# Optimized Extraction of Records from the Web Using Signal Processing and Machine Learning

**Roberto Panerai Velloso**[1], **Carina F. Dorneles**[1]

[1]Universidade Federal de Santa Catarina - UFSC
Florianópolis (SC)

rvelloso@gmail.com, carina.dorneles@ufsc.br

***Abstract.*** *In this paper, we present an optimization of our previous record extraction approach from web pages. The proposed optimization improves the upper bound from $O(nlogn)$ to $O(n)$ while maintaining the same qualitative results as before (i.e., no loss in efficacy). We have achieved the following results: a 47% improvement in runtime efficiency when compared to our previous work and 95% f-score (same as our previous work).*

***Resumo.*** *É apresentado, neste artigo, uma otimização realizada em cima de um trabalho anterior, dos mesmos autores, que propõe uma técnica de extração de registros estruturados de páginas web. A otimização proposta melhora a complexidade de tempo de $O(nlogn)$ para $O(n)$ enquanto mantém os mesmos resultados qualitativos apresentados anteriormente (i.e., sem perda de eficácia). Os seguintes resultados foram obtidos: melhoria de 47% no tempo de execução quando comparado com o trabalho anterior e f-score de 95% (mesmo obtido no trabalho anterior).*

## 1. Introduction

Automatically extracting semi-structured data from the web is a very important and difficult task. Once extracted, this structured data can be used in a number of applications (e.g., price comparison, semantic keyword search, metasearch, etc.). The difficulty in extracting this data lies in the fact that it is published in very diverse and loosely structured formats in various domains, it is intended for a human audience, hence it is not trivial to **automatically** extract it.

There is quite a lot of research on the subject. Some proposals try to find similar neighboring subtrees in the document [Liu et al. 2003]; others search for patterns in visual information [Simon and Lausen 2005]. There are also approaches that target more specific data sources like tabular data formatted with specific HTML tags[Cafarella et al. 2008]. It is a consensus in the scientific community that the extraction of structured data from the web is still an open problem[Schulz et al. 2016].

In our previous work[Velloso and Dorneles 2017, Velloso and Dorneles 2019], we have outlined an approach that extracted structured data from web pages using signal processing and machine learning. Here, we go a step further and improve its time complexity. Whenever dealing with web scale data, efficiency is crucial and any runtime improvement is relevant, but it is especially relevant when it is an asymptotic improvement.

From an engineering point of view, whenever we propose a solution to a problem, we should care a great deal about its feasibility. In computer science, feasibility often means space and time complexity. Hence, we focus on optimizing our extraction approach. Our previous work was limited by the time complexity ($O(nlogn)$) of the fast Fourier

transform. We use the Fourier transform to detect the size and the number of record within a structured region of a web page. That seemed to be a hard limit to break, but we have managed to overcome it by avoiding unnecessary computation and circumventing other issues that emerged as consequence of this optimization (more details on that in Section 4).

The improvement and contribution proposed in this paper is the following: **asymptotic reduction of the time complexity** necessary to detect the records within a structured region and hence **reducing overall time complexity**. Our previous approach already outmatched the state-of-the-art with respect to runtime performance and it had a $O(nlogn)$ time complexity that we now managed to lower to $O(n)$ by avoiding unnecessary computation when inspecting a data region's power spectrum. The result of this optimization is a 47% gain in runtime keeping the same f-score as our previous approach (around 95%).

The rest of this paper is organized as follows: Section 2 presents the needed background and briefly describe our extraction approach; Section 3 describes some of the relevant related work in this research field; Section 4 details the optimization proposed; Section 5 compares and discusses the results; Section 6 concludes the paper.

## 2. Background

In this section, we present a brief description of our previous work. This description is needed for completeness and to understand what we are proposing to optimize in this paper. Our previous work tackles the problem of automatically extracting data records from web pages with the use of signal processing techniques.

The whole process is based on the observation that the structured content of a document must have at least some structural consistency, even in the presence of noise, so that it looks somewhat organized to the reader. In other words, the records are laid out so that it is obvious to the human reader that they are related to each other and they refer to same subject (or entity). Such an organization necessarily reflects on the structure of the DOM tree and its style definition and, consequently, on its tag path sequence (TPS) representation.

Figure 1 depicts the entire process of record extraction proposed in our previous work. First, the HTML document is parsed into a DOM tree using *libtidy* (a W3C endorsed HTML cleanup and standardization tool). After this, Step 1 converts the DOM tree into a TPS; Step 2 divides the TPS into several structured regions; Step 3 identifies the record boundaries within each region; Step 4 classifies the regions as content or noise and; Step 5 aligns the records in tabular form. Step 3 (in dark gray color) is the object of the optimization described in this paper.

In the first step, the web page is converted from a DOM tree into a signal (i.e., a sequence, or string, of integer values), this conversion process retains structure information from the original document in the form of cycles in the signal; this signal is then segmented into cyclic subsequences (i.e., subsequences that contain structure); after that, each subsequence's power spectrum is analyzed to detect the main **period** and **frequency** present in it, these two values represent average **record size** and **record count**, respectively; after record detection we classify each region as content or noise; in the last step the detected records are aligned.

To allow the use of signal processing techniques, the document is converted

to a sequence representation as illustrated in Figure 2. Figure 3 shows an example of a real web page converted to sequence and its main content region on display. More details on that are beyond the scope of this paper and we refer the reader to [Velloso and Dorneles 2017, Velloso and Dorneles 2019].

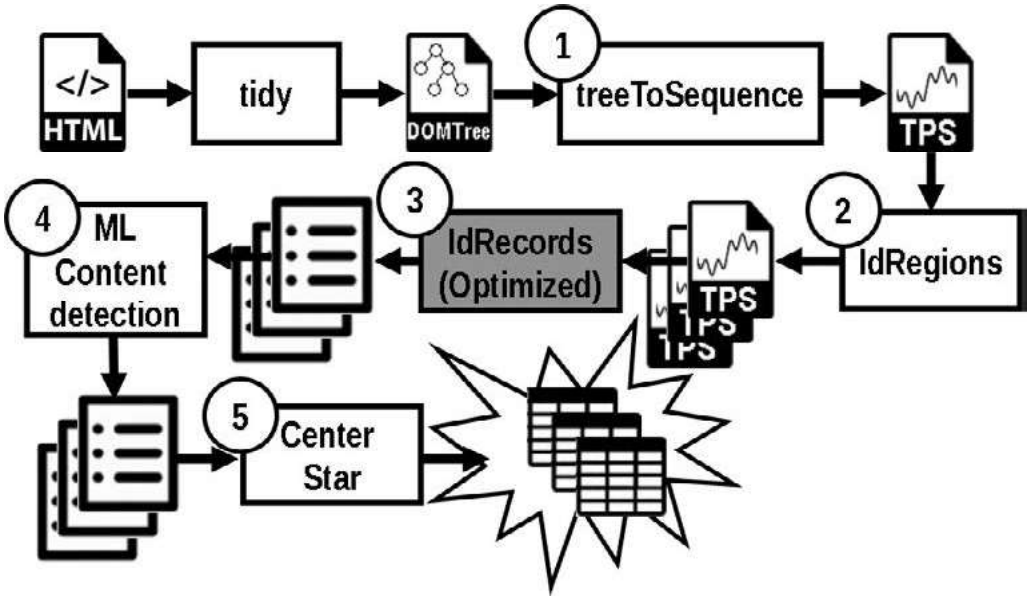


**Figure 1. Extraction system overall diagram.**

## 3. Related Work

We can find relevant works in the research field of web structured extraction dating as far back as the 90's. Despite being researched for quite some time now, the problem of extracting records from web pages is still open. There are many surveys that agree on this point (e.g., [Sleiman and Corchuelo 2012, Ferrara et al. 2014, Schulz et al. 2016, Varlamov and Turdakov 2016, Roldán et al. 2019]).

In [Liu et al. 2003, Zhai and Liu 2005, Liu and Zhai 2005, Jindal and Liu 2010, Shi et al. 2015, Wai et al. 2017], the structured data is detected automatically using tree edit distance. The records are extracted, mainly, from result/list pages and they are assumed to be similar in structure and contiguous on the document.

In [Crescenzi et al. 2001, Kayed and Chang 2009, Arasu and Garcia-Molina 2003], the extraction is "site oriented". Several pages from the same template are required to train the extractor. The algorithm then detects what is template (i.e., invariant) and what is data (i.e., variant). These approaches can be used to extracted data from detail pages as well.

In [Guo et al. 2019] the redundancy in search result records is exploited to perform extraction in detail pages. Similar to what is done in "site oriented" approaches.

In [Cai et al. 2003, Simon and Lausen 2005, Liu et al. 2009, Grigalis 2013], the document is fully rendered in a browser and the visual information generated in the process is then used to leverage the extraction. The rendering process can be quite computationally expensive and it is not clear whether this is feasible in a production environment or even if it is worth exchanging this computation for the results achieved.

In [Cafarella et al. 2008, Wang et al. 2012, Elmeleegy et al. 2011, Chu et al. 2015, Zhang et al. 2013, Qiu et al. 2015], the information is extracted

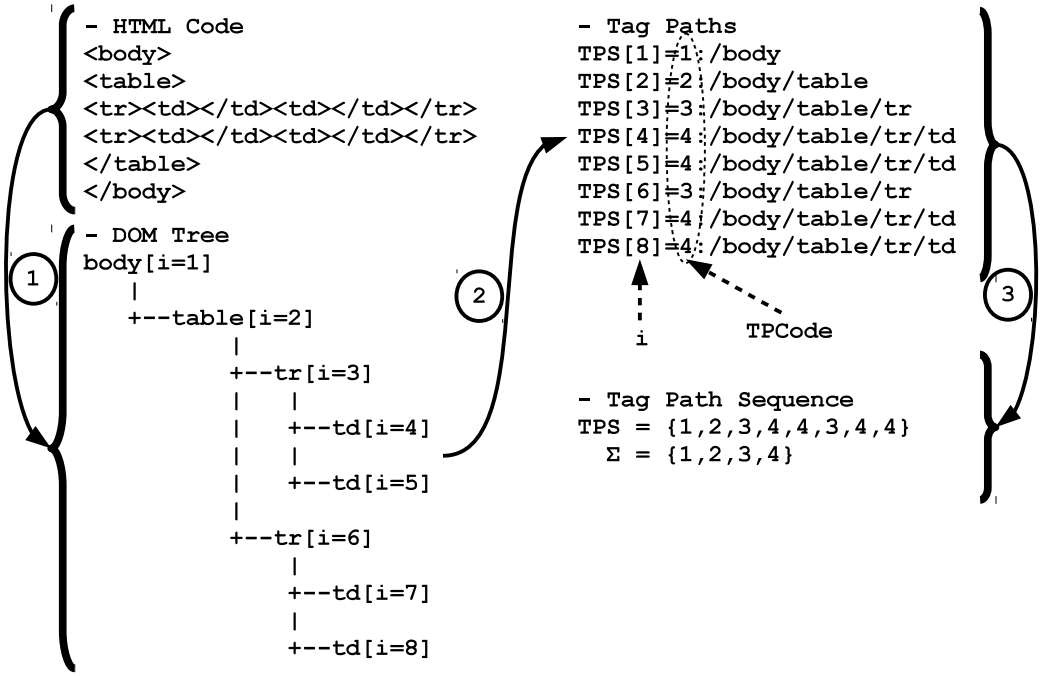


**Figure 2. tag path sequence construction.**

from very specific sources like <table>,<ol>,<ul> and specific page templates (e.g., top-k lists). The extraction quality is very good, but limited in range since we can not extrapolate these approaches to other types of structured data, even if the underlying structure is the same (e.g., a <table> extractor will not work at all in a table formatted with ).

In [Xie et al. 2012, Miao et al. 2009, Velloso and Dorneles 2013, Fang et al. 2018], a different document representation is used to analyze document structure: a sequence of tag paths. Converting the DOM tree to a sequence/string allows the use of different algorithms and at the same time retains most of the document structure information that we are interested in. In our work we use this same sequence representation to perform extraction.

Most works cited here either have superlinear time complexity or, when they neglected time complexity analysis, assuming they have linear time complexity, they still would have a very high constant factor (e.g., some approaches render the document in a browser engine). We are of the opinion that complexity matters, since it can render an approach to be impractical/unfeasible.

In our work we have strived to maintain some degree of generality (i.e., avoid *ad hoc* and *a priori* definitions and assumptions) and at the same time achieved good results by trading off recall for precision (our approach detects records with similar size and structure and tends to ignore records with disparate sizes). We have avoided rendering the document in a browser engine and kept the computational complexity within acceptable boundaries. In this paper we improved this aspect (computational efficiency) even further.

## 4. Optimization

The time complexity of our approach is dominated by step 3 in Figure 1 (record identification) which employs the FFT to detect the records in a structured region. The Fourier

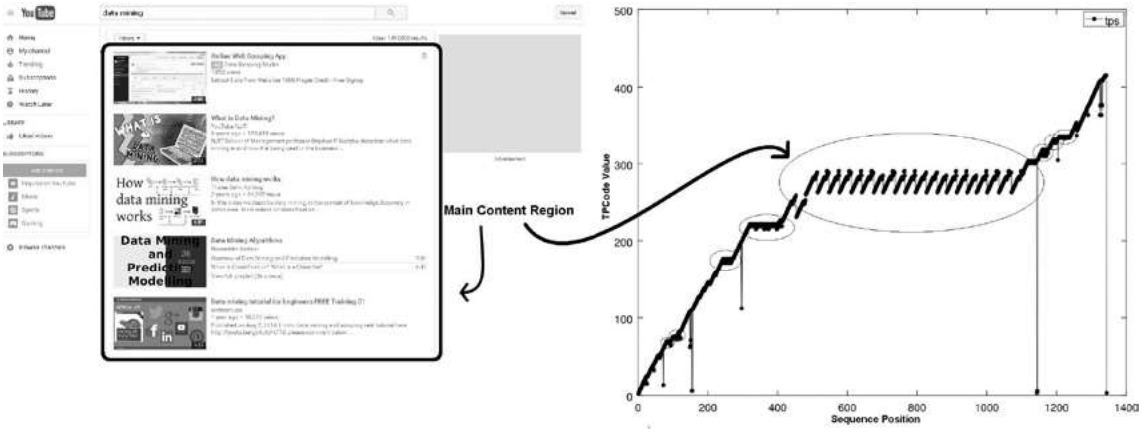


**Figure 3. How structured content appears in the sequence (structured regions encircled).**

transform allows us to do that by decomposing the input signal into its constituent frequencies, so we can analyze each component and decide if it is relevant or not. The FFT algorithm has $O(nlogn)$ time complexity and all other steps of our approach have linear time complexity, so the overall time complexity is $O(nlogn)$.

Although it is not known whether the Fourier Transform can be performed in less than $O(nlogn)$ time, in our particular situation we can improve the upper bound of our approach to $O(n)$ time. This is only possible because we search for specific frequencies in the region's PSD, so we can avoid unnecessary work (i.e., computing the entire PSD) and compute only the coefficients we are checking against. We can see this as a "PSD on demand", or a "lazy PSD". For example, in Figure 4 we detected some regularity in $value = 255$, it is evenly distributed along the x-axis, so we check its respective frequency only, there is no need to compute the entire PSD, only a few coefficients around frequency $20$ are needed.

Computing only the required coefficients gives rise to another problem: we need to know if that specific frequency is a peak in the PSD, so how can we compare one coefficient with the rest of the PSD without entirely computing it? We need a way to relativize/standardize the coefficients, because there is not much use in knowing only the raw value of one coefficient (or a few, for that matter). In our previous approach we had the full PSD at hand, so we were able to standardize the coefficients with respect to the PSD's mean and standard deviation. This standardization is what allowed us to decide when a frequency is relevant or not (i.e., if it represents a peak or not). For a better understanding, this situation is illustrated in Figure 5 where, if we have the full PSD at our disposal, we can easily see that frequency $20$ represents a peak, because its power is much higher than the rest of the PSD. On the other hand, if we only have the coefficients around frequency $20$ (lazy PSD), and we know nothing about the rest of the PSD, then we have nothing to compare against.

Fortunately there is a way to compare one coefficient against the entire PSD without computing it: **Parseval's Identity** (Equation 1). Informally, Parseval's Identity states that "the total power of a signal is equal to the power of its spectrum", so the sum of a squared signal is equal to the sum of its power spectrum density. That means we can sum our squared signal (in $O(n)$ time), divide the sum by the signal's length and we get

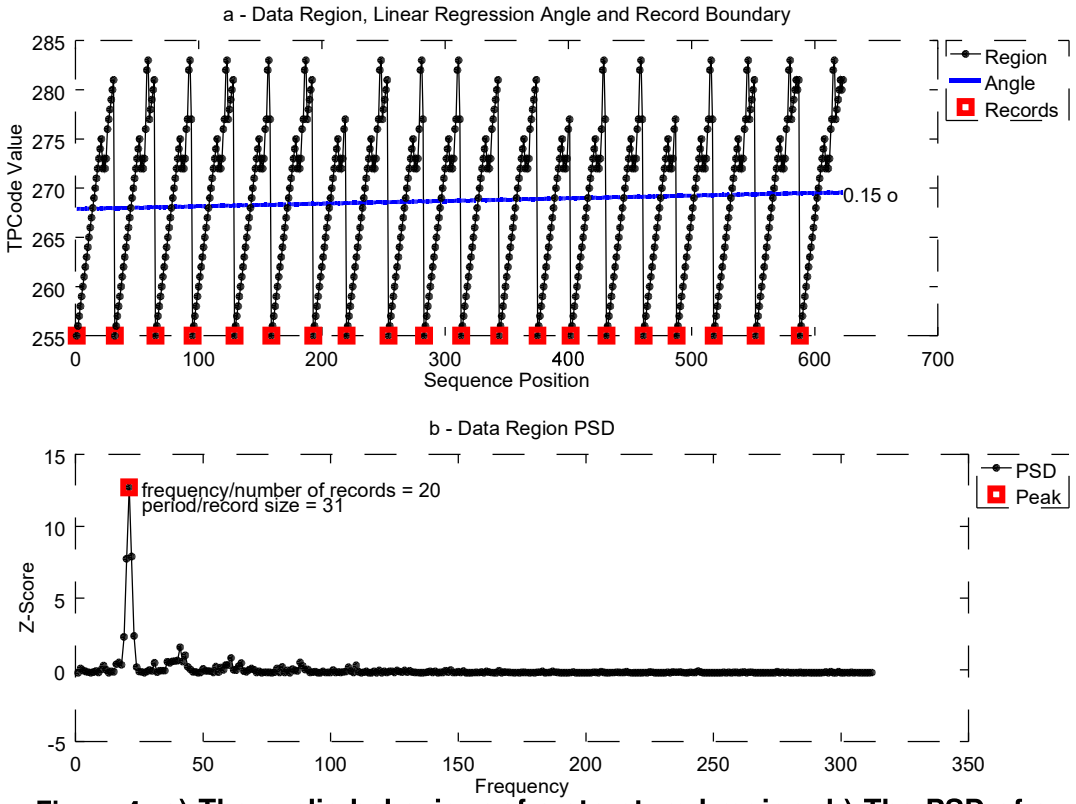


**Figure 4. a) The cyclic behaviour of a structured region; b) The PSD of a structured region.**

the PSD's average value (Equation 2). Knowing the average of the entire PSD we can now compare a coefficient against it and decide if it is relevant or not. In other words, we can standardize the coefficients using the average.

$$\sum |x[n]|^2 = \sum |X[f]|^2 \tag{1}$$

In Equation 1, $x[n]$ is the structured region (the signal) and $X[f]$ is the region's PSD.

$$E(PSD) = \frac{\sum |x[n]|^2}{N} \tag{2}$$

In Equation 2, $E(PSD)$ is the PSD's average value, $x[n]$ is the structured region and $N$ is the region's length.

Putting it all together, there is no more need to compute the entire PSD, only a few coefficients and the PSD's average value. The individual coefficients can be calculated in $O(n)$ time (using Equations 3 and 4) as well as the PSD's average (using Equation 2). With these modifications we have effectively lowered the time complexity of our approach from $O(nlogn)$ time to $O(n)$ time **as long as we keep the number of computed coefficients small** (and that is the case, as we show in Section 5 and in Figure 7). If the number of computed coefficients is allowed to grow proportional to input size, then the time complexity would degrade to $O(n^2)$.

Conceptually there is no difference between the two approaches (full PSD vs sin-

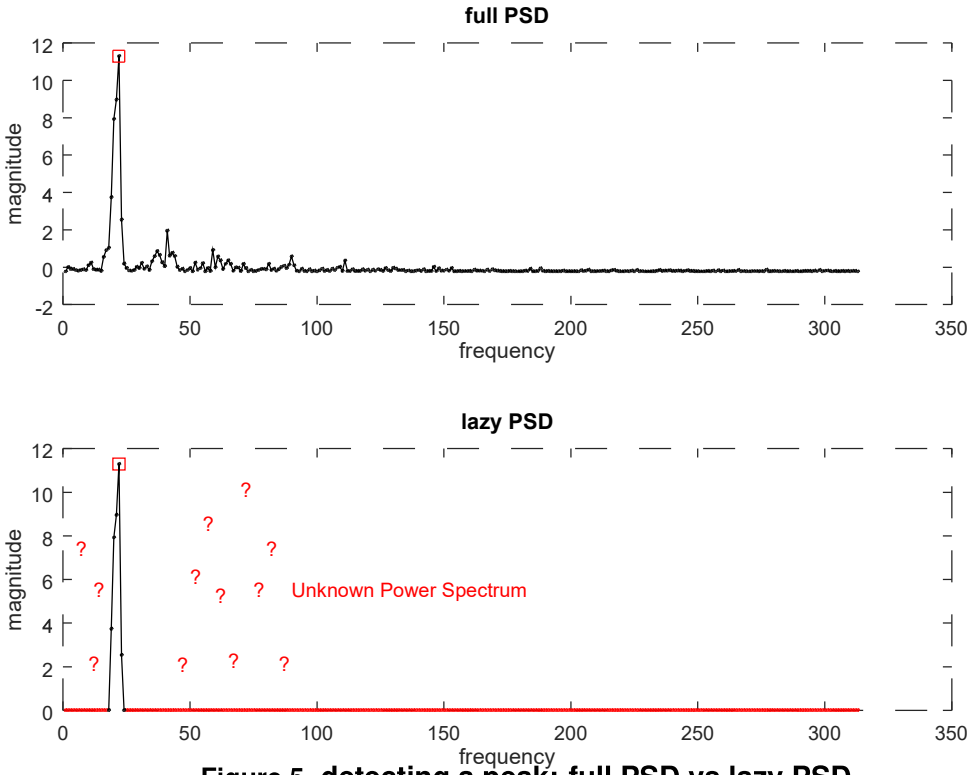


**Figure 5. detecting a peak: full PSD vs lazy PSD.**

gle coefficient), both methods calculate the exact same values for all coefficients of a given signal. In our record detection approach the only significant difference between the two methods occurs when we standardize the coefficients: when using the entire PSD we standardize with respect to mean and standard deviation whereas when using single coefficients we standardize using only the mean. We are doing this because in order to compute the standard deviation we need the full PSD, and our objective is precisely to avoid this computation. This difference in standardization seems to be not so significant, after all, because we were able to achieve the same qualitative results in record detection using both methods, only much faster when computing only a few number of coefficients.

To shield this modification from the rest of the implementation, avoiding introducing any kind of interference in runtime analysis, we have used the *strategy design pattern* to switch between both implementations (full and lazy PSD) while keeping the rest of the system identical for a fair comparison. The actual class diagram implemented in our system is depicted in Figure 6, showing the respective time complexity of each operation, depending on the strategy used. Full PSD construction takes $O(nlogn)$ time because that is where FFT is computed and looking up a coefficient is done in $O(1)$ since they are all precomputed; lazy PSD construction takes $O(n)$ because that is where we compute the signal mean, using Equation 2, and looking up a coefficient is done in $O(n)$, using Equations 3 and 4.

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-2\pi i k n / N} \tag{3}$$

$$PSD = |X_k|^2 \tag{4}$$

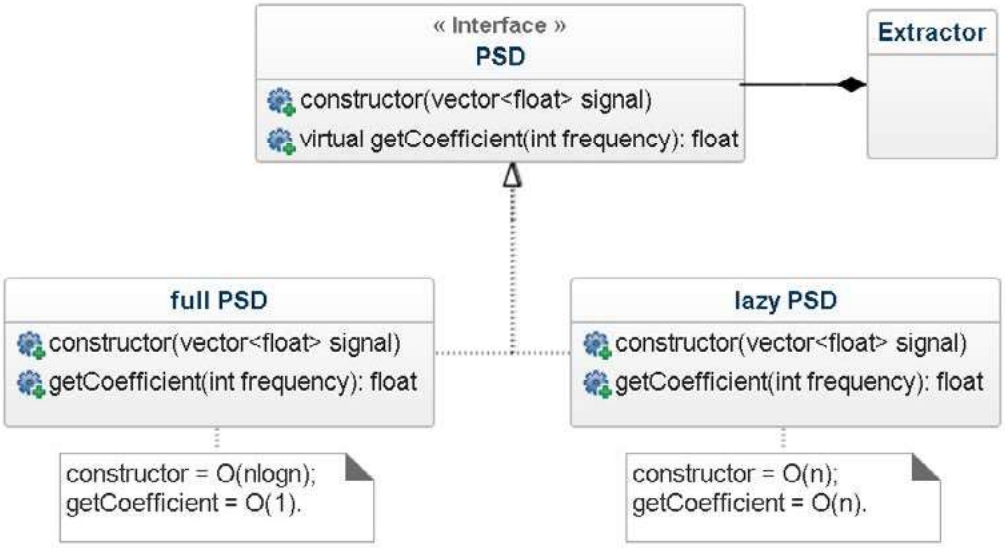


**Figure 6. Strategy class diagram and time complexity.**

As a bonus, the optimized version of our algorithm is actually easier to implement since we can just use the naive Fourier transform algorithm (i.e., a direct implementation of Equation 3) instead of the more complicated algorithm for fast Fourier transform.

Just out of curiosity, a similar approach (Goertzel's algorithm[Goertzel 1958]) is used for realtime DTMF tone detection in telephone lines using low end microcontrollers that can not perform FFT in realtime, but are capable of computing single coefficients in realtime. Similarly, in this application, there is a limited number of tones to be detected, and so there is no need to compute the entire transform, only the few coefficients that represent each tone.

## 5. Results

We have outlined in Section 4 a way of lowering the time complexity that is inherent to the FFT by means of lazy computation of PSD. Here we evaluate the results achieved by doing so.

We have decided to perform this experiment to evaluate the runtime behaviour of our approach in practical scenarios, not only in theory. We think this is necessary because the asymptotic behaviour depends on the number of coefficients computed during record detection. One way to demonstrate that this number is kept constant (i.e., that is does not grows proportional to input size) is by empirical evaluation.

Table 1 shows our dataset summary. It consists of 327 very diverse HTML documents, recently collected from various templates, domains and sites. There are 266 documents with both structured content and structured noise and 61 documents that contain only structured noise.

Figure 7 presents a comparison of both approaches: black color represents computing whole PSD using FFT in $O(nlogn)$ time (the dashed line is a regression and triangles are individual HTML documents); red color represents lazy PSD computation in $O(n)$ time.

**Table 1. Input dataset summary.**

| | | |
|---|---|---|
| # Content Regions | 254 | 47.65% |
| # Noise Regions | 279 | 52.35% |
| Total | 533 | 100% |
| # Structured documents | 266 | 81.35% |
| # Unstructured documents | 61 | 18.65% |
| Total | 327 | 100% |

We can see a $47,17\%$ improvement in runtime when using lazy PSD, compared to full PSD computation: full PSD takes 70.763s to process our 327 documents dataset and lazy PSD takes 37.384s. The linear regression is also adherent to the lazy PSD runtime ($R^2 = 89.04\%$), corroborating our claim of $O(n)$ runtime. If the number of computed coefficients were not constant but proportional to the input size, the actual time complexity would be $O(n^2)$ (i.e., $n$ coefficients computed in $O(n)$) which is higher than computing the full PSD.

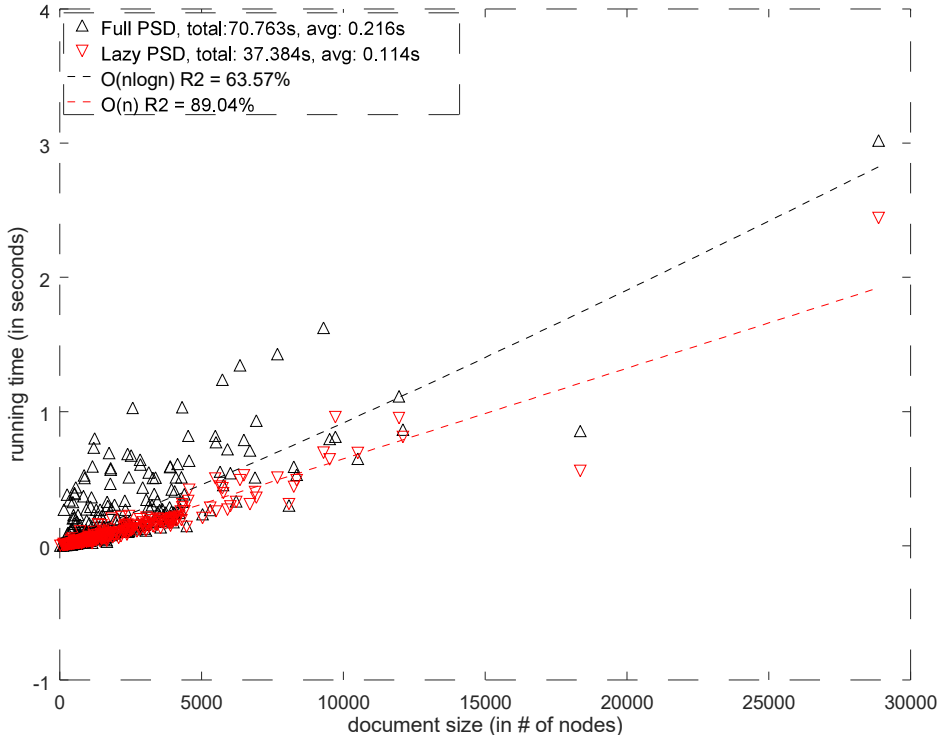


**Figure 7. Runtime improvement with lazy PSD.**

In Table 2, we reproduce the qualitative results obtained in citevelloso2019web using our extraction approach compared with other, state-of-the-art, approaches found in the literature. This results have not changed when we applied the optimization proposed in this paper, indicating that the change in standardization (using only the mean instead of mean and standard deviation), needed by our optimized algorithm, did not impact the quality of the results. In Table 2 we have two comparisons: **LR** is a logistic regression model trained to classify structured content in a controlled environment; **VOT** is a heterogeneous voting ensemble trained to classify structured content in an open environment.We consider an environment to be controlled if the input is guarantee to contain structured content

and, conversely, we consider an open environment to be one where we do not guarantee the input contains any structure whatsoever. In both cases we have outmatched the other approaches.

**Table 2. Result comparison with other approaches.**

| Algorithm | Prec. | Recall | F-Score | Acc. |
|---|---|---|---|---|
| TPC[Miao et al. 2009] | 90.40% | 93.10% | 91.73% | n/d |
| MDR[Liu et al. 2003] | 59.80% | 61.80% | 60.78% | n/d |
| Our models[Velloso and Dorneles 2019] | | | | |
| LR | **95.45**% | 95.45% | **95.45**% | 94.80% |
| VOT | **93.02**% | 90.91% | **91.95**% | 90.91% |

## 6. Conclusion

In this paper we have presented a considerable improvement over our previous work. The optimization proposed reduced the asymptotical upper bound from $O(nlogn)$ to $O(n)$ and on an empirical analysis it showed an improvement of more than 47% in runtime efficiency. We have achieved all this, however, without sacrificing quality, as shown in the results.

We do not believe to be possible to reduce the asymptotical upper bound even further, at least not without assuming something about the input (e.g., like binary search assumes the input is sorted), for a simple reason: in a general extraction algorithm we will have to analyze each node of the document at least once to decide if it should be extracted or not. By doing that we are already at $O(n)$. But we believe it is possible to reduce the associated constant factor by filtering parts of the document prior to the more heavy extraction processing.

## Referências

Arasu, A. and Garcia-Molina, H. (2003). Extracting structured data from web pages. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 337–348. ACM.

Cafarella, M. J., Halevy, A., Wang, D. Z., Wu, E., and Zhang, Y. (2008). Webtables: exploring the power of tables on the web. *Proceedings of the VLDB Endowment*, 1(1):538–549.

Cai, D., Yu, S., Wen, J.-R., and Ma, W.-Y. (2003). Extracting content structure for web pages based on visual representation. In *Asia-Pacific Web Conference*, pages 406–417. Springer.

Chu, X., He, Y., Chakrabarti, K., and Ganjam, K. (2015). Tegra: Table extraction by global record alignment. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 1713–1728. ACM.

Crescenzi, V., Mecca, G., Merialdo, P., et al. (2001). Roadrunner: Towards automatic data extraction from large web sites. In *VLDB*, volume 1, pages 109–118.

Elmeleegy, H., Madhavan, J., and Halevy, A. (2011). Harvesting relational tables from lists on the web. *The VLDB Journal - The International Journal on Very Large Data Bases*, 20(2):209–226.

Fang, Y., Xie, X., Zhang, X., Cheng, R., and Zhang, Z. (2018). Stem: a suffix tree-based method for web data records extraction. *Knowledge and Information Systems*, 55(2):305–331.

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

Goertzel, G. (1958). An algorithm for the evaluation of finite trigonometric series. *The American Mathematical Monthly*, 65(1):34–35.

Grigalis, T. (2013). Towards web-scale structured web data extraction. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 753–758. ACM.

Guo, J., Crescenzi, V., Furche, T., Grasso, G., and Gottlob, G. (2019). Red: Redundancy-driven data extraction from result pages? In *The World Wide Web Conference*, pages 605–615. ACM.

Jindal, N. and Liu, B. (2010). A generalized tree matching algorithm considering nested lists for web data extraction. In *Proceedings of the 2010 SIAM International Conference on Data Mining*, pages 930–941. SIAM.

Kayed, M. and Chang, C.-H. (2009). Fivatech: Page-level web data extraction from template pages. *IEEE transactions on knowledge and data engineering*, 22(2):249–263.

Liu, B., Grossman, R., and Zhai, Y. (2003). Mining data records in web pages. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 601–606. ACM.

Liu, B. and Zhai, Y. (2005). Net–a system for extracting web data from flat and nested data records. In *International Conference on Web Information Systems Engineering*, pages 487–495. Springer.

Liu, W., Meng, X., and Meng, W. (2009). Vide: A vision-based approach for deep web data extraction. *IEEE Transactions on Knowledge and Data Engineering*, 22(3):447–460.

Miao, G., Tatemura, J., Hsiung, W.-P., Sawires, A., and Moser, L. E. (2009). Extracting data records from the web using tag path clustering. In *Proceedings of the 18th international conference on World wide web*, pages 981–990. ACM.

Qiu, D., Barbosa, L., Dong, X. L., Shen, Y., and Srivastava, D. (2015). Dexter: large-scale discovery and extraction of product specifications on the web. *Proceedings of the VLDB Endowment*, 8(13):2194–2205.

Roldán, J. C., Jiménez, P., and Corchuelo, R. (2019). On extracting data from tables that are encoded using html. *Knowledge-Based Systems*.

Schulz, A., Lässig, J., and Gaedke, M. (2016). Practical web data extraction: are we there yet?-a short survey. In *2016 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, pages 562–567. IEEE.

Shi, S., Liu, C., Shen, Y., Yuan, C., and Huang, Y. (2015). Autorm: An effective approach for automatic web data record mining. *Knowledge-Based Systems*, 89:314–331.

Simon, K. and Lausen, G. (2005). Viper: augmenting automatic information extraction with visual perceptions. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 381–388. ACM.

Sleiman, H. A. and Corchuelo, R. (2012). A survey on region extractors from web documents. *IEEE Transactions on Knowledge and Data Engineering*, 25(9):1960–1981.

Varlamov, M. and Turdakov, D. Y. (2016). A survey of methods for the extraction of information from web resources. *Programming and Computer Software*, 42(5):279–291.

Velloso, R. P. and Dorneles, C. F. (2013). Automatic web page segmentation and noise removal for structured extraction using tag path sequences. *JIDM*, 4(3):173.

Velloso, R. P. and Dorneles, C. F. (2017). Extracting records from the web using a signal processing approach. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 197–206. ACM.

Velloso, R. P. and Dorneles, C. F. (2019). Web page structured content detection using supervised machine learning. In *International Conference on Web Engineering*, pages 3–18. Springer.

Wai, F. K., Yong, L. W., Thing, V. L., and Pomponiu, V. (2017). Cmdr: Classifying nodes for mining data records with different html structures. In *TENCON 2017-2017 IEEE Region 10 Conference*, pages 1862–1862. IEEE.

Wang, J., Wang, H., Wang, Z., and Zhu, K. Q. (2012). Understanding tables on the web. In *International Conference on Conceptual Modeling*, pages 141–155. Springer.

Xie, X., Fang, Y., Zhang, Z., and Li, L. (2012). Extracting data records from web using suffix tree. In *Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics*, page 12. ACM.

Zhai, Y. and Liu, B. (2005). Web data extraction based on partial tree alignment. In *Proceedings of the 14th international conference on World Wide Web*, pages 76–85. ACM.

Zhang, Z., Zhu, K. Q., Wang, H., and Li, H. (2013). Automatic extraction of top-k lists from the web. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, pages 1057–1068. IEEE.