computed similarity score for the listings is 86% which means that 43 listings were identical in tree structure. The results of

![Figure 5.8: Document analysis pre-and post execution of the filtering capabilities measured in pixels and elements for test 1](image)

the document pruning measurements demonstrate the filtering capabilities measured in pixels and document elements. The document was computed to contain a total of 3007 elements, in which the listings consisted of 1003 elements generating a sub-tree of roughly 33% of the original size. The same post-execution result occupies roughly 58% of the document space. It is evident from these findings that the listings occupy the majority of the space on the document. When querying the page for the unique identifier a pruned sub-tree of the listing is generated and the DOM for test site one is generated below. The generated subtree starts from the unique identifier

![Figure 5.9: Generated DOM tree from the selection generated by the document analysis](image)

and generates a sub-tree from the $L_0$ to $L_N$. The sub-tree is used for the
5.2. Document Analysis Measures

The sub-tree does not specifically visualise the attributes and values associated with the nodes on various levels such as classes and additional node attributes.

5.2.3 Measurements from Test 2

The evaluation measurement was conducted using the same systematic process and as displayed in the figure below the algorithm was able to successfully identify the listings. Furthermore, as displayed in Figure 5.10 the language used is Swedish. The systematic process of computing the occupation area of the respective classes for all the elements on the page was conducted and generated 81 unique classes for which the top 10 classes are in the table below. The table figures highlighted in Figure 5.11 display the similarity of the sibling nodes for the elements with the respective classes. To lock our content of interest, we retrieve the class that occupies the most pixels and has a similarity percentage over the pre-defined threshold. For this test, the identifier was dynamically set to media.item_row_.ptm.pbm.nmt. The results from the measurements in test one and two for the

![Figure 5.10: Measurements of test 2 classes and their occupation on the page](image1)

![Figure 5.11: Measurements of test 2 classes and their computed similarity score](image2)
various platforms render interesting results. We can notice that the systematic structure of the presentational elements in the Swedish platform which is tested in test two renders all automotive listings with the same fields. This is proven by the similarity computation that rendered the main container with a 100% similarity index. The drawback however for using the document analysis algorithm is that the right sidebar with the collection of "sponsored" automotive listings are not detected by the algorithm. Although, in our analysis the sponsored listings on the right sidebar were able to be located on the regular listings but this required navigating through the pages using the pagination feature. The results of the document pruning measurements demonstrate the filtering capabilities measured in pixels and document elements. The document was computed to contain a total of 2898 elements, in which the listings consisted of 657 elements generating a sub-tree of roughly 23% of the original size. The same post-execution result occupies roughly 26% of the document space. It is evident from these findings that the listings despite occupying a large part of the page, they only make up a small part of the global document. We investigated this and came to the conclusion that the contributing factor for this phenomenon is due to the page layout being fixed, rather than fluid/responsive. The global
5.2. Document Analysis Measures

document is based on the entire document, and influenced by factors such as the screen resolution. When querying the page for the unique identifier a pruned sub-tree of the listing is generated and the DOM for test site one is generated below. The generated subtree starts from the unique identifier

\[
\begin{align*}
\text{article} & \quad \text{div} \\
\text{ul} & \quad \text{img} & \quad \text{header} & \quad \text{h1} & \quad \text{p} & \quad \text{footer} \\
\text{div} & \quad \text{time} & \quad \text{a} & \quad \#text & \quad \text{div} \\
\text{a} & \quad \#text & \quad \#text \\
\#text
\end{align*}
\]

**Figure 5.14:** Generated DOM tree from the selection generated by the document analysis

and generates a sub-tree from the $L_0$ to $L_N$. The sub-tree is used for the wrapper generation process. The sub-tree does not specifically visualise the attributes and values associated with the nodes on various levels such as classes and additional node attributes.

5.2.4 Measurements from Test 3

The evaluation measurement was conducted using the same systematic process and this is a secondary Swedish platform for classified ad listings. The Figure 5.15 displays the platform and the language used for this measurement. The systematic process of computing the occupation area of the respective classes for all the elements on the page was conducted and generated 47 unique classes for which the top 10 classes are in the table below.

![Figure 5.15: Measurements of test 3 classes and their occupation on the page](image-url)

<table>
<thead>
<tr>
<th>Class Description</th>
<th>Occupation Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>side-menu-animation-wrapper</td>
<td>4.12 × 10^5</td>
</tr>
<tr>
<td>site-main</td>
<td>4.12 × 10^5</td>
</tr>
<tr>
<td>layout-grid.layout-base.search...</td>
<td>3.64 × 10^6</td>
</tr>
<tr>
<td>search-result-main</td>
<td>3.2 × 10^6</td>
</tr>
<tr>
<td>item-card.wish-list.placeholder</td>
<td>2.33 × 10^6</td>
</tr>
<tr>
<td>item-card.body</td>
<td>1.72 × 10^6</td>
</tr>
<tr>
<td>item-card-image</td>
<td>1.6 × 10^6</td>
</tr>
<tr>
<td>adsense-content</td>
<td>1.3 × 10^6</td>
</tr>
<tr>
<td>adsense.adsense-active</td>
<td>9.79 × 10^5</td>
</tr>
<tr>
<td>adsense-content-block</td>
<td>4.12 × 10^5</td>
</tr>
</tbody>
</table>
The table figures highlighted in Figure 5.16 display the similarity of the sibling nodes for the elements with the respective classes. To lock our content of interest, we retrieve the class that occupies the most pixels and has a similarity percentage over the pre-defined threshold. For this test, the identifier was dynamically set to item-card.wish-list.placeholder. Compared to the findings of the first test, the measurements retrieved from the third platform share more similarities with the second test. In both the second and third test the computed findings suggest that the presentational structure of the elements are identical. From the Figure 5.17 we can see that the algorithm detected the parent container for the listings. The problem with identifying the parent container is that our algorithm for the document analysis does not check the size of the direct children under the parent node. The implementation of the algorithm could therefore be improved by taking into account the size of the children elements and if only one node is found, it should traverse one level deeper and compute the similarity. Although, the measurements and the content of interest was identified there
is room for improvement by integrating additional logic. The results of

![Circle Chart](image)

**Figure 5.18:** Document analysis pre-and post execution of the filtering capabilities measured in pixels and elements for test 3

the document pruning measurements demonstrate the filtering capabilities measured in pixels and document elements. The document was computed to contain a total of 2391 elements, in which the listings consisted of 1591 elements generating a sub-tree of roughly 33% of the original size. The same post-execution result occupies roughly 52% of the document space. It is evident from these findings that the listings occupy the majority of the space on the document. Although, compared to the previous findings for other platforms we can see that the DOM elements and their children make up the majority of the DOM tags. When querying the page for the unique identifier a pruned sub-tree of the listing is generated and the DOM for test site one is generated below.

![DOM Tree](image)

**Figure 5.19:** Generated DOM tree from the selection generated by the document analysis
5.2.5 Measurements from Test 4

The platform used for the fourth test is associated with the platform from the first test, although the language of the content is in German. Although, the platforms are associated with the same company we found that there were a number of differences in the presentation of the elements. The computation of the unique classes and their occupation on the page utilised by the HTML nodes generated 85 unique classes in total. The ten classes displayed in the table below indicate the classes with the highest document occupation. We notice that there are a number of classes shared between the platforms used in test 1 and 4. Although, the pages render different values and the respective classes occupy either more space, or significantly less. Compared to the findings of the first test when computing the similarity for the classes in the top table and the additional elements we notice minor differences in the node similarity where the main identifier has higher similarity score. The findings generated by the document analysis algorithm locks the iden-
5.2. Document Analysis Measures

identifier in this test generating a 88% similarity in comparison with 86% from the first test. Furthermore, the results from the test suggest that 44 of 50 listings on the page were similar in the HTML structure at that particular level. The results of utilising the unique identifier generates the following HTML document. The highlighted parts show the listings retrieved by utilising the identifier generated by our initial document analysis algorithm. The identifier for the listings generates the following tree from the document pruning process. There are a number of differences when compared to the first test, although a number of elements share similar structures and re-usable UI elements rendering similar document structure. The results of

<table>
<thead>
<tr>
<th>Filtered document elements</th>
<th>Filtered space (px)</th>
<th>Post-execution document</th>
<th>Post-execution space (px)</th>
</tr>
</thead>
<tbody>
<tr>
<td>58%</td>
<td>42%</td>
<td>38%</td>
<td>62%</td>
</tr>
</tbody>
</table>

**Figure 5.22:** Document analysis algorithm executed on test site 4

the document pruning measurements demonstrate the filtering capabilities measured in pixels and document elements. The document was computed to contain a total of 2345 elements, in which the listings consisted of 902 elements generating a sub-tree of roughly 38% of the original size. The same post-execution result occupies roughly 58% of the document space. It is evident from these findings that the listings occupy the majority of the space on the document. When querying the page for the unique identifier a pruned sub-tree of the listing is generated and the DOM for test site one is generated below. The generated subtree starts from the unique identifier

**Figure 5.23:** Document analysis pre-and post execution of the filtering capabilities measured in pixels and elements for test 4
and generates a sub-tree from the $L_0$ to $L_N$. The sub-tree is used for the wrapper generation process. The sub-tree does not specifically visualise the attributes and values associated with the nodes on various levels such as classes and additional node attributes.
5.3 Wrapper Generation Analysis

5.3.1 Measurements from Test 1

The wrapper generation process that was used for test 1 generated the following extraction rules that are highlighted in the table below. The table summarises the information retrieval rules that were populated as part of the wrapper generation process. The extraction rules generated by this particular step are used to retrieve the various content fields associated with the information retrieval process. To visualize the extraction rules we generated a number of figures to illustrate how the extraction results can be used to manipulate the various content fields. The results are visualized in the figure above and in this particular test we found that all elements are divided and located in separate placeholders. The title of the listing which includes the make and the model, price, year and image were all designed to be wrapped inside separate nodes. This generated extraction rules that only contain two duplicate rules for the title retrieval and the URL retrieval, which is natural since the title is wrapped within the link node so the extraction rule is both able to retrieve the title and the link simultaneously. Furthermore, the presentation layers of this platform enables our algorithm.

<table>
<thead>
<tr>
<th>Field</th>
<th>Extraction Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>h3[class$=lvtitle] &gt; a[class$=vip]</td>
</tr>
<tr>
<td>URL</td>
<td>h3[class$=lvtitle] &gt; a[class$=vip]</td>
</tr>
<tr>
<td>Image</td>
<td>a[class$=img imgWr2] &gt; img[class$=img]</td>
</tr>
<tr>
<td>Year</td>
<td>ul[class$=lvdetails left space-zero full-width] &gt; li:not([class]):nth-child(2)</td>
</tr>
<tr>
<td>Price</td>
<td>li[class$=lvprice prc] &gt; span[class$=bold]</td>
</tr>
</tbody>
</table>

TABLE 5.1: Wrapper generation rules for the content fields in test 1
to parse the contents with ease rather than using multi-field nodes which require post-processing to extract the information required. The results for identifying the listings are presented in the figure and for this particular platform all the data fields were identified correctly. We compute the mean value for the similarity scoring algorithm which is used to identify the vehicle make, and model respectively. For this particular test, the mean value is computed to be 97%. We can derive that based on the figure the wrapper generation process would be able to identify the fields for all the listings.

5.3.2 Measurements from Test 2

The wrapper generation process that was used for the second test followed the same systematic process and the generation of the information retrieval rules are highlighted in the table below. To visualize the extraction rules we generated a number of figures to illustrate how the extraction results can be used to manipulate the various content fields. For instance, rather than retrieving the text elements of the particular listing we manipulate the nodes with the extraction rules highlighted in the table above. Although, one drawback of the document analysis algorithm is that the identifier for the collection of listings which are highlighted in orange return the main listings on the page. Thus, any listings that may be sponsored, highlighted and placed outside this container will not be matched and this is clearly evident in the results from the second test which has a number of listings to the right of the main container which are not annotated in orange. However, the results for the wrapper generation process were able to identify all the content fields with one minor difference. Upon closer inspection the algorithm identified the title to contain the title of the listing (make, model) and the production year of the vehicle so the difference between the results
from this particular platform is that it contains multi-field values which our algorithm was able to identify. This is evident by looking at the information retrieval rules described in the table where the title, url and year share the same extraction rule. From previous chapters, we know that our novel approach handles multi-fields by utilizing post-processing techniques which utilise a mix of information extraction, and regular expressions to parse the information of interest. The measurement results presented in this par-

<table>
<thead>
<tr>
<th>Field</th>
<th>Extraction Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>h3[class$=lvtitle] &gt; h1[class$=h5 media-heading ptxs] &gt; a[class$=item_link]</td>
</tr>
<tr>
<td>URL</td>
<td>h1[class$=h5 media-heading ptxs] &gt; a[class$=item_link]</td>
</tr>
<tr>
<td>Image</td>
<td>a[class$=pull-left item-link nohistory image_container has_multiple_images] &gt; img[class$=item_image]</td>
</tr>
<tr>
<td>Year</td>
<td>h1[class$=h5 media-heading ptxs] &gt; a[class$=item_link]</td>
</tr>
<tr>
<td>Price</td>
<td>div[class$=media-body desc] &gt; p[class$=list_price font-large]</td>
</tr>
</tbody>
</table>

Table 5.2: Wrapper generation rules for the content fields in test 2

Figure 5.27: Wrapper generation results from the generated extraction rules for test 2
5.3.3 Measurements from Test 3

The wrapper generation process that was used for test 3 generated the following extraction rules that are highlighted in the table below. The table summarises the information retrieval rules that were populated as part of the wrapper generation process. The extraction rules generated by this particular step are used to retrieve the various content fields associated with the information retrieval process. To visualize the extraction rules we gen-
5.3. Wrapper Generation Analysis

<table>
<thead>
<tr>
<th>Field</th>
<th>Extraction Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>h3[class$\text{=item-card-details-header}] &gt; a</td>
</tr>
<tr>
<td>URL</td>
<td>h3[class$\text{=item-card-details-header}] &gt; a</td>
</tr>
<tr>
<td>Image</td>
<td>a &gt; img</td>
</tr>
<tr>
<td>Year</td>
<td>h3[class$\text{=item-card-details-header}]</td>
</tr>
<tr>
<td>Price</td>
<td>span[class$\text{=item-card-details-price}] &gt; span[class$\text{=item-card-details-price-amount}]</td>
</tr>
</tbody>
</table>

**Table 5.3:** Wrapper generation rules for the content fields in test 3

The figure below illustrates how the extraction results can be used to manipulate the various content fields. For instance, rather than retrieving the text elements of the particular listing, we manipulate the nodes with the extraction rules highlighted in the table above. From the figure below, it is evident that the wrapper generation process was able to retrieve the fields although upon closer inspection, it is also evident that a number of the listings on the page do not publish any production year at all. Furthermore, it is also evident that in the cases where platforms utilise user input to produce the title, it is often lacking in descriptive titles. Our algorithm, however, was able to determine the correct field for the year which happens to be the multi-field title. Although, in this particular platform, there were no sponsored or highlighted listings published on the page which allows the wrapper generation process to identify all the listings on the page. The measurements presented by the figure are different from the previous test platforms as the minor difference is that the majority of the listings do not contain a valid year attribute which is reflected in the results. Although, the results indicate that only 55% of the listings production year were retrieved this is misleading, as only 55% of the listings contain a production year. The algorithm was therefore able to identify all the valid years associated with the listings, generating a 100% success ratio for identifying valid...
years. Furthermore, there was a lower mean similarity score for this particular platform as the title of the listings appear to be unstructured, contain spelling errors or abbreviations for various vehicle makes which generated a lower similarity score than usual.

### 5.3.4 Measurements from Test 4

The wrapper generation process that was used for test 4 followed the same systematic process and the generation of the information retrieval rules are highlighted in the table below. To visualize the extraction rules we generated a number of figures to illustrate how the extraction results can be used to manipulate the various content fields. For instance, rather than retrieving the text elements of the particular listing we manipulate the nodes with the extraction rules highlighted in the table above. Although, what sets this particular platform apart from the previous measurements is that it does not publish the production year of the listings. For instance, on the table that contains the extraction rules it is evident that there is no extraction rule

<table>
<thead>
<tr>
<th>Field</th>
<th>Extraction Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>h3[class$=lvtitle] &gt; a[class$=vip]</td>
</tr>
<tr>
<td>URL</td>
<td>h3[class$=lvtitle] &gt; a[class$=vip]</td>
</tr>
<tr>
<td>Image</td>
<td>a[class$=img imgWr2] &gt; img[class$=img]</td>
</tr>
<tr>
<td>Year</td>
<td>li[class$=lvprice prc] &gt; span[class$=bold]</td>
</tr>
<tr>
<td>Price</td>
<td></td>
</tr>
</tbody>
</table>

**Table 5.4:** Wrapper generation rules for the content fields in test 3
5.3. Wrapper Generation Analysis

for the production year attribute. To remedy this one could approximate

\[
F \quad IGURE \quad 5.31: \quad \text{Wrapper generation results from the generated extraction rules for test 4}
\]

the production year by utilising the reference sets constructed to find the model, make and the model name with additional attributes such as the engine cubicmeters. Although, depending on the span of the production years this may generate too long production year spans to provide valuable information. The last figure visualize the results for the process of identify-

\[
\begin{array}{cccccc}
\text{Titles} & \text{URLs} & \text{Years} & \text{Prices} & \text{Images} \\
\hline
100 & 100 & 100 & 100 & 100 \\
80 & 80 & 80 & 80 & 80 \\
20 & 20 & 20 & 20 & 20 \\
0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\[
\text{FIGURE 5.32: Evaluation results for the identification of the data fields for the listings in test 4}
\]

ing various data fields, and similar to the previous platform this platform shared similarities as the platform by itself does not expose the production
It relies on user input to provide in-depth listing titles, which explains that only 20% of the listings had a production year in their listing title.

5.4 Information Retrieval Measurements

5.4.1 Measurements and Analysis for Platform 1

The test conducted for the first platform generates the following table. Inspecting the table it is evident that the information retrieval using the wrapper generation rulesets was able to retrieve the majority of the listings. The only deviation where the algorithm was unable to acquire the complete fields is the years attribute. The information retrieval process was only able to retrieve year attributes for 38 listings, with a total of 50 listings. Furthermore, it is evident from the table that the two different approaches to identifying nodes with specific classes yield the same end-result. Inspecting the source document it is evident that certain attributes for this particular platform are listed in a list. This however does not pose any issues for the information retrieval process as it is built to separate between child nodes of the list parent node. The problem lies in the non-standardized output, and where listings can contain additional attributes (such as expiry, bidding details) which interfere with the rulesets as the algorithm only supports one set of rules for extraction of data properties. Thus, the problem is apparent in the code blocks below which belong to two separate listings.

```
1 <ul>
2   <li>
3     <span class="tme">
4       <span>...</span>
5     </span>
6   </li>
7   <li>
8     From Canada
9   </li>
10  <li>
11     Year: 1968
12   </li>
13 ... 
14 </ul>
```

```
1 <ul>
2   <li>
3     From Canada
4   </li>
5   <li>
6     Year: 1975
7   </li>
8 ... 
9 </ul>
```
The code block with the highlighted field to the left contains additional attributes in comparison with the code block to the right whose format makes up the majority of the listings. Consider the retrieval of the year node for both of these code blocks and the problem is apparent. For instance, in the first code block it is wrapped inside the third child element whilst the retrieval of the year is located in the second child element. The algorithm is unable to handle edge cases such as this since the assumption was made that the output of the documents for listings that belong to one particular platform is standardized. This issue could potentially be solved by having multiple extraction rules for data properties so that multiple rules per field would attempt to retrieve the data until it is successful with the retrieval process.

5.4.2 Measurements and Analysis for Platform 2

The test for the secondary platform generated significantly worse results for the information retrieval process using the rules derived from the wrapper generation process. In our information retrieval process our information retrieval process managed to retrieve all the field properties for all the listings published. Furthermore, both methods of identifying the nodes with Data Matching classes (ending) Matching classes (containing)

<table>
<thead>
<tr>
<th>Types</th>
<th>Matching classes (ending)</th>
<th>Matching classes (containing)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Fields</td>
<td>Retrieved</td>
</tr>
<tr>
<td>Titles</td>
<td>50</td>
<td>50/50</td>
</tr>
<tr>
<td>Urls</td>
<td>50</td>
<td>50/50</td>
</tr>
<tr>
<td>Images</td>
<td>50</td>
<td>8/50</td>
</tr>
<tr>
<td>Years</td>
<td>50</td>
<td>50/50</td>
</tr>
<tr>
<td>Prices</td>
<td>50</td>
<td>50/50</td>
</tr>
</tbody>
</table>

Table 5.6: Information retrieval results for the second platform using the wrapper generation rulesets.

particular classes generated the same accuracy. What sets this particular test apart from the other tests is that there is lazy loading of the images associated with the listings. We noticed that only the first few listings of the page return valid image links, whilst the remaining nodes are only populated once the document is rendered and the objects are relevant. The relevancy of the items is dependent on the position of the document, and once the items are visible on the screen the items are populated. This could be resolved with various techniques, although with the construction of the existing solution for retrieving data it is unable to execute JavaScript as we only parse the contents of the documents. Additionally, however it must be noted that in our research we found this particular platform to present the highest similarity index for the siblings, and the most structured presentation of the listings throughout the platform.
5.4.3 Measurements and Analysis for Platform 3

The third platform test generated satisfactory results for the ability to capture title, links, image resources and price information although the algorithm was only able to identify a valid year for 52% of the listings. In our study of the results we found that this particular platform does not publish the production year associated with the listing. It is therefore important for the contributor/creator of the listing to mention the production year in the title of the listing. Due to the algorithmic design and the base in which the algorithm operates it is unable to parse the property for all fields. In order to resolve a valid production year, the algorithm would have to be redesigned to perform an additional web request to view the detailed view for each listing which does not contain a valid property in the simplistic listing view. Thus, from the 26 listings that our algorithm was able to associate a production year we reached a 100% detection ratio. In the examples below which are retrieved from the platform we show the difficulties in relying on user-based input to populate the data properties as opposed to systems exposing information in a formatted, standardized way.

"Peugeot 605 i bra skick och få miltal"
Eng: Peugeot 605 in good condition and low mileage

"Toyota Corolla 1,6 3d -98 - Ny Besiktad"
Eng: Toyota Corolla 1,6 3d -98 - New Inspected
Therefore, we conclude that this table is misleading due to the scope of the search scope in which the algorithm operates on. In all 24 remaining listings where a production year was not able to be found the solution would be to perform an additional request to open the detailed view of the listing and locate the data properties. With the existing design of the artefact it is unable to resolve the production year for this particular scenario, although it is possible for the algorithm today to resolve the years a particular model was produced in.

5.4.4 Measurements and Analysis for Platform 4

The findings from the fourth test suggest that the rules derived from the wrapper generation process successfully managed to identify a number of data properties. The only area where the algorithm was unable to locate all the properties is the years field. Although, the result is significantly better than the previous test for the third platform it also relies on user input to relay production year information. The primary problem is that the platform does not specifically expose production year, thus it is up to the creator of the listing to expose that information either in the description field, or the title. Furthermore, for some adverts there are inconsistencies where a number of users relay this information in the title, and a minor part of the listings have this information contained within the description field. The platform does not specifically expose production year, thus it is up to the creator of the listing to expose that information either in the description field, or the title. Furthermore, for some adverts there are inconsistencies where a number of users relay this information in the title, and a minor part of the listings have this information contained within the description field.

<table>
<thead>
<tr>
<th>Data Types</th>
<th>Matching classes (ending)</th>
<th>Matching classes (containing)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Titles</td>
<td>50/50</td>
<td>50/50</td>
</tr>
<tr>
<td>Urls</td>
<td>50/50</td>
<td>50/50</td>
</tr>
<tr>
<td>Images</td>
<td>50/50</td>
<td>50/50</td>
</tr>
<tr>
<td>Years</td>
<td>44/50</td>
<td>44/50</td>
</tr>
<tr>
<td>Prices</td>
<td>50/50</td>
<td>50/50</td>
</tr>
</tbody>
</table>

**Table 5.8:** Information retrieval results for the fourth platform using the wrapper generation rulesets.

In this particular example it is clear that the first advert contains a valid production year, although the second advert omits any information regarding the production year of the vehicle. The same proposed solution that was suggested for the third platform can be applied here although we achieved significantly higher coverage here. In comparasion with the findings from the previous test in which we were only able to achieve 52% detection ratio for the year property although that is 100% of the adverts which contained a valid year. In this particular test we were able to identify 88% of the year properties contained within the collection of listings. The remaining 12% consisted of 4% which had the year relayed through the description field whereas the extraction rules were specifically generated by the wrapper generation process to retrieve the year property through the title and 8% of which contained no such information. This particular platform
however presented the largest number of inconsistencies and suggests that modifications can be made to the algorithmic design to improve the results by primarily utilising fallback extraction rules for the information retrieval process. These rules are generated through the wrapper generation process.
Chapter 6

Discussion

The study has a primary research question, followed by two additional sub-questions that are described in the introduction of the study. The main research question is concerned with how we can generate information retrieval wrappers by utilizing background knowledge to retrieve automotive listings from unstructured text published on classified ad pages on the Internet. The other two research questions are concerned with the performance and the optimization of the information retrieval process, and in what ways these can be improved respectively. The last research question is concerned with the integration and alignment of data retrieved from various sources, and in what way we can integrate it into a single service.

To address the primary research question, the research study conducted demonstrates that the use of pre-processing algorithms, information extraction algorithms to identify data properties allows the construction of information retrieval wrappers tailored for each document specifically. Furthermore, the results measured for various components in the algorithm indicate that the system is well-capable of identifying various data properties, and retrieving the same values when available. Primarily, the algorithm for detecting data properties is leveraged by regular expressions, fuzzy string matching and training data that we refer to as reference sets. The reference sets are data sets for each car manufacturer, and their associated models, production years. The limitations of the pre-processing algorithm, and the wrapper generation are that the items must share similar presentational structure to be identified as part of the document analysis algorithm, but also that the relevant training data must be populated. Although, we were unable to demonstrate the extension of this research to other domains we argue that the research is generalizable. Furthermore, the key findings of this study support our claim that the research can be reproduced, extended to other documents.

To address the first sub-question, the optimisation aspects of the information retrieval process were primarily in adjusting the search scope for which the wrapper generation builds upon. With the integration of document pre-processing the measurements demonstrate the vast reduction in trees rendered by the document analysis algorithm that only returns the containers for the collection of the listings. Furthermore, the performance measurements during the study indicate that the system is performant, and scalable due to the architectural integration by splitting the workload to separate agent containers. To address the last sub-research question, the figure below indicates roughly the application area of the research study and how
the system intends to provide a unified resource for exposing this information. The goal of the system is to generate wrappers dynamically for a collection of documents from the same origin, thus identifying the various data properties associated with nodes in each document. Therefore, wrappers are generic and contain figures such as the brand/manufacturer, model, production year and price etc. With this assumption, it is therefore possible to model a generic database which allows to store this information and make it accessible through a unified service. The idea is that once the processing of the documents has been made, they are generic and the contents can be stored in the same database. The primary usage for the artefact that is described in this study is to enable documents published on the web to be queried from external services. The artefact can enable information on structured web pages to be accessible by third-party APIs by processing the information on these documents and grouping the information, provided the reference sets are populated accordingly. What specifically differentiates this study is that we combine two fields (a) information extraction and (b) wrapper generation in order to improve the performance of the information retrieval process. The other research carried out within this particular field has concentrated on improving the information extraction and utilising various techniques for wrapper generation such as document matching. We argue that there are several advantages of utilising our approach for information retrieval, specifically as the wrapper generation process determines the grouping of the information by loading the most relevant reference set, thus making the information retrieved relational, relevant and easily integrated with public APIs. The advantages are clear when compared to wrapper generation, although information extraction techniques also benefit from the same advantage as our method with significant performance costs. The majority of the information extraction techniques are resource-demanding, which we argue renders wrapper-based approaches more ideal for large-scale information retrieval from structured web pages. Although, one common method of generating wrappers is by comparing two or more documents of the same type to determine which
fields are dynamic. The objective of the document matching process is to determine which fields contain information, and the goal is to identify all string mismatches based on the source document and the remaining documents. Furthermore, to identify optional fields there is also analysis which identifies tree mismatches in the node structure of the document. In our study, the document analysis algorithm aims to identify the content of interest which has two key advantages, primarily to prune the document tree and filter information outside the bounds of the listings. Thus, the primary disadvantage of using the document-matching approach to retrieve information is that information which may not be of interest is also retrieved during the information retrieval process. As the figure indicates below, the document analysis consists of a multi-step process, in which the primary step is to identify the content of interest with a sub-tree generated of all nodes wrapped within the collection of listings. In comparison with the previous research carried out it intends to impact the performance by optimising the documents and reducing the complexity for future steps, which include the wrapper generation process. In our measurements retrieved through the evaluation of the artefact, we found that as much as 77% of the document could be removed after the initial document analysis. Collectively, the measurements suggest that it is possible to prune the documents greatly whilst retaining the information of interest. Thus, we argue that our study is relevant as it is able to retrieve the information published on these pages by utilising various techniques, such as tree similarity, information extraction techniques to generate wrappers. Furthermore, in comparison with traditional information extraction techniques they perform significantly worse in terms of performance (execution time, resource-demands).
and for large-scale information retrieval we argue that wrapper-based information retrieval is preferable. Additionally, the alignment of the information is done by utilising reference sets so the information we retrieve is automatically grouped within the domain of the reference set. Unfortunately, due to time constraints we were unable to integrate additional reference sets for other domains and develop an algorithm to determine which is the most relevant reference set for the wrapper-generation process.

### 6.0.1 Challenges and Reflection

However, during our study we acknowledged that there are a number of fatal flaws with our assumption and the current algorithmic design. Our assumption was that the content published through these platforms share the same presentational layer, but we encountered that certain platforms have optional fields which ruin the rules generated by the wrapper-generation process. The artefact is built to have single rules for extraction of data properties, although due to optional fields which impact the trees of the specific listing nodes we are unable to capture all the information. This is a major drawback when compared to existing research carried out within this field, such as information extraction alone, or document-based wrapper generation methods such as document matching, mismatching. There are multiple solutions to solve the problem of this, such as tree merging and generating unified trees for all listings. However, during the research we thought that the best approach to solve this is to be able to have a list of fallback extraction rules, should the primary rule fail. Consider the problem, where 10% of the listings have the title in a different container, the primary extraction rule would revert to the fallback solution and attempt to extract the title until the condition is satisfied. Additionally, the results also indicate that lazy loading attributes and resources is frequently common with web applications, and this poses certain challenges for algorithms that are primarily concerned with the parsing of the documents, rather than rendering the documents in web browsers. This breaks our algorithm, as it is unable to capture lazy loaded resources. This issue could however be resolved by utilising tools, such as Selenium, which is being used by the document analysis algorithm although this would affect the performance measurements derived from the evaluation significantly.

Furthermore, one key area that is worth investigating is how the artefact would scale with the integration of additional reference sets in other domains. Additionally, there has been a number of studies carried out to investigate the construction of reference sets from listings themselves to satisfactory results. The study however has proved that it is possible to identify the content of interest for documents that relate to buy, and sell platforms with great pruning effects. Furthermore, the wrapper generation is suitable for large-scale information retrieval and paired with information extraction, tree similarity algorithms we argue that the measurements derived from the evaluation prove that the artefact works as intended. Although, the key area for improvement is to design the artefact to support more than one extraction rule for data properties. Furthermore, in regards
to lazy loaded properties it might be useful to investigate automatic methods to detect scripts within the document that are executed when rendering the document.

Currently, as previously discussed there are a number of issues with the document analysis algorithm as the tree similarity scoring is based on identifying siblings that contain the same tree structure for their direct children. The problem with this assumption is that optional fields may be present even at the direct children, causing the algorithm to incorrectly present the data. It would be more appropriate to pick the listing with the least amount of immediate childrens and use that for comparison, and check that the other siblings contain the immediate children of the target. This is inherently a design flaw that was not recognized during the research project, although the findings suggest that despite the inherent flaws it was possible to recognize the identifier for the content of interest, despite the small sample size.
Chapter 7

Conclusions and Future Work

The study proves that the pairing of two techniques, information extraction and wrapper generation make it possible to generate information retrieval wrappers automatically by using information extraction techniques to identify, and label data properties in documents. The wrapper generation process consists of several sequential steps, in which one of these steps is to automatically identify the content of interest. The main goal of the study was to generate wrappers specifically for specific domain models to specifically retrieve nested, similar siblings that share similar tree structures and presentational view. In comparison to previous work within the field of wrapper generation, and those that are specifically built by using document to document matching where several documents are compared to a single document from the same origin to determine tag, and string mismatches.

We argue that this approach is too generic, and although can be used for our purpose it is more suitable for generic use cases whereas our approach is built primarily to fetch, and generate rules for domain-related data properties such as automotive manufacturers, models, years and meta data in relation to these fields that are contained within the listing. In other related studies, we find a variety of successful methods to generate wrappers with or without a user overseeing the wrapper generation process and defining the contents of interest.

In [19] Liu et al. describe a wrapper generation process primarily leveraged by computing XML files that are used to describe the extraction rules for the document structure, but requires user input for the computation of the wrapper rules. In comparison to the study presented in this paper, we use a novel approach of computing the content of interest dynamically by iterating through the document and counting the tag frequency, and the pixel density of the elements in the DOM. This requires no user input, and albeit small sample size was proven to be successful for a number of platforms as shown in the results and evaluation of the document analysis algorithm. In a similar study [30] the authors use a similar approach and seed background domain knowledge in the shape of XML types that the authors argue can be extended to other platforms. The study is similar in how they utilise background knowledge to identify, and determine the wrapper generation rule. We decided to utilise a more domain-driven approach of using reference sets and utilise fuzzy string matching algorithms to automatically identify approximate strings and identify string-based values. For price values we use a generic regular expression that is used to retrieve the price information. In addition, their approach is based on identifying
and computing the most frequent pattern which works for structured documents, although when the data is presented strictly with user-defined input fields it is more problematic as their algorithm currently computes wrapper rules by identifying, and selecting wrapper rules for structured resources.

The method we use is arguably more reliable and previous studies with the same matching technique have displayed promising results. Additionally, in comparison with previous work we render the documents entirely instead of traversing the contents on a line to line basis. We claim novelty for the pre-processing algorithms used to determine the contents of interest by primarily rendering the document and using the pixel density, and tag frequency in relation to the document and window size. To the best of our knowledge, based on research in related studies there has not been work in determining the content of interest in semi-structured documents on the Web. The tag frequency and pixel density, paired with the tree similarity algorithm allow us to determine the collection of nested elements that share similar structure. This is possible by using a strong assumption that the collection of the automotive listings published on the document will occupy the most space on the document, and that they retain similar presentational structure. In the small sample size of the measurements presented in the results and evaluation we found that the hypothesis holds true, and despite a number of flaws in the tree similarity algorithm it is possible to lock the content of interest by computing the tag frequency, and the pixel density of the elements.

In relevant studies highlighted in the background chapter, we identify a number of studies whose initial step is to identify the content of interest, although they are either based on user-defined input to highlight the areas of interest or identifying similar patterns in the document. Thus, the primary novel contribution is the pre-processing step of the document analysis algorithm which aims to identify the collection of the listings on the page. The measurements that support this claim are published in the results and evaluation. The figures that illustrate the images in the document analysis chapter prove that the document analysis algorithm was automatically able to compute the collection of the listings by following the strong statement that the collection of listings share similar structure, and occupy the most space on the document. Furthermore, most wrapper generation methods that strictly interpret the document without rendering the document are not able to determine the size of the element nodes on the document, as they are usually stated in external resources.

We argue that the wrapper generation is a complex task, which cannot be explained as a single step, thus we split the structure of the system into a sequence of steps, and together they make a unified system where domain data from several origins are merged and stored locally. We can therefore make the valid assumption that, since the results and evaluation chapter present findings for each of these steps collectively, and they are sequential steps the findings for the information retrieval which are the last step (in which the wrapper generation rules are used to retrieve listings) the findings suggest that the hypothesis is true. Although, the sample size is too small to objectively state that the algorithm and the findings from this
study can be extended to any document on the Web, it would require additional studies to prove this. Although, since the flow of the system follows a logical sequence of steps, a single failure in one of these steps would invalidate the results presented in the information retrieval and prove the hypothesis otherwise.

The findings generated from the evaluation methods generate evidence to support our hypothesis and our theory to use domain-knowledge to generate information retrieval wrappers. The nature of the problem is complex, and thus broken into a logical sequence of steps in which the majority of the steps were evaluated. As previously stated, the study primarily demonstrates that it is (a) possible to combine background knowledge in the shape of reference sets, pre-seeded information to generate information retrieval wrappers for semi-structured and structured documents, and (b) determine the content of interest in documents where nested, similar siblings are used to present the relevant content by using a novel approach where we look at the tag frequency of the elements, in combination with the pixel density and occupation on the document. This is made possible by rendering the entire document, in comparison with previous work that treat the documents on a line-to-line basis and attempt to generate navigational wrapper rules for the DOM structure.

The identification of the content of interest is seeded into the wrapper generation process, thus enforcing our claim that the study was proven successful. The sequential steps in which the research was undertaken means that any error in the chain of events, will inevitably impact the outcome of the study, and render most of the measurements retrieved invalid. The findings from the document analysis, and wrapper generation support the claim, as it demonstrates findings for each part of the artefact and presents the final findings. We did however find a number of areas in which improvement can be made, primarily the algorithm for the tree traversal which is used for the document analysis relies on strictly comparing the sibling elements for all child nodes. The analysis of the measurements described in the results indicate that this was not the case, and although the elements share large similarities there are edge cases with optional field properties. Currently, we do not have these edge cases and previous studies have been more successful at establishing optional field properties and handling them accordingly. Furthermore, our prototype identified nodes by using their type and associated classes there are edge cases in which the classes do not entirely match, one potential area to remedy this is to hold a collection of extraction rules for a single data entity.

We argue that the usage of this study, and the artefact developed is suitable for documents in which there is a collection of similar elements that relay the information desired, so that spans into categories of advertisements, listings, news aggregators, web stores, blogs. This is primarily because the document analysis algorithm makes it selection primarily on tag frequency and pixel occupation on the screen surface. The only limiting factor in this is that the wrapper generation process is limited to reference sets, and the construction of these can be time consuming depending on the content of interest. We have extended the research into other domains with success,
although due to time constraints and limitations we were unable to document the findings and prove the extensibility and the generic nature of the artefact proposed in this study.

7.0.1 Future Work

The work within this field is rapidly changing and with the recent shift towards single page applications and dynamic interfaces presents new challenges. Information published on the Web is largely unqueryable primarily because there is a lack of adoption of the semantic web technologies. We argue that the method proposed here is a great way of exposing large chunks of unqueryable data through the use of wrappers tailored for documents on the Web specifically. The study however needs more extensive testing and evaluation with multiple reference sets. There is large amount of work that can be put towards constructing reference sets automatically from the listings themselves, by the use of machine learning and a variety of other algorithms. Previous studies indicate that this is possible, and the results indicate that it can be used to great success.
Appendix A

Performance Measurements

**Figure A.1:** Visualized measurements of the document analysis

**Table A.1:** Measurements for the execution of the document analysis

<table>
<thead>
<tr>
<th>Test</th>
<th>Exec. speed (s)</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>63 seconds</td>
<td>3007</td>
</tr>
<tr>
<td>1</td>
<td>70 seconds</td>
<td>3007</td>
</tr>
<tr>
<td>2</td>
<td>48 seconds</td>
<td>2898</td>
</tr>
<tr>
<td>2</td>
<td>50 seconds</td>
<td>2898</td>
</tr>
<tr>
<td>3</td>
<td>47 seconds</td>
<td>2391</td>
</tr>
<tr>
<td>3</td>
<td>54 seconds</td>
<td>2391</td>
</tr>
<tr>
<td>4</td>
<td>55 seconds</td>
<td>2345</td>
</tr>
<tr>
<td>4</td>
<td>60 seconds</td>
<td>2345</td>
</tr>
</tbody>
</table>

**Figure A.2:** Visualized measurements of the wrapper generation process

**Table A.2:** Measurements for the execution of the wrapper generation process

<table>
<thead>
<tr>
<th>Test</th>
<th>Exec. speed (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2335</td>
</tr>
<tr>
<td>1</td>
<td>2541</td>
</tr>
<tr>
<td>2</td>
<td>1421</td>
</tr>
<tr>
<td>2</td>
<td>1598</td>
</tr>
<tr>
<td>3</td>
<td>1186</td>
</tr>
<tr>
<td>3</td>
<td>1241</td>
</tr>
<tr>
<td>4</td>
<td>2238</td>
</tr>
<tr>
<td>4</td>
<td>2437</td>
</tr>
</tbody>
</table>
Appendix B

CPU Usage Measurements

CPU usage for the document analysis

CPU usage for the wrapper generation process
Appendix C

Memory Usage Measurements

Memory usage for the document analysis

Memory usage for the wrapper generation process
Bibliography


