Advances in Nonlinear Variational Inequalities
ISSN: 1092-910X
Vol 27 No. 3 (2024)

Optimizing Data Extraction using Preprocessing for Enhanced Efficiency

Santosh V. Chobe[1], Swati Nikam[2]

[*1]Pimpri Chinchwad College of Engineering & Research, Ravet, Pune; sanchobe@gmail.com

[2]Pimpri Chinchwad College of Engineering & Research, Ravet, Pune; swatinikam3@gmail.com

[*]Correspondence: sanchobe@gmail.com

Article History:

Received: *09-03-2024*

Revised: *03-05-2024*

Accepted: *29-05-2024*

Abstract:

With the exponential growth of the internet, an abundance of information has become readily available. Extracting valuable data from the web is crucial for applications such as meta-querying and comparison shopping. However, the heterogeneous nature of web information poses a significant challenge to the extraction process. The web can be classified into the surface or visible web and the deep or invisible web. While conventional search engines can index the surface web, they fall short when it comes to the deep web.

To access the deep web, users must submit queries to web databases, and the results are encapsulated in dynamically generated web pages containing data records. Traditional search engines struggle to index these dynamic pages, necessitating a specialized program for efficient information extraction from the deep web. Web search engines generate result pages based on user queries, making it crucial to automatically extract data from these pages for various applications.

In this context, we propose an innovative data extraction method called Effective Data Extraction using Preprocessing (EDEP). The EDEP approach begins by parsing the input HTML page, constructing a tag tree, and subsequently eliminating irrelevant tags from the tree. Notably, our system efficiently handles scenarios where auxiliary information, such as recommendations or comments, is intermixed between query result records (QRRs), causing them to be non-contiguous. EDEP also effectively manages result pages containing single QRRs.

Through experimental results, it is evident that EDEP outperforms existing data extraction methods, showcasing its efficacy in handling the complexities associated with web data extraction.

Keywords: Deep Web, Information extraction, Surface Web, Web mining, Wrapper induction.

I. INTRODUCTION

The wealth of information available on the Internet is experiencing exponential growth, establishing the Internet as the most extensive knowledge base ever created and openly accessible. However, harnessing this information becomes a challenge due to its unstructured

or semi-structured nature. Web pages, designed for human browsing, present a hurdle for applications seeking access.

The Internet can be categorized into the surface or visible Web and the deep Web or invisible Web. The surface Web, also known as the visible Web, refers to the section indexable by conventional search engines [16, 20]. In contrast, the deep Web remains beyond the reach of these search engines [17, 21]. To delve into the deep Web, users submit queries via a query interface to Web databases, and the ensuing results are encapsulated in dynamically generated Web pages constituting data records. The dynamic nature of these pages, referred to as query result pages, poses indexing challenges for conventional search engines.

To leverage the valuable information within these query result pages, extraction becomes imperative. Automatic data extraction is a requisite for various applications, including meta-querying and comparison shopping. The accuracy of data extraction plays a pivotal role in ensuring the seamless functionality of these applications.

The remaining paper is organized as follows:

Section 2 provides a comprehensive summary of various approaches to data extraction, offering insights into the diverse methodologies employed in this field. Section 3 conducts a thorough review of related work, encompassing an examination of various tools and techniques employed in the realm of data extraction. In section 4 the details of the proposed methodology are elaborated, explaining the steps and processes involved in the innovative approach presented in this study. Section 5 presents the experimental results obtained from the implementation of the proposed methodology, accompanied by a detailed discussion of the findings. Finally, section 6 concludes the paper by summarizing key insights and contributions, offering a conclusive perspective on the proposed methodology and its implications for the broader field of data extraction.

II. Information Extraction

In the domain of Information Extraction (IE), the primary emphasis often revolves around specific websites, while Web mining delves into uncovering patterns from diverse Web documents. Web information extraction addresses the challenge of discovering relevant information from the vast expanse of the Web, and a crucial tool in this process is a program known as a *Wrapper*.

Information extraction from the Web is essential for aggregating and integrating data from various sources, enabling functionalities such as meta-querying and comparison shopping. With the increasing number of organizations disseminating information on the Web, the extraction of data from these web pages has become a significant necessity.

There exist three principal approaches to data extraction:

i. **Manual approach:** In this approach, a human programmer manually crafts a program by scrutinizing a web page and using pattern specification languages to extract the desired data. However, this approach can be cost-intensive, demanding that users possess substantial programming knowledge to effectively utilize such tools.

ii. **Wrapper induction:** This technique adopts a semi-automatic and supervised learning approach. A set of training examples, manually labeled, is employed to learn data extraction rules. The manual labeling involves marking the data items on the training pages that the user intends to extract.

iii. **Automatic extraction:** Working as an unsupervised approach, automatic extraction automatically discovers patterns or rules for data extraction. This method eliminates the need for manual efforts in labeling, enabling it to perform data extraction on a large scale without significant manual intervention.

III. Related Work

The field of information extraction from the web has seen considerable research and development, with various approaches and methodologies being explored. In this section, we provide an overview of related work, highlighting key contributions and insights from existing literature.

WIEN: Wrapper Induction for ENtities

Introduced by Kushmerick et al. [7], WIEN represents one of the pioneering wrapper induction systems designed for web information extraction. This system is characterized by its utilization of predefined extractors [8] and places a primary focus on extractor architectures. WIEN incorporates a set of predefined extractors, contributing to its ability to learn wrappers based on labeled information from a designated set of training pages. The core functionality of WIEN involves learning a wrapper, a crucial component for extracting relevant data from web pages. This learning process is driven by the manual labeling of relevant information in the training pages. A notable limitation of WIEN is its assumption that attributes are ordered within a data record. This assumption imposes constraints on scenarios where attributes may not follow a specific order, impacting the adaptability of WIEN in certain contexts [15].

WIEN faces challenges when dealing with missing values in the data extraction process. The inability to handle missing values represents a limitation that may impact the system's performance in real-world scenarios. WIEN-generated wrappers have constraints when it comes to handling nested structures within web pages [10]. The system may encounter difficulties extracting information from nested elements, limiting its applicability in situations where nested structures are prevalent. Despite its limitations, WIEN stands as an early contribution to the field of wrapper induction systems, offering insights into extractor architectures and laying the foundation for subsequent advancements in web information extraction methodologies. Researchers and practitioners have since built upon these insights to address the identified limitations and enhance the overall effectiveness of wrapper induction systems.

RAPIER: Rule-Based Information Extraction System

RAPIER is a rule-based information extraction system known for its ability to learn rules for extracting information from documents. Unlike some systems, RAPIER operates without the

need for prior parsing or post-processing during the learning phase. It employs a bottom-up relational learning algorithm [5], and its primary focus is on field-level extraction. RAPIER utilizes a bottom-up relational learning algorithm to derive extraction rules from documents. This approach enables the system to learn without the necessity for pre-parsing or post-processing, streamlining the information extraction process. RAPIER concentrates on field-level extraction, aiming to capture specific information within documents. This targeted approach enhances the system's precision in identifying and extracting relevant data. The extraction rules in RAPIER are structured into three slots, providing a comprehensive framework for matching different parts of the document. These slots include:

- **Pre-filler:** Matches the text to the left of the filler.
- **Slot Filler:** Matches the actual content or information to be extracted.
- **Post-filler:** Matches the text to the right of the filler.

RAPIER adopts a dynamic approach to rule learning. It starts with specific rules but evolves towards more general rules as the learning process progresses. This adaptability enhances the system's ability to handle diverse document structures. RAPIER is termed "single-slot" because it is designed to handle a single record from each input document. This characteristic positions it effectively for scenarios where extracting isolated pieces of information is the primary requirement [4, 10]. RAPIER's distinctive features, such as its rule structure and emphasis on field-level extraction, make it a valuable tool for targeted information extraction tasks. Its ability to adapt rules dynamically contributes to its effectiveness in handling diverse document structures and content formats.

WHISK: Multi-Slot Extraction with Regular Expressions

WHISK, introduced in [4], is a system designed for generating multi-slot extraction rules for documents. The rules in WHISK are generated through a covering algorithm, relying on regular expression patterns to identify the context of relevant phrases. Notably, WHISK exhibits versatility by being capable of handling both highly structured and unstructured text documents. WHISK employs a covering algorithm to generate multi-slot extraction rules for documents. This algorithm uses regular expression patterns to identify the context of phrases deemed relevant for extraction. WHISK's flexibility allows it to handle a wide range of document structures, accommodating both highly structured and unstructured text documents. This adaptability makes WHISK applicable to diverse information extraction scenarios. WHISK utilizes a supervised learning algorithm to induce novel rules from a set of hand-tagged instances. The learning process involves the interleaving of rule induction and instance annotation to optimize human effort, demonstrating an iterative learning approach [19]. WHISK adopts a strategy where the most general rule covering all instances is initially induced. This rule is then systematically extended by adding terms one by one, contributing to a refined set of rules [10].

SRV: Single-Slot Extraction Rules with Naive Bayes Classifier

In [6], Freitag introduces SRV, a system focused on generating single-slot extraction rules. SRV takes tagged documents as input, extracting features that describe tokens potentially extractable from a document. The system follows a process of tokenization on input documents, labeling tokens as extraction targets or not. Available features are then utilized to express extraction rules, and a Naive Bayes classifier is employed in conjunction with a relational learner for constructing new rules. SRV is specifically designed for generating single-slot extraction rules, making it suitable for scenarios where extracting isolated pieces of information is the primary objective [1, 10]. SRV takes tagged documents as input and extracts features that describe tokens eligible for extraction. This feature extraction process contributes to the definition of extraction rules. The input documents undergo tokenization, and each token is labeled as either an extraction target or not. This labeling process is foundational for subsequent rule generation. SRV leverages a Naive Bayes classifier in conjunction with a relational learner to construct new rules. This combination of algorithms enhances the system's ability to detect patterns and generalize extraction rules.

Both WHISK and SRV showcase distinctive approaches to information extraction, with WHISK specializing in multi-slot extraction rules with regular expressions and WHISK focusing on single-slot extraction rules using a combination of tokenization, labeling, and machine learning algorithms.

SoftMealy: Learning-Based Web Data Extraction

In the work presented in [2], Hsu and Dung introduced SoftMealy, a learning-based Web Data Extraction platform. SoftMealy adopts a bottom-up inductive approach, relying on finite-state transducers (FST) for learning extraction rules. The use of FST allows for variations in the structures of extractors and facilitates the handling of missing attributes and attribute combinations in the input. SoftMealy employs a learning-based approach to extract information from web pages. The platform learns extraction rules through a bottom-up inductive process, enhancing its adaptability to diverse web page structures. FST plays a pivotal role in SoftMealy's approach. It consists of two parts:

- **Body Transducer:** Used for the part of the page containing tuples.
- **Tuple Transducer:** Employed for extracting tuples from the body.

SoftMealy is designed to handle variations in the structures of extractors. This flexibility allows it to adapt different document layouts and structures, enhancing its utility for a wide range of web data extraction tasks.

STALKER: Hierarchical Data Extraction Wrapper Induction

Muslea et al. [3, 9] introduced STALKER, a supervised learning hierarchical data extraction wrapper induction system. STALKER identifies the intended information for extraction by manually positioning a set of tokens on the web page. Notably, STALKER can handle empty values, hierarchical structures, and non-ordered items. STALKER utilizes a supervised learning approach to induce hierarchical data extraction rules. This approach enhances the

system's ability to generalize and adapt to different web page structures. STALKER introduces the concept of an Embedded Catalog (EC) tree, which describes the structure of semi-structured documents. Attributes for extraction are represented by leaves, while internal nodes represent lists of tuples. STALKER is designed to handle scenarios involving empty values and hierarchical structures. This adaptability increases its effectiveness in extracting information from diverse web page layouts.

CTVS: Automatic Data Extraction from Query Result Pages

CTVS, as described in [18], is a system designed for automatic data extraction from query result pages. It involves two main steps: record extraction and record alignment.

1. Record extraction – It takes place by following the steps explained below -

In Tag Tree Construction step, a tag tree is built for a query result page. A tag in an HTML page is represented by each node in the tag tree and tags enclosed inside it are children of the node.

Next, all potential data regions are identified by the Data Region Identification module that contains data which is generated dynamically. Data regions are identified by Record Segmentation module and then segmented into data records based on tag patterns. From these segmented data records, the data regions that contain similar records are merged by Data Region Merge module. Finally, in the Query Result Section Identification step, one data region from merged ones is selected that contains QRRs.

2. Record alignment - In this step, the attribute values of the QRRs in query result page are aligned into a table so that these values for the same attribute are aligned into the same table column.

QRR alignment takes place by following three steps explained below -

i. Pairwise QRR alignment – In this step, the data values are aligned in a pair of QRRs.

ii. Holistic alignment - In this step, the data values in all the QRRs are aligned.

iii. Nested structure processing - In this step, the nested structures that are present in QRRs are identified.

CTVS compares and aligns the QRRs, but it requires that the result page contains more than one QRR. This is one of the limitations of CTVS. It is unable to extract data from the Web page shown in Figure 1 as it contains single QRR. CTVS provides better data extraction accuracy over existing data extraction methods [18].

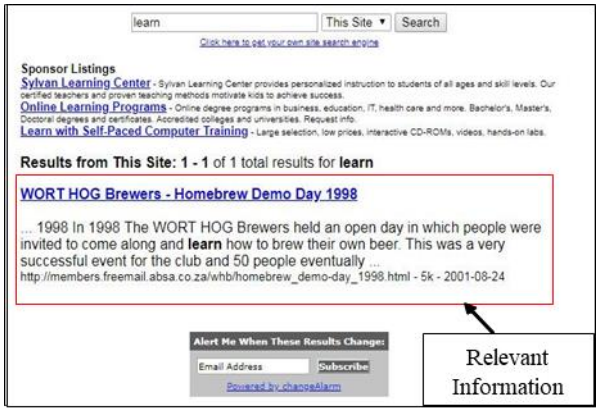

Figure 1: A page with single QRR with no repetitive information/pattern

IV. Proposed Work

In the proposed system architecture of Effective Data Extraction using Preprocessing (EDEP), presented in Figure 2, several modules contribute to the data extraction process. These modules include Parsing, Tag Tree Generation, Data Filtering, and Data Region Extraction. The system focuses on preprocessing by removing irrelevant tags and extracting relevant data records for display and storage in a database.

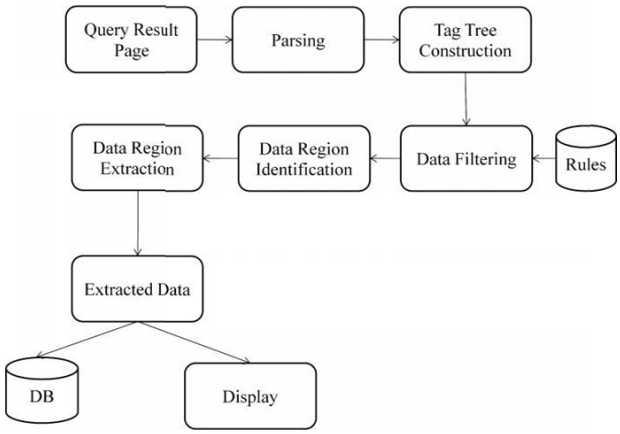

Figure 2: Proposed Architecture of EDEP

Algorithm shown in Figure 3 generates a tag tree of given HTML page and removes the unwanted tags from it (lines 1-3). It then removes the single child of a node recursively, if present (lines 4-5).

```
Procedure Main()
1.    input HTML page
2.    generate tag tree
3.    remove unwanted tags
4.    if a node has a single child then
5.          remove the child recursively
6.    dataRecordInfo = detectPattern()
7.    sorted_list_of_nodes = sortElements(dataRecordInfo)
8.    node = getRelevantElements(sorted_list_of_nodes)
9.    store extracted records into a file as well to the database
```

Figure 3: main algorithm

In the function *detectPattern*, shown in Figure 4, the tag tree is traversed in a depth- first manner. During the visit to each node, the information of the node like tag name, number of children, tags of children etc. is collected and stored in a node list called *dataRecordInfo* (line 1). The *nodeCompare* method compares the nodes from *dataRecordInfo* and each node from the tag tree and returns true or false depending upon the similarity between the nodes (lines 2-3). If the nodes are similar then the HTML contents of the tag are appended to existing corresponding node content in the *dataRecordInfo* after incrementing the node counter otherwise a new node entry is made in the *dataRecordInfo*. The algorithm returns nodes stored in *dataRecordInfo* (lines 4-9).

```
Function detectPattern()
1.    dataRecordInfo = details of a node like tag name, number
of children, tags of children etc.
2.    for each node from tagtree
3.    val = nodeCompare (node);
4.    if (val) then
5.          increase node counter of previous similar tags
append HTML contents of the node to previous similar node
contents in the dataRecordInfo
7.    else
8.          add a new node to the dataRecordInfo
9.    return dataRecordInfo
```

Figure 4: detect pattern algorithm

The algorithm *sortElements* shown in Figure 5 rearranges the nodes based on a count of number similar nodes and returns the same. Finally, algorithm *getRelevantElements*, shown in Figure 6, finds and returns the node that has maximum text length.

```
Function sortElements (dataRecordInfo)
1.    Arrange the nodes in non-increasing order of number of
similar nodes
2.    return sorted_list_of_nodes
```

Figure 5: sort elements algorithm

```
Function getRelevantElements(sorted_list_of_nodes)
1.  find the node with maximum text length
2.  return the node with maximum text length
```

Figure 6: get relevant elements algorithm

V. Experiments

Experiments similar to the ones detailed in [13] and [18] were conducted to evaluate the performance of EDEP using three datasets and compared with DeLa [11], ViPER [13], ViNTs [14] and CTVS [18]. EDEP is implemented in JAVA and executed on Intel Core i3 2.4GHz CPU with 4 GB memory.

5.1 Datasets

The following datasets are used for performance comparison of EDEP, CTVS, ViNTs, and DeLa.

Dataset 1 (PROFUSION) - It contains 100 websites collected from profusion.com. 10 queries are submitted and the first 10 result pages are collected manually. Also, a nonexistent term is submitted as a query to the website to get a no-result page [14].

Dataset 2 (TBDW) - It provides the TestBed for extraction of information from deep Web. It has 51 online databases from which five result pages are created for each database. Like CTVS, we directly use the first page in each database as the test page [12].

Dataset 3 (VSDR) - It contains 100 websites in four categories namely education, government, medical, and general. Yahoo search engine was used to collect websites in education category. The query provided was "American universities" and the websites were selected randomly. Search.com was used to collect websites in government and medical categories. Search engines like Google, AltaVista, Yahoo, etc. were used to collect websites in general category [14].

From the above datasets, three different types of pages were considered viz., general dataset, a dataset that consists of pages with single QRR and the dataset that contain QRRs with auxiliary information like discount information, recommendation, or advertisement.

5.2 Performance Measures

Performance of our system, for extracting QRRs, is measured by using standard measures - precision and recall that are defined as follows:

$$\text{precision} = \frac{Cc}{Ce}$$

$$\text{recall} = \frac{Cc}{Cr}$$

Where *Cc* is the number of correctly extracted QRRs, *Ce* is the number of extracted QRRs, and *Cr* is the total number of QRRs present in result pages.

5.3 Experimental Results

Table 1 shows details of result pages which consist of single QRR. CTVS is unable to extract the records from these result pages whereas EDEP could extract QRRs from the three datasets as shown in Table 2.

Table 1: Data Set Properties for Single QRR

Dataset	**PROFUSION**	**TBDW**	**VSDR**
Total Number of QRRs	6912	255	485
Number of pages with Single QRR	21	12	17

Table 2: Comparison of Performance for the Result pages with single QRR

Dataset	**PROFUSION**		**TBDW**		**VSDR**	
Number of pages with Single QRR	21		12		17	
Method	EDEP	CTVS	EDEP	CTVS	EDEP	CTVS
Number of extracted QRRs	14	0	9	0	13	0
Precision (%)	30.43	0	64.29	0	34.29	0
Recall (%)	66.67	0	75	0	76.47	0

However, the precision values for EDEP are lower when dealing with pages containing a single QRR. EDEP operates under the assumption that the major part of a web page is occupied by relevant information. In cases where relevant information is small, and irrelevant information has a larger text, EDEP may incorrectly identify irrelevant information as relevant and extract it. A page shown in Figure 1 contains both relevant and irrelevant information. EDEP extracts relevant information properly from this page because the text length of the relevant information is larger than that of the irrelevant information. However, a page shown in Figure 7 also contains relevant and irrelevant information, but the irrelevant information has a larger text than the relevant information. In this case, EDEP extracts the irrelevant information, as its text length is larger.

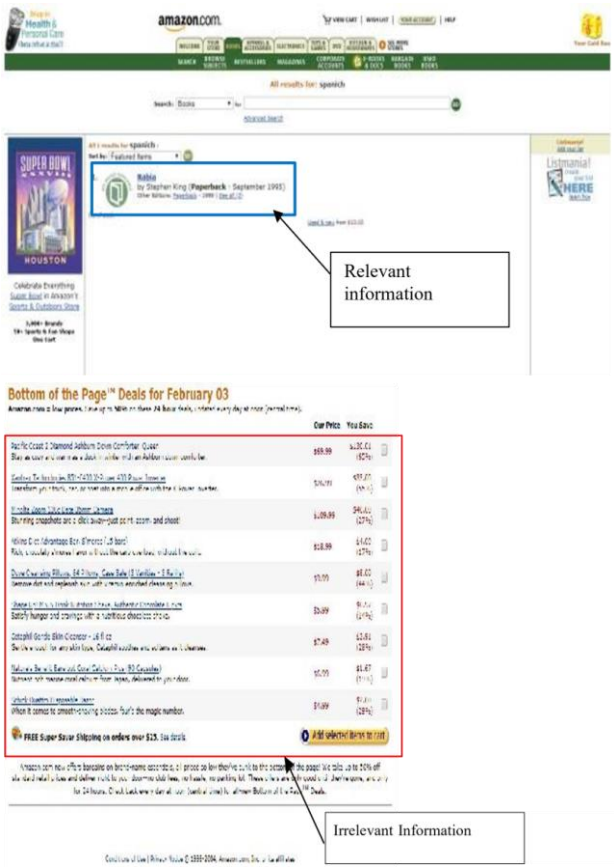

Figure 7: A page with single QRR with irrelevant repetitive information/pattern

VI. Conclusion

Extracting relevant information from web pages is a challenging task due to the presence of numerous pieces of irrelevant information. Existing methods like Dela and CTVS have been reported in the literature. CTVS appears to be a robust method but faces challenges in extracting relevant information from web pages with a single QRR.

EDEP (Effective Data Extraction using Preprocessing) is introduced as a two-step process for automatic data extraction from result pages. In the first step, a tag tree is built for the result page, and irrelevant tags are removed through preprocessing. In the second step, EDEP extracts records from the preprocessed pages and stores them in a database for further use. Experimental results demonstrate that EDEP outperforms existing methods, particularly in extracting information from web pages containing a single QRR. Precision and recall metrics are provided for three datasets, indicating substantial improvement over CTVS, especially in scenarios with a single QRR. Precision values for EDEP are reported as 30.43%, 64.29%, and 34.29% across the tested datasets. Recall values for EDEP are reported as 66.67%, 75%, and 76.47% across the tested datasets. Notably, CTVS shows 0% precision and recall for the same scenario.

EDEP's capability to extract information from web pages with a single QRR is highlighted, but it is noted that success depends on the length of relevant information being

higher than irrelevant information. Overcoming this limitation is identified as a potential area for future work.

Discussion:

The challenges mentioned highlight a limitation of EDEP, particularly in scenarios where the assumption about the major part of the page containing relevant information may not hold true. In cases where irrelevant information dominates in terms of text length, EDEP might struggle to accurately identify and extract the relevant content. This information provides insights into the behavior of EDEP and indicates a potential area for improvement or consideration in scenarios where the size of relevant information is relatively small compared to irrelevant content with larger text.

References

[1] Freitag, D. Information extraction from HTML: Application of a general machine learning approach. *In AAAI/IAAI*, 517-523, (1998).

[2] Hsu, C. N., & Dung, M. T. Generating finite-state transducers for semi- structured data extraction from the web. *Information systems*, *23(8)*, 521-538, (1998).

[3] Muslea, I., Minton, S., & Knoblock, C. A hierarchical approach to wrapper induction. *In Proceedings of the third annual conference on Autonomous Agents*, 190-197, (1999).

[4] Soderland, S. Learning information extraction rules for semi-structured and free text. Machine learning, *34(1-3)*, 233-272, (1999).

[5] Mooney, R. Relational learning of pattern-match rules for information extraction. *In Proceedings of the Sixteenth National Conference on Artificial Intelligence*, Vol. 334, (1999).

[6] Freitag, D. Machine Learning for Information Extraction in Informal Domains. *Machine learning*, *39(2-3)*, 169-202, (2000).

[7] Kushmerick N. Wrapper induction: Efficiency and expressiveness. *Artificial intelligence*, *118(1-2)*, 15-68, (2000).

[8] Baumgartner, R., Flesca, S., & Gottlob, G. Visual Web Information Extraction with Lixto. *In The VLDB Journal*, (2001).

[9] Muslea, I., Minton, S., & Knoblock, C. A. Hierarchical wrapper induction for semistructured information sources. *Autonomous Agents and Multi-Agent Systems*, *4(1-2)*, 93-114, (2001).

[10] Laender, A. H., Ribeiro-Neto, B. A., Da Silva, A. S., & Teixeira, J. S. A brief survey of web data extraction tools. *ACM Sigmod Record*, *31(2)*, 84-93, (2002).

[11] Wang, J., & Lochovsky, F. H. Data extraction and label assignment for web databases. *In Proceedings of the 12th international conference on World Wide Web*, 187-196, (2003), ACM.

[12] Yamada, Y., Craswell, N., Nakatoh, T., & Hirokawa, S. Testbed for information extraction from deep web. *In Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, 346-347, (2004), ACM.

[13] Simon, K., & Lausen, G. ViPER: augmenting automatic information extraction with visual perceptions. *In Proceedings of the 14th ACM international conference on Information and knowledge management*, 381-388, (2005), ACM.

[14] Zhao, H., Meng, W., Wu, Z., Raghavan, V., & Yu, C. Fully automatic wrapper generation for search engines. In Proceedings of the 14th international conference on World Wide Web, 66-75, (2005), ACM.

[15] Chang, C. H., Kayed, M., Girgis, M. R., & Shaalan, K. F. A survey of web information extraction systems. *IEEE Transactions on Knowledge and Data Engineering*, *18(10)*, 1411-1428, (2006).

[16] Liu, B. Web data mining: exploring hyperlinks, contents, and usage data. *Springer Science & Business Media*. (2007).

[17] Hong JL, Siew EG, Egerton S. ViWER-data extraction for search engine results pages using visual cue and DOM Tree. *In 2010 International Conference on Information Retrieval & Knowledge Management (CAMP)*, 167-172, (2010), IEEE.

[18] Su, W., Wang, J., Lochovsky, F. H., & Liu, Y. Combining tag and value similarity for data extraction and alignment. *IEEE Transactions on knowledge and Data Engineering*, *24(7),* 1186-1200, (2012).

[19] Ferrara, E., De Meo, P., Fiumara, G., & Baumgartner, R. Web data extraction, applications and techniques: A survey. *Knowledge-based systems, 70*, 301-323, (2014).

[20] "What is the difference between the Surface Web, The Deep Web and the Dark Web?" https://resources.infosecinstitute.com/topic/what-is-the-difference-between-the-surface-web-the-deep-web-and-the-dark-web/ Retrieved 2023-06-15.

[21] Patil, T.A. and Chobe, S., 2017, August. Web Crawler for searching Deep web sites. In *2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA)* (pp. 1-5). IEEE.