Vrije Universiteit Amsterdam

Bachelor Thesis

Extract text from HTML pages for advanced relation extraction

---

Author: Koki Hirose (2697989)

1st supervisor: Jacopo Urbani
2nd reader: Kees Verstoep

A thesis submitted in fulfillment of the requirements for the VU Bachelor of Science degree in Computer Science

August 4, 2023

Abstract

Developing highly efficient techniques for extracting text from HTML pages holds significant potential advantages as well as challenges due to the complexities of web pages. This paper introduces automated text extraction through HTML, which employs effective methods such as HTML parsing and noise elimination, to facilitate relation extraction. The HTML parsing approach focuses on decoding HTML text contents and tag filtering to eliminate unnecessary information while preserving relevant content securely. In the noise elimination stage, the text is stripped and prioritized, centralizing the text content to optimize extraction for relation extraction. For this purpose, the chosen model is the Open IE system provided by StanfordCoreNLP, which targets the extraction of subject-verb-object triples. To systematically evaluate the efficacy of the proposed methods, a series of comprehensive experiments were conducted on ten real HTML-based websites. The obtained results were meticulously compared against those of a baseline model, which deployed a simplistic approach, involving direct text extraction from HTML pages without any of the three processes: HTML parsing, tag filtering, or noise elimination. The findings reveal that our model successfully achieved reduced extracted text size, improved runtime, and higher accuracy compared to the baseline model. These outcomes demonstrate the model's capability to overcome the intricacies of HTML formatting and attain accurate text extraction results. The code for this project is available at `https://github.com/Hikouki0408/BSc_thesis_VU`

Keywords: NLP, Information-Retrieval, HTML-Parsing, Tag-filtering, Automated-Extraction, Translation, Summarization, Web-Information, Open IE.

1. Introduction

In recent years, the exponential growth of online content has underscored the importance of information retrieval and knowledge extraction. HTML (Hypertext Markup Language) serves as the foundation for creating visually appealing, interactive, and accessible content on the web, powering diverse platforms, for instance, online articles, social media, and E-commerce websites. However, due to the combination of textual information and structural components on HTML pages, automated text extraction through HTML faces significant challenges. Automate text extraction is reasonably considered one of the most important tasks for modern digital applications, including Information Retrieval and Automated Content Generation. Within the context of Information Retrieval, it plays a crucial role in enabling the development of algorithms and search engines that efficiently retrieve relevant information from vast amounts of textual data. By automating the process of extracting meaningful content from sources, such as HTML pages, these systems can provide users with quick and accurate access to the information they seek (Deepack and Sharma, 2012)[1]. Additionally, automated content generation has seen a significant surge in demand, driven by its applications in various areas. Content summaries are one prominent use case where algorithms automatically generate concise and informative summaries of larger pieces of text. This capability proves to be invaluable, especially when dealing with large volumes of information. Another important application can be found in multilingual content translation. Given the increasing global interrelation, the need for accu-

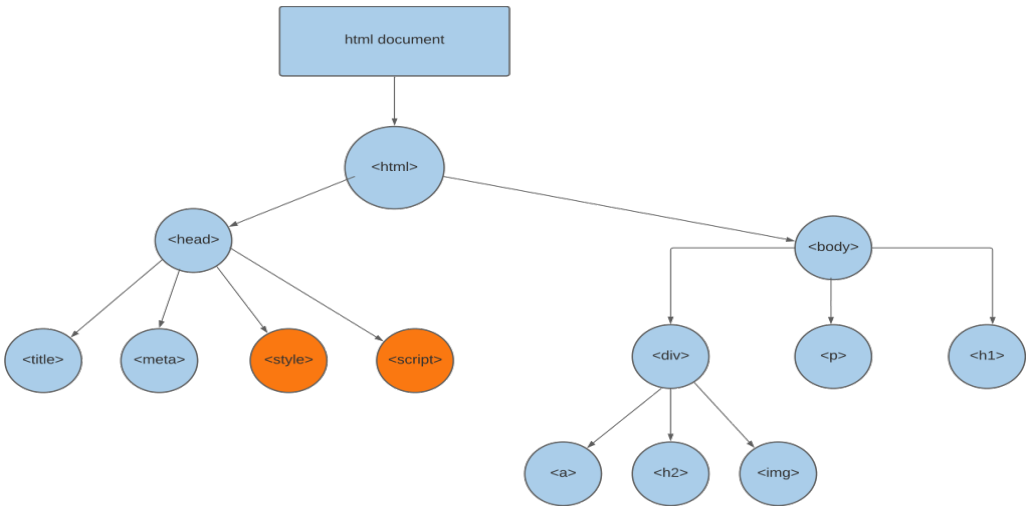

Figure 1: DOM Tree

rate and rapid translation of content across different languages has become vital. Automated content generation systems can now proficiently translate texts from one language to another, facilitating effective communication for individuals and businesses across language barriers.

To accomplish efficient automated text extraction from web content, we must address several complex challenges in extracting text from HTML pages. The complexities arise from the presence of nested elements, inline styles, and JavaScript-generated content within HTML pages. Effectively parsing and identifying the pertinent textual content while eliminating noise necessitates the application of sophisticated algorithms and techniques. Moreover, non-textual components such as images, advertisements, navigation menus, and footers may obstruct the extraction process. Addressing these obstacles demands the development of intelligent algorithms capable of distinguishing between relevant textual information and extraneous elements. Furthermore, the dynamic nature of HTML pages, subject to frequent updates, modifications, and formatting changes, requires adaptable extraction techniques to maintain strength and relevance over time.

In this paper, we present efficient techniques for extracting text from HTML pages to facilitate relation extraction. We confront the challenges posed by HTML's intricate structural elements and formatting tags, seeking to overcome their complexities and attain accurate extraction results.

2. Background and Related Work

2.1 HTML structure

HTML structure refers to the organization and hierarchy of elements within an HTML document. To visualize this structure, the Document Object Model (DOM) tree is often utilized. An example DOM tree is illustrated in Figure 1. Each HTML element is represented by a node in the tree, with the topmost node being the root element. For example, in a simple HTML document containing a `<body>` element with nested `<div>` and `<p>` elements, the `<body>` element would be the parent node, and the `<div>` and `<p>` elements would be its child nodes in the DOM tree representation. The DOM tree provides a logical structure that

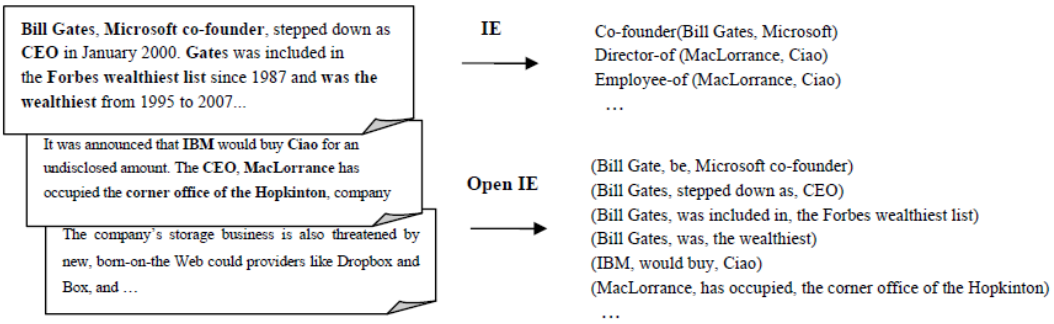

Figure 2: IE vs Open IE

Note: This figure is reproduced from Vo and Bagheri [5]

facilitates the manipulation and traversal of HTML elements by web browsers and applications, enabling efficient parsing and extraction of content from HTML pages. As part of the fundamental structure of HTML, there are typically two essential tags, which are `<head>` and `<body>`. The `<head>` tag is primarily used to contain metadata, reference external resources, as well as incorporating JavaScript `<script>` and CSS codes `<style>`. Lastly, the `<body>` tag represents the principal area within the HTML document where the main textual content is positioned.

2.2 Relation extraction

As NLP has emerged as a pivotal technology in the current development of AI chat products, relation extraction is one of the fundamental tasks in NLP that entails the identification and extraction of the semantic relationships or associations between entities mentioned in a given text. For example, in the sentence: "Bill Gates, Microsoft co-founder, stepped down as CEO in January 2000," there are three entities mentioned: "Bill Gates," "Microsoft co-founder," and "CEO". The relationship between these entities is that Bill Gates, in his role as the co-founder of Microsoft, stepped down from the position of CEO in January 2000. Relation extraction algorithms can be applied to analyze this sentence and identify the relationship "stepped down as" between the entities "Bill Gates" and "CEO". By utilizing these algorithms, a relation extraction model can automatically recognize and extract meaningful relationships between entities in textual data, allowing for improved understanding and organization of information. This plays an essential role in various natural language processing tasks, facilitating tasks such as information extraction and text summarization. Thus, the primary objective of relation extraction is to capture the interconnections between different entities and provide structured information about their interactions or attributes.

In the field of relation extraction, it is common to distinguish between traditional Information Extraction (IE) methods and Open IE approaches as discussed by Wu et al.[2]. In Figure 2, traditional IE methods typically rely on predefined schemas, templates, or ontologies to extract specific types of relations from text. These methods require prior knowledge and manual effort to design and define the extraction rules or patterns. On the other hand, Open IE approaches have gained popularity due to their flexibility and unsupervised nature. Unlike traditional methods, Open IE systems are not dependent on predefined schemas or

templates(Banko et al, 2007)[3]. They are designed to extract a wide range of relations without prior knowledge of the specific domain or relation types. Importantly, the primary objective of relation extraction is to facilitate the automated extraction of organized data from unstructured textual sources. This extracted information holds significant potential for various applications, comprising but not restricted to the construction of knowledge graphs, addressing queries, and integration of diverse datasets.

Practical challenges in relation extraction may arise due to the inherent variability present in textual data, the need for domain-specific knowledge, and the prevalence of noise and errors. Indeed, the variability of language poses a significant hurdle in relation extraction as texts can exhibit varied word orders, sentence structures, and writing styles, demanding adaptability in relation extraction algorithms. It is vital for these algorithms to recognize relationships amidst such variations and ensure consistent and accurate extraction results. This calls for the implementation of robust techniques capable of handling different linguistic patterns effectively. Moreover, domain-specific knowledge commonly presents a unique challenge as various domains possess their own specialized terminologies, jargon, and contextual nuances. To accurately extract relationships within these domains, it is necessary to incorporate domain-specific resources and expertise. This can be achieved through the utilization of domain-specific lexicons, ontologies, or training on corpora specific to the domain. Augmenting the algorithm's performance can be accomplished by accessing domain experts or specialized datasets, which facilitate a more effective capture of domain-specific relationships.

Lastly, the presence of noise and errors in textual data poses a challenge to relation extraction. Documents may contain OCR (Optical Character Recognition) errors, typographical mistakes, grammatical errors, or incomplete information. Preprocessing steps, which are spell-checking, grammar correction and data cleaning can possibly assist in mitigating noise-related issues and improving the quality of extracted relationships. On top of that, linguistic irregularities such as idiomatic expressions or sarcasm may further complicate the extraction process since they deviate from literal or straightforward language usage. Idiomatic expressions frequently convey meanings that cannot be deduced by analyzing individual words, requiring a deeper understanding of cultural or contextual nuances.

2.3 Related work

A related study conducted by Nethra et al.[4], introduces an approach for web content extraction that involves converting an HTML web page into a DOM tree and extracting features from it. By using the DOM tree, they carefully consider HTML parsing and identify HTML tags to efficiently extract text from the pages. Their careful identification of HTML tags primarily serves the purpose of excluding noisy data such as links, advertisements, headers, and footers, during the text extraction from HTML pages. Through the application of their devised approach, the content extraction model achieved higher precision, recall, f-measure, and accuracy for web content extraction. Moreover, Duc-Thuan and Bagheri [5] present another relevant project that provides an overview of two generations of Open IE systems, highlighting their strengths, weaknesses, and application areas. The first generation focuses on constructing a general model based on unlexicalized features. This means that the generation does not heavily rely on specific words or lexical information. Instead, it focuses on features such as Part-of-Speech (POS)

tags and shallow tags. These tags are utilized as features to support the identification and extraction of relations (Mesquitaet al.,2013)[6]. In the second generation, efforts are dedicated to addressing issues related to incoherent and uninformative relations. The provided studies offer constructive insights within the realms of HTML parsing and relation extraction regarding the extraction of text from HTML pages for the purpose of relation extraction.

3. Our solution

3.1 HTML parsing

To ensure the accurate extraction of textual content from HTML pages, the first step is undertaken to perform HTML parsing. HTML parsing refers to the process of analyzing and interpreting HTML text content from it. Observeably, textual contents in HTML pages are typically encoded by the default encoding set in the server's configuration, hence our HTML parsing engine purposes to read and decode HTML text content by the method: `html_content = response.read().decode('utf-8')`. The `response.read()` method retrieves the HTML content from the response object, such as an HTML file or URL of the website. As HTML pages contain different character encodings, decoding is necessary to convert them into human-readable text. In this case, `decode('utf-8')` is utilized, leveraging the UTF-8 character encoding, which supports a wide range of characters from different languages and symbol sets.

3.2 Tag filtering

As all the text content on HTML pages is enclosed within HTML tags, the second step involves HTML tag filtering. This follows the initial process of converting HTML pages into a DOM tree where the `<head>` and `<body>` tags serve as the parent nodes while the remaining tags are nested within these parent nodes as child nodes. To extract meaningful text from HTML pages, two different approaches are employed, focusing on the `<head>` and `<body>` tags respectively. Recognizing the significance of understanding the usage and purpose of HTML tags, we conducted a targeted survey to explore their application and relevance to textual content with ten real websites. To facilitate this survey, we developed a custom program capable of extracting text from HTML pages while identifying the corresponding tags used. After carefully analyzing the data from the HTML pages of these ten websites, we summarized our findings, which are presented in Figure 3. The survey report includes information on seven HTML tags, their frequency (%), the extracted text from each tag, and an example of the text content. The term "frequency" in this context denotes the average occurrence of HTML tags containing textual content. As an example, suppose the `<button>` tag appears ten times on an HTML page, and out of those occurrences, only five of them collect textual content. In this scenario, the frequency would be 50.00%. This valuable data provides insights into how often HTML tags containing textual content appear across various websites. Then, the text after the frequency is the actual text extracted from the respective tags. Lastly, we compiled a list of example text contents and categorized the types of information based on the extracted text.

In accordance with the survey report, the approach within the `<head>` tag is to focus on extracting text content specifically from `<title>` and `<script>`. Since the `<head>` tag is primarily intended to contain the title of the page, links to external stylesheets, scripts, character encoding information, and other meta tags as noted by (Zhou al.,2013)[7], exploring only these

- `<title>`: 100.00%, text: Vrije Universiteit Amsterdam - Wikipedia
  ex) Title of the HTML page, Names, Introduction.
- `<style>` and `<iframe>`: 0.00%, these didn't contain any text
- `<svg>`: 43.05%, text: Menu, Chevron, Search, Pinterest, Twitter, YouTube
  ex) Name of the logo, Title or Message of the graphical elements
- `<form>`: 98.15%, text: move to sidebar, hide, email, Search for
  ex) Navigations, Text fields, Checkboxes, Links
- `<button>`: 73.10%, text: Search, Click, Quick links, Work with us
  ex) Submit buttons, Checkboxes, Radio buttons, Links
- `<footer>`: 97.50%, text: Get SocialContact, Copyright, CampusPrivacy
  ex) Contact details, Copyright notices, Links to related documents
- `<nav>`: 91.94%, text: Home, Search, Links, About, Contact
  ex) Navigation messages, Menus, Sections, Pages

Figure 3: Comprehensive Analysis of HTML Tag Usage: A Summary Report

two tags is sufficient. Additionally, the text content inside of `<head>` tag was exclusively found within the `<title>` tag, with a frequency of **100.00%**. The reason behind exploring the `<script>` tag is its potential to store text contents such as opening messages and headlines of text content even though such cases were not observed during the survey. Extracting text content from the `<script>` tag requires executing the enclosed JavaScript code. This additional step is necessary since the `<script>` tag often encompasses dynamic content generated by JavaScript, which may not be directly accessible as static text within the HTML structure. Our engine manages to execute JavaScript code in a headless Chrome browser using the Selenium WebDriver. Selenium is a popular automation testing framework primarily used for automating web browsers. This method allows to execute arbitrary JavaScript code directly in the browser, accessing and manipulating the Document Object Model of the webpage. This approach ensures that our engine effectively extracts text content from the `<script>` tag, leaving no valuable information overlooked during the extraction process. The `<script>` tag within `<body>` is handled by the same method.

The process of HTML tag selection within the `<body>` tag, on the other hand, is mainly tag removal. Given the diverse range of HTML tags that can be utilized within the `<body>` tag, it is vital to acknowledge the potential for a wide variety of tags to appear, and the structure can differ depending on each HTML page. Hence, our method aims to filter out irrelevant tags instead of selecting relevant tags. With the data in Figure 3, the `<style>` and `<iframe>` tags did not extract any text content due to their functionality, and the resulting frequency was **0.00%**. The `<svg>` tag was utilized to embed Scalable Vector Graphics (SVG). The tag mostly contained various

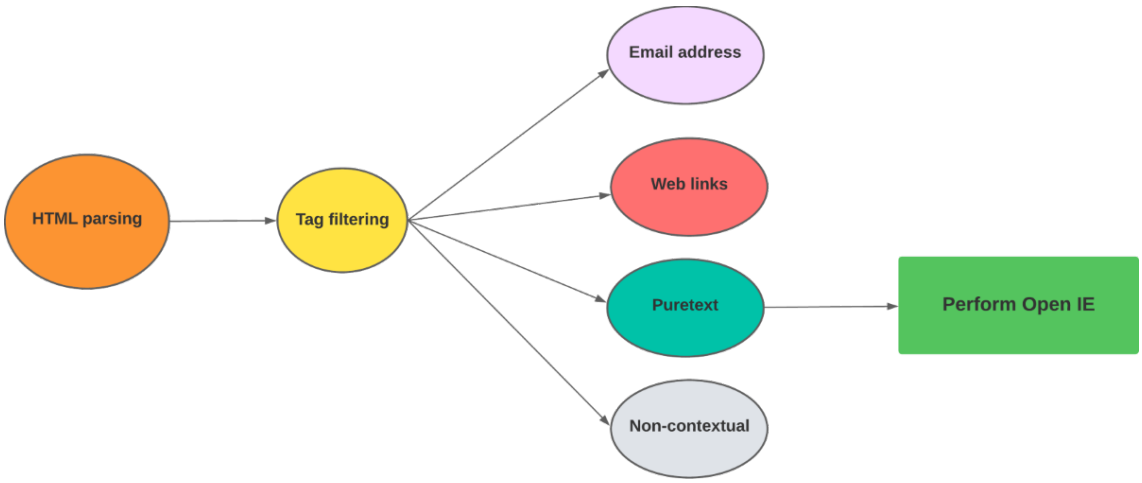

Figure 4: HTML parsing to Noise elimination

SVG attributes and the title of the graphics, which are considered irrelevant to the relation extraction process. The `<form>` and `<button>` tags typically involved input elements such as text fields, checkboxes, radio buttons, and submit buttons, which are reasonably considered unnecessary content for our purposes. Similarly, the `<footer>` and `<nav>` tags mainly held information such as contact details, copyright notices, links to related documents, and navigation messages. Since such information is unlikely to be relevant to our goal of extracting text, exploring these tags is considered unnecessary. Overall, inside of the `<body>` tag, we aim to filter out these seven tags, which are: [`<style>`, `<iframe>`, `<svg>`, `<form>`, `<button>`, `<footer>`, `<nav>`].

3.3 Noise elimination

Noise elimination primarily focuses on eliminating noise and prioritizing textual information. Due to the presence of various extraneous elements such as tags, external links, and navigational comments in HTML pages, which are unrelated to the substantive text content, noise elimination plays a pivotal role in eliminating these noises. By removing these irrelevant elements, noise elimination can ensure the extraction of only the pertinent text. To remove noise and unwanted characters, the Python built-in string method 'strip()' is utilized, enabling the removal of leading and trailing whitespace or specific characters from the strings extracted from HTML pages. As our engine traverses all the texts from HTML pages, these texts are defined as 'data', and applying 'data.strip()' generates a new string with the whitespace removed.

Regarding prioritizing the textual information, our method categorizes the extracted text into four labels, which are 'Email', 'Weblinks', 'Puretext', and 'Non-contextual', as depicted in Figure 4. It is noteworthy that HTML pages commonly incorporate interactive links, primarily facilitated by the utilization of the 'href' attribute. Noticeably, website URLs and email addresses, encapsulated within tags featuring this attribute, are prevalent examples of such implementation. It is important to highlight that these specific elements do not bear any inherent information relation within the textual input. As a consequence, the model refrains from considering such textual contents for the purpose of relation extraction, instead assigning them

(1) Email.pattern = re.compile(r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b')

(2) URL.pattern = re.compile(r'^(?:https?://|www\.)\S+\.[a-zA-Z]{2,}(?:[/?#]\S*)?$')

(3) Non-contextual is elif not = re.search(r'[a-zA-Z0-9]', data)

ex_1)"世界", "اَلْعَالَمُ", ".", "%" #text does not contain any any alphabet or numbers
ex_2) "3.3%" is not Non-contextual since it contains "3"

Figure 5: Predefined text formats and patterns for the labels

the labels 'Email' and 'Weblinks'. Regarding the label called 'Non-contextual', in case the extracted text does not include any alphanumeric characters or numbers, it is categorized as 'Non-contextual'. The example text input is a text of symbols or special characters or punctuation marks or foreign languages (Chinese, Japanese, etc.) that do not use any alphabet or numbers.

Our engine performs such a classification system based on predefined text formats and patterns, which are listed in Figure 5. Furthermore, the last label defined as 'Puretext' is responsible for performing relation extraction among the text contents. After the completion of noise elimination and the establishment of three labels, the remaining text contents are categorized as 'Puretext' under this specific label. This approach enables our system to prioritize and give the utmost importance to textual information. Aligned with our experimental objectives of conducting relation extraction, our engine can concentrate exclusively on examining the text contents classified as 'Puretext' in collaboration with Open IE, which will be discussed in the next section.

3.4 Relation extraction model

The relation extraction model utilized in this study is based on a Python NLP library provided by the StanfordCoreNLP framework. To extract relations from text, we decided to utilize the 'openie' annotator within the framework. This annotator combines rule-based and statistical techniques to identify and extract relations between entities mentioned in the text. Its primary focus is on extracting subject-verb-object triples, which represent the relationships between entities. Regarding its performance, the StanfordCoreNLP framework, as presented in an Open IE project by Xu et al.[8], achieved a decently high precision, ranging from **0.73** to **0.79**, under various experimental conditions. The reason why we selected the Open IE model is that Open IE models often prioritize high recall, aiming to extract as many relations as possible from the input text. In other words, due to the high recall nature of Open IE models, failure to appropriately filter out noise and irrelevant text information can significantly impair performance. Therefore, the model can effectively illustrate the importance of noise elimination or filtering, which is the prime challenge of our text extraction model.

To understand the characteristics of the model and demonstrate the workflow, we provided an example of text input with its corresponding result in Figure 6. When the extracted text contains two sentences with a period ('.') separating them, the Open IE model extracts three sets of triples from

<This Open IE is performed by StanfordCoreNLP>

"Text": "The Vrije Universiteit Amsterdam is being founded in 1880. The VU Amsterdam is one of two large, publicly funded research universities in the city.",

<Result>
1. Subject: Vrije Universiteit Amsterdam, Relation: is, Object: is founded in 1880
2. Subject: two large funded research universities, Relation: is in, Object: city
3. Subject: VU Amsterdam, Relation: is one of, Object: two large publicly funded research universities in city

Note: If remove the period "." between the sentences above, only one set is extracted

<Result>
1. Subject: two large funded research universities, Relation: is in, Object: city

Figure 6: Open IE performed by StandfordCoreNLP

the input. However, if the period is missing between the two sentences, the model only extracts one set of triples. The period can be interchanged with either ('?') or ('!') whereas the presence of other punctuation marks or noise can disrupt the sentence detection of the Open IE model. This observation highlights the impact of sentence format and noise on the results of Open IE. This sentence format adjustment is extensively employed to compile the expected subject-verb-object triples into datasets, thereby assisting the relation extraction model in detecting sentences within a text input.

4. Experimental Setup

In order to conduct experiments to evaluate the performance of our text extraction model, we have selected ten real HTML-based websites: [10]-[19] (Web1-Web10) for analysis. The selected websites encompass a diverse range of sources, including official university websites, blogs, online articles, news platforms, as well as food and travel reviews. The text contents from these websites are extracted and compiled into datasets, which will be further elaborated upon in section 4.2.

4.1 Base model vs Our model

In this experiment, the evaluation focuses on three aspects, which are efficiency, runtime, and accuracy, aiming to evaluate our model from various perspectives. To establish a comparison, we created a base model that involves a straightforward approach, extracting all text directly from HTML pages without any of the three processes - HTML parsing, tag filtering, or noise elimination. This base model implements the Python built-in library 'BeautifulSoup' for text extraction. Thus, the extracted text from the base model carries all textual contents found in the HTML pages. With the base model and our proposed model, we will calculate the runtime from HTML text extraction to Open IE for both. This analysis enables us to determine whether our model effectively reduces the runtime across all our methods. In the context of efficiency, the objective is to evaluate the model's ability to reduce the overall size or length of the extracted text in comparison to the original HTML pages. Essentially, this measurement quantifies the model's efficiency in condensing or eliminating unnecessary or redundant information while preserving the relevant content. Moreover, accuracy plays

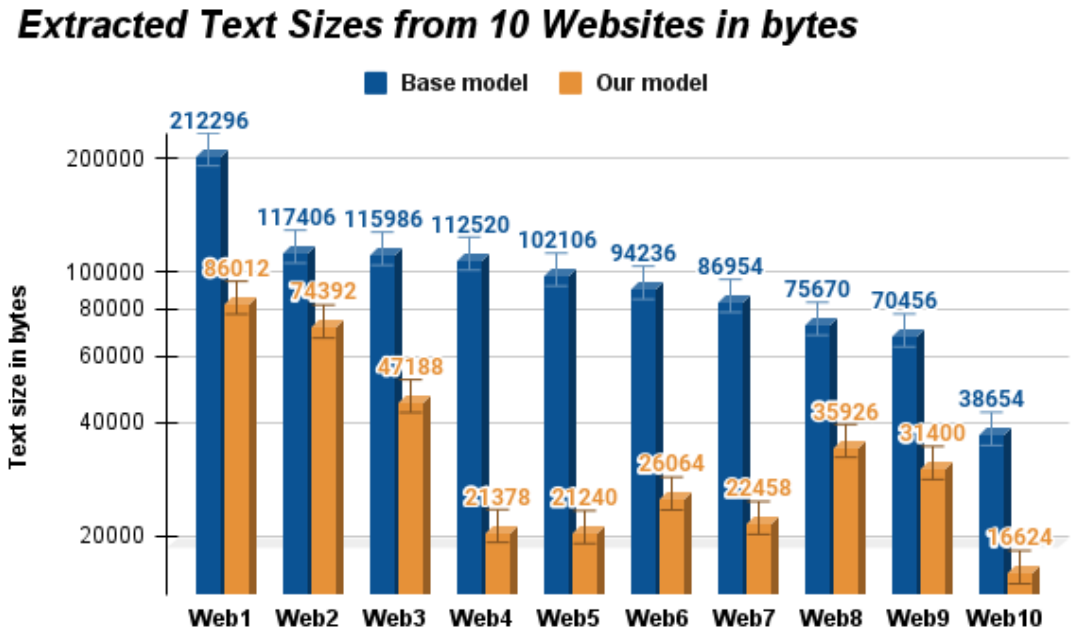

Figure 7: A Comparative Analysis of Extracted Text Reduction

an important role in this experiment as it serves as another crucial metric to assess the reliability and precision of the text extraction model. For evaluating the model's accuracy, we will employ three commonly used evaluation metrics: precision, recall, and F1 score. These metrics are widely utilized in information retrieval and natural language processing tasks to gauge the model's performance and effectiveness.

4.2 Dataset

The structure of the dataset essentially encompasses all the anticipated subject-verb-object triples extracted by the Open IE model. This dataset enables us to calculate the accuracy of our model by comparing its output with the expected results. However, it is vital to note that the text format and punctuation marks in the input text can influence the results as discussed in section 3.4. Therefore, we have taken great care in extracting the expected results by manually dividing the whole text into smaller sections and adjusting the text format. For example, we have added a period '.' after each title and sentence, which can help to mitigate the impact of noise and sentence detection errors. This approach ensures reasonably reliable relation extraction, taking into consideration of sentence detection and noise removal (Wu and Weld, 2010)[9]. Since we do not aim to optimize the Open IE model, the gold standard with the model is to ensure that it reads each sentence, by manually adding a period after each sentence.

5. Evaluation

5.1 Efficiency

Efficiency in this context refers to the measure of how effectively our text extraction using HTML parsing, noise elimination, and tag filtering, reduces the size of the ex-

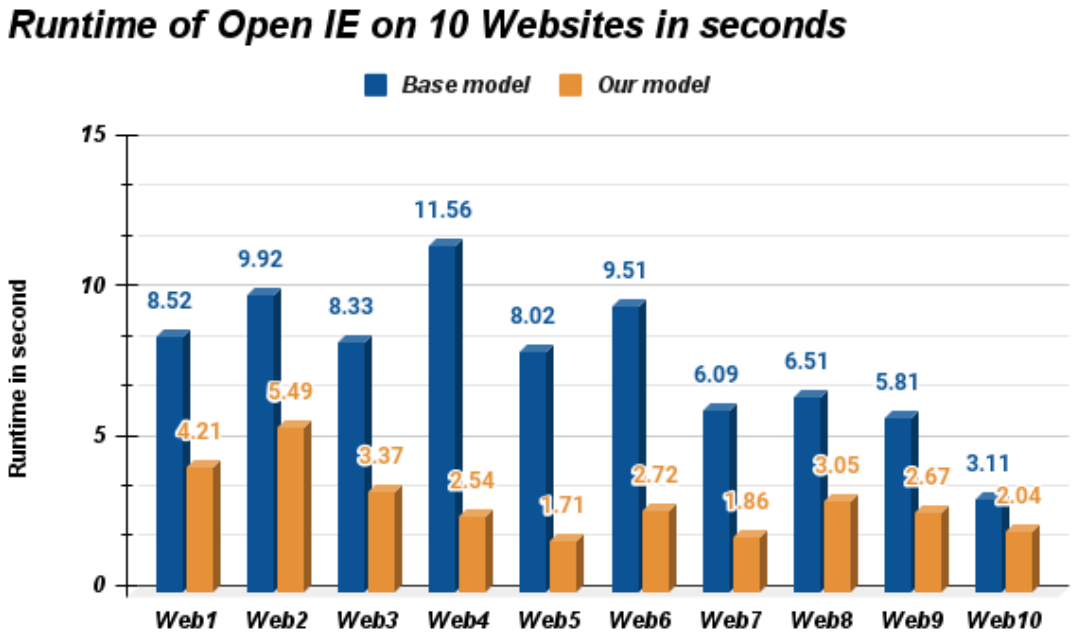

Figure 8: A Comparative Analysis of Runtime Reduction

tracted text. The data presented in Figure 7 demonstrates that our model consistently outperforms the base model across all ten websites. The results indicate significant reductions in the numerical values obtained by our model compared to the base model. Notably, Web4 and Web5 experienced the highest reductions, with **80.97%** and **79.14%** respectively. This suggests that our model significantly outperformed the base model on these websites, resulting in a substantial improvement in the specified metric. On the other hand, Web2 and Web8 demonstrated the smallest reductions in performance metrics after applying our model. The data for Web2 gained a reduction of **36.58%** while the result for Web8 experienced a slightly larger reduction of **52.54%**. On average, across all the ten websites, our model achieved a reduction of approximately **63,264 bytes**. This value represents the average improvement in text size reduction when using our model compared to the base model across all websites. Consequently, the average percentage reduction among all ten websites is approximately **63%**. This value signifies the average improvement achieved across all websites, expressed as a percentage reduction in the specified metric. Upon observing the analysis of text reduction, it becomes evident that the extent of reduction achieved by our model can vary significantly depending on several factors. One particularly influential factor revolves around how extensively a website relies on JavaScript code and the prevalence of dynamic content within its pages. These elements can affect the degree of reduction our model can accomplish.

5.2 Runtime

As presented in Figure 8, the data analysis compares the runtime reductions of ten websites using both the base model and an optimized version. Notably, our model

Avg-Accuracy	Precision	Recall	F1
Base model	0.60	0.86	0.77
Our model	0.84	0.87	0.88

Table 1: Evaluation metrics with 10 real-website-based datasets (Avg)

with Web5 achieved the most remarkable improvement, experiencing a reduction of **6.31** seconds (**78.68%** decrease), marking it as the website with the largest reduction. Following closely, the result with Web4 demonstrated the second-largest reduction where the runtime decreased by **9.02** seconds (**77.99%** decrease). Conversely, the data with Web10 achieved the smallest reduction, with a runtime reduction of **1.07** (**34.37%** decrease). Similarly, the reduction for Web1 displayed the second-smallest reduction, with the runtime reduced by **4.31** seconds (**50.59%** decrease). On average, the optimized model improved runtime efficiency by **4.29** seconds (**55.27%** decrease) across all websites. These findings emphasize the effectiveness of the optimization methods in enhancing our model's performance. In relation to the results of text reduction, it is notable that the largest and second-largest reductions were both achieved on Web4 and Web5, respectively. However, the smallest reductions were observed on different websites. This suggests that reducing the amount of text does not consistently lead to a proportional reduction in runtime. Instead, the impact on runtime heavily depends on the nature of the text the model is processing. Specifically, text that includes interactive features and dynamic functionality, such as script codes, tends to consume more runtime compared to simple text within HTML tags.

5.3 Accuracy

It is vital to keep in mind that simply reducing runtime and improving efficiency does not guarantee a successful model. If these improvements come at the expense of lower accuracy, the model may not be considered reliable. This is because the model may remove text contents without considering the importance of the extracted information, which goes against the primary goal of relation extraction.

We calculate the accuracy of these models by applying three evaluation metrics: Precision, Recall, and F1 score. The evaluation values for these metrics are calculated using the following formulas:

(1) TP = triples correctly extracted
(2) FP = triples wrongly extracted
(3) FN = triples failed to extract

In evaluating the extracted triples, our assessment considers a subject-verb-object triple as correct only if all the subject, verb, and object match the expected triples. In Table 1, the reported precision, recall, and F1-score are the average values obtained by evaluating both the base model and our model on 10 different datasets. As a result, our model demonstrated a noteworthy advancement over the base model across all three performance metrics. Notably, there is a remarkable **39.09%** increase in average precision, indicating a heightened level of accuracy in correctly identifying positive instances. Furthermore, there is a slight yet statistically significant **0.96%** improvement in average recall, indicating the model's enhanced ability to capture a greater amount of relevant information. However, the most notable progress is observed in the average F1-score, which exhibits an impressive

surge of **13.90%**. This expresses a decent enhancement in accuracy compared to the base model.

By achieving higher precision, recall, and F1 score, as proven by our model's results, we can confidently assert that our model has not only improved efficiency but also enhanced the accuracy of relation extraction. This improvement ensures that the extracted information is more reliable and aligns with the primary goal of relation extraction.

6. Conclusion

In conclusion, we presented our efficient text extraction techniques utilizing HTML parsing and noise elimination for advanced relation extraction. Our HTML parsing approach primarily focuses on decoding HTML text contents and tag filtering to remove unnecessary or redundant information while preserving the relevant content securely. The noise elimination stage involves stripping the extracted text and prioritizing textual information by categorizing it into four labels. After centering the target text contents, we employed Open IE using the Python NLP library provided by StanfordCoreNLP as our chosen model for relation extraction. In evaluating our proposed model, we conducted experiments focusing on runtime, efficiency, and accuracy as evaluation aspects to assess its performance. As a consequence, our model successfully reduced the size of extracted text contents from ten real HTML-based websites while minimizing the runtime. Additionally, our model achieved higher precision, recall, and F1 score compared to the base model with all the ten HTML-based datasets, demonstrating improved accuracy. Therefore, we can conclude that our proposed model not only increases efficiency but also improve the accuracy of relation extraction, thereby ensuring the extraction of more reliable information that aligns efficiently with the primary goal of relation extraction.

In future research, although our HTML parsing and tag filtering greatly aid in efficient automated text extraction, further optimization of noise elimination can be attained by incorporating additional predefined text patterns and rules. Our automated text extraction engine continues to gather secondary information. Another area for improvement may lie in identifying sentence formats for relation extraction. By incorporating Sentence Boundary Detection (SBD) capabilities into the program, the Open IE system can be optimized further. Achieving this goal would involve leveraging existing SBD algorithms or training machine learning models on annotated sentence boundary datasets. Furthermore, enhancing the Open IE model to account for the nuances of sentence structure assumingly holds the potential to amplify the performance to a greater extent.

Acknowledgements

I would like to express my sincere gratitude to my esteemed professor, Jacopo Urbani, for their exceptional guidance, invaluable insights, and unwavering support throughout the entire duration of this bachelor thesis project.

Author

Author: Koki Hirose, Vrije Universiteit Amsterdam, Amsterdam, The Netherlands.
Email: k.hirose@student.vu.nl
Project: https://github.com/Hikouki0408

References

[1] Deepak Garg and Deepika Sharma, CSED, Thapar University, Patiala, India, "Informa-

tion Retrieval on the Web and its Evaluation", 2012.

[2] Ruidong Wu, Yuan Yao, Xu Han, Ruobing Xie, Zhiyuan Liu, Fen Lin, Leyu Lin, Maosong Sun, "Open Relation Extraction: Relational Knowledge Transfer from Supervised Data to Unsupervised Data", 2019.

[3] Michele Banko, Michael J Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni, TuringCenter Department of Computer Science and Engineering University of Washington, "Open Information Extraction from the Web", 2007.

[4] K. Nethra, J. Anitha, and G. Thilagavathi, "WEB CONTENT EXTRACTION USING HYBRID APPROACH", 2014.

[5] Duc-Thuan Vo and Ebrahim Bagheri, "Open Information Extraction", 2016.

[6] Filipe Mesquita, Jordan Schmidek, and Denilson Barbosa, Department of Computing Science, University of Alberta, Canada, "Effectiveness and Efficiency of Open Relation Extraction", 2013.

[7] Ziyan Zhou and Muntasir Mashuq, "Web Content Extraction Through Machine Learning", 2013.

[8] Ying Xu, Mi-Young Kim, Kevin Quinn, Randy Goebel, and Denilson Barbosa, Department of Computing Science, University of Alberta, "Open Information Extraction with Tree Kernels", 2013.

[9] Fei Wu and Daniel S. Weld, University of Washington Seattle, WA, USA, "Open Information Extraction using Wikipedia", 2010.

[10] https://theluxurytravelexpert.com/2020/12/14/best-hotels-in-the-worldl

[11] https://en.wikipedia.org/wiki/Vrije_Universiteit_Amsterdam

[12] https://research.ibm.com/blog/utility-toward-useful-quantum

[13] https://www.hotcars.com/upcoming-cars-worth-waiting-for/#2023-fisker-ocean

[14] https://www.tudelftcampus.nl/time-to-shake-up-the-pile-driving-industry

[15] https://hackr.io/blog/what-is-programming

[16] https://www.engadget.com/best-android-phone-130030805.html?guccounter=1&guce_referrer=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8&guce_referrer_sig=AQAAAMJRC35y42RkEpGFN410RsxpbKvMCO1

[17] https://www.amsterdamfoodie.nl/amsterdam-food-guide/indonesian-restaurants-in-amsterdam-rijsttafel

[18] https://stackoverflow.blog/2023/05/31/ceo-update-paving-the-road-forward-with-ai-and-community-at-the-center

[19] https://www.euronews.com/travel/2023/02/27/long-queues-and-scams-will-the-new-eu-entry-system-cause-border-chaos