UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Roberto Panerai Velloso

Optimized Record Extraction From Web Pages Using Signal Processing and Machine Learning

Florianópolis (SC)
2020

Roberto Panerai Velloso

Optimized Record Extraction From Web Pages Using Signal Processing and Machine Learning

Tese submetida ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Santa Catarina para a obtenção do título de doutor em Ciência da Computação.
Orientadora: Prof^{a}. Carina Friedrich Dorneles, Dr^{a}.

Florianópolis (SC)
2020

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Velloso, Roberto Panerai
   Optimized Record Extraction From Web Pages Using Signal Processing and Machine Learning / Roberto Panerai Velloso ; orientador, Carina Friedrich Dorneles, 2020.
   79 p.

   Tese (doutorado) - Universidade Federal de Santa Catarina, Centro Tecnológico, Programa de Pós-Graduação em Ciência da Computação, Florianópolis, 2020.

   Inclui referências.

   1. Ciência da Computação. 2. mineração na web. 3. extração de registro. 4. detecção de estrutura. 5. detecção de conteúdo. I. Dorneles, Carina Friedrich. II. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Ciência da Computação. III. Título.

Roberto Panerai Velloso

Optimized Record Extraction From Web Pages Using Signal Processing and Machine Learning

Florianópolis (SC), 2020.

Dedico este trabalho à minha família e a todos que me
apoiaram e me suportaram durante sua realização.

ACKNOWLEDGEMENTS

Agradeço à minha esposa Ana Karina e minha filha Luísa pelo apoio, colaboração e compreensão. Aos meus pais por me ensinarem o valor da educação e do conhecimento, sempre me incentivando a seguir adiante. Aos meus irmãos por estarem sempre disponíveis toda vez que precisei de ajuda. À minha orientadora pelo tempo dedicado, por todas as críticas construtivas, pela orientação e pela paciência ao longo destes anos.

"I don't know anything, but I do know that everything is interesting if you go into it deeply enough."
Richard Feynman

"One of the basic rules of the universe is that nothing is perfect. Perfection simply doesn't exist. Without imperfection, neither you nor I would exist."
Stephen Hawking

"Sábio é o homem que chega a ter consciência da sua ignorância"
Apparício Fernando de Brinkerhoff Torelly

RESUMO

A extração de dados estruturados (i.e. registros) de páginas da web permite uma série de aplicações importantes e possui imenso valor devido à quantidade e diversidade de informações disponíveis que podem ser extraídas. Esse problema, embora amplamente estudado, permanece em aberto pois não é trivial. Devido ao volume dos dados, uma abordagem viável precisa ser automática e eficiente (e, é claro, eficaz). É apresentada aqui uma nova abordagem, automática e computacionalmente eficiente, usando técnicas de processamento de sinais para detectar regularidades e padrões na estrutura de páginas da web e também aprendizado de máquina supervisionado para classificar os dados extraídos como conteúdo ou ruído. Também é apresentado um estudo comparativo das várias técnicas de aprendizado de máquina supervisionado, incluindo ensembles homogêneos e heterogêneos, para resolver o problema de classificação de conteúdo e ruído em páginas da web. Utilizamos o aprendizado de máquina, especificamente, para resolver o problema de detectar conteúdo em dados semiestruturados (por exemplo, resultados de pesquisa de comércio eletrônico) em duas situações distintas: primeiro em um ambiente controlado contendo apenas documentos com conteúdo estruturado e depois; em um ambiente aberto em que a página da web que está sendo processada pode ou não ter conteúdo estruturado. As características usadas para classificar o conteúdo são obtidas automaticamente a partir da abordagem de extração. Além de comparar o desempenho entre diferentes modelos, também foi realizada uma ampla análise das combinações de caracteŕisticas para apurar sua relevância para o problema. A abordagem proposta segmenta a página da web, detecta as regiões de dados dentro do documento, identifica os limites (início e fim) dos registros, alinha os registros encontrados e os classifica como conteúdo ou ruído. Também é apresentada uma otimização da abordagem de extração ingênua. A otimização proposta melhora o limite superior de $O(nlogn)$ para $O(n)$, mantendo os mesmos resultados qualitativos (ou seja, sem perda de eficácia) e alcançando uma melhoria de 11,77% no tempo de execução. Os resultados demonstram que a abordagem proposta tem comportamento linear de complexidade de tempo e f-score de cerca de 93% em um ambiente controlado e 91% em um ambiente aberto. A abordagem proposta é mais eficiente e tão eficaz quanto o estado da arte além de abordar a questão da detecção de conteúdo, normalmente negligenciada na maioria dos trabalhos.

Palavras-chave: mineração na web. extração de registro. detecção de estrutura. recuperação de informação. alinhamento de registros. detecção de conteúdo. remoção de ruído.

RESUMO ESTENDIDO

INTRODUÇÃO

Na web existe um grande volume de informação disponibilizada para consumo dos usuários de maneira levemente (ou fracamente) estruturada (e.g., produtos em sites de comércio obedecem uma mesma formatação/padronização).

Estes dados, embora possuam alguma forma de organização, não estão prontos para serem utilizados por máquinas, sendo necessário organizá-los de maneira mais estrita e formal, de maneira que um software possa utilizá-los sem necessidade de intervenção humana.

As informações semi-estruturadas possuem os mais variados formatos (i.e., layouts) e pertencem aos mais variados domínios (e.g., compras, motores de busca, passagens aéreas, etc). Esses dois fatores dificultam bastante o problema de extração destes dados estruturados, pois a abordagem precisa lidar, de maneira geral, com vários formatos e domínios diferentes.

É apresentada aqui uma nova abordagem, automática e computacionalmente eficiente, usando técnicas de processamento de sinais para detectar regularidades e padrões na estrutura de páginas da web.

Uma outra questão que foi abordada na pesquisa é a existência de dados estruturados que não são conteúdo de interesse (e.g., template do documento, menus, rodapés, anúncios, etc). Esses dados, chamados de ruído, embora tenham estrutura, devem ser eliminados.

OBJETIVOS

O objetivo desta pesquisa é investigar maneiras de extrair dados estruturados de páginas da web e propor uma abordagem para o problema que seja eficaz, computacionalmente eficiente e exija o mínimo de intervenção humana.

A abordagem proposta extrai dados estruturados a partir dos mais variados layouts e domínios dentro das seguintes restrições: os registros devem ser contíguos na página e devem ter formato e tamanho similares.

Para realizar a extração dos dados, a página é convertida para uma representação de sequência e, em cima desta sequência, são utilizadas técnicas de processamento de

sinais para detectar padrões no documento.

Uma vez que os dados estruturados são detectados no documento, é realizada a sua classificação como conteúdo ou ruído, utlizando técnicas de aprendizado de máquina.

Em síntese, esta pesquisa aborda as seguintes questões:

1. extração eficiente de dados estruturados através do uso de técnicas de processamento de sinais;
2. extração eficaz de dados estruturados através do uso de técnicas de processamento de sinais;
3. classificação eficaz do conteúdo estruturado através de técnicas de aprendizado de máquina.

METODOLOGIA

Para avaliar o eficácia da abordagem proposta foram utilizadas as métricas de precisão, revocação, f-score e acurácia. A eficácia foi avaliada em dois cenários diferentes:

1. cenário controlado: todos os documentos processados tem conteúdo estruturado;
2. cenário aberto: os documentos processados podem ter apenas ruído estruturado e nenhum conteúdo estruturado;

A avaliação da eficiência computacional foi realizada de maneira empírica, medindo o tempo de execução de cada documento do dataset em função do seu tamanho.

Na construção dos modelos de machine learning, para classificação de conteúdo, também foram analisadas as estatísticas e a relevância das características utilizadas no treinamento dos modelos.

RESULTADOS

Na avaliação da eficiência computacional, a abordagem demonstrou comportamento linear em relação ao tamanho da entrada (O(n)).

Na avaliação da eficácia, foram obtidos os seguintes resultados nos cenários avaliados:

Table 1 – Sumário de resultados

Cenário	Precisão	Revocação	Acurácia	F-Score
cenário controlado	93.30%	93.85%	93.02%	93.57%
cenário aberto	90.45%	92.52%	91.33%	91.47

CONCLUSÃO

A abordagem proposta é mais eficiente e tão eficaz quanto o estado da arte além de abordar a questão da detecção de conteúdo, normalmente negligenciada na maioria dos trabalhos.

Palavras-chave: mineração na web. extração de registro. detecção de estrutura. recuperação de informação. alinhamento de registros. detecção de conteúdo. remoção de ruído.

ABSTRACT

Extracting structured data (i.e. records) from web pages enables a number of important applications and has immense value due to the amount and diversity of available information that can be extracted. This problem, although vastly studied, remains open because it is not a trivial one. Due to the scale of data, a feasible approach must be both automatic and efficient (and of course effective). We present here a novel approach, mostly automatic and computationally efficient, using signal processing techniques to detect regularities and patterns in the structure of web pages and supervised machine learning to classify extracted data as content or noise. We also present a comparative study using several supervised machine learning techniques, including homogeneous and heterogeneous ensembles, to solve the problem of classifying content and noise in web pages. We use machine learning, specifically, to tackle the problem of detecting content in semi-structured data (e.g., e-commerce search results) under two different settings: a controlled environment with only structured content documents and; an open environment where the web page being processed may or may not have structured content. The features are automatically obtained from the underlying extraction approach. Besides comparing the performance between different models we have also conducted extensive feature selection/combination experiments. Our approach segments the web page, detects the data regions within it, identifies the records boundaries, aligns the records and classifies them as content or noise. We also present an optimization over the naïve extraction approach. 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) and achieving 11.77% improvement in runtime efficiency. Results show linear time complexity behaviour and an f-score of about 93% in a controlled setting and 91% in an open setting. Our proposal is more efficient and just as effective as the state-of-the-art approaches and, in addition, we deal with content detection, which is neglect by most works.

Keywords: web mining. record extraction. structure detection. information retrieval. record alignment. content detection. noise removal.

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

Throughout the web, data are often presented in a semi-structured fashion (e.g., shopping items, news, search engine results, etc.) with the objective of organizing semantically related entities in order to facilitate the comprehension of the document by a human user. These semi-structured data, although varied in layout and domain, have similar characteristics that can be used to identify them inside a document and to improve their structure to the point where it is possible to access them as relational data (i.e., as structured data).

The extraction of structured information is undeniably both important and difficult. It is an important task due to the ever growing amount of information published and readily available on the web and because structured information can be used to enrich, and allow, a number of applications (e.g., price comparison, semantic keyword search, metasearch, etc.) (CAFARELLA, Michael J et al., 2008b; BALAKRISHNAN et al., 2015). It is a difficult task because the data are laid out for human consumption and published in a variety of formats and templates, as observed in Michael J. Cafarella et al. (2011). Moreover, the Beckman report on database research (ABADI et al., 2016) mentions the diversity of data as a key challenge in this research field. The subset of structured data we address in our work is a major data source according to Michael J. Cafarella et al. (2011), since it can be considered a generalization of the data addressed by WebTables (CAFARELLA, Michael J et al., 2008a), we quote from Michael J. Cafarella et al. (2011):

> "*Hypertext-based data models*. These models, in which page authors use combinations of HTML elements (such as a list of hyperlinks), perform certain data-model tasks (such as indicate that all entities pointed to by the hyperlinks belong to the same set); this category can be considered a **generalization** of the observation that HTML tables are used to communicate relations"

Identifying the underlying content structure in a web document was attempted before. The existing approaches range from solutions to specific instances of the problem (e.g., extracting only data formatted with specific HTML tags (CAFARELLA, Michael J et al., 2008a; ELMELEEGY et al., 2011; QIU et al., 2015)) to broader attempts that search for patterns in a web document (LIU, B. et al., 2003; LIU, B.; ZHAI, 2005; ZHAI; LIU, B., 2005; MIAO et al., 2009; GRIGALIS, 2013). They also vary in degree of automation where there can be supervised, semi-supervised and unsupervised approaches. It is our understanding that the scientific community should strive to achieve the highest possible level of automation when dealing with web extraction, due to the volume of data, otherwise an eventual approach will not scale up. Existing work fails to definitively solve the problem, as observed in past surveys (SLEIMAN; CORCHUELO, 2012; SCHULZ et al., 2016; ROLDÁN et al., 2019), they solve just a part of it. Besides this, in our research, we came to the conclusion that most proposals either rely too heavily on supervised methods or; are based on too simplistic heuristic rules and hence do not survive the test of time or; they have a high time complexity and can not handle web scale data. The ideal solution, in this scenario, should be as much unsupervised as possible, computationally efficient, effective (at least near perfect precision) and as general as possible (e.g., domain independent and HTML syntax independent).

Our work is a compromise between a general approach and a specific one because we have developed it based on general observations about structured data, but we address only a subset of all structured data: records with similar structure **and** size (note that we do not limit ourselves to specific templates or HTML tags). By doing that we trade recall for precision and keep some degree of generality. Generality, here, is desired to avoid being ephemeral: too specific approaches tend to rapidly become obsolete. We also managed to do so mostly automatically[1], efficiently, without rendering the full web page in a browser engine, and without heuristic rules associated with HTML tags (that is undesirable because HTML syntax and programming practices change over time, compromising the approach's longevity).

In summary, our research approaches the following issues:

1. **efficient** extraction of structured data through the use of signal processing techniques;

2. **effective** extraction of structured data through the use of signal processing techniques;

3. **effective** content classification through the use of machine learning techniques.

Here we outline our approach to tackle the problem of structured extraction (or record extraction) and content detection from web pages, using an alternative representation of the DOM tree, some signal processing techniques and supervised machine learning to detect useful content. Our approach first converts the DOM tree to a sequence representation; segments the sequence into regions; identifies the regions with structured content; locates the records boundaries within the structured content regions; classifies extracted data as content or noise and; aligns the records into a table as shown in Figure 1. The technique outlined here has the following characteristics: it takes a single page as input, it is computationally efficient ($O(n)$ on the size of the document) and domain independent. It is also effective, as demonstrated in the results, and HTML syntax independent, i.e., there are no heuristic rules targeted at specific tags.

[1] with the exception of content detection, where we employ supervised machine learning

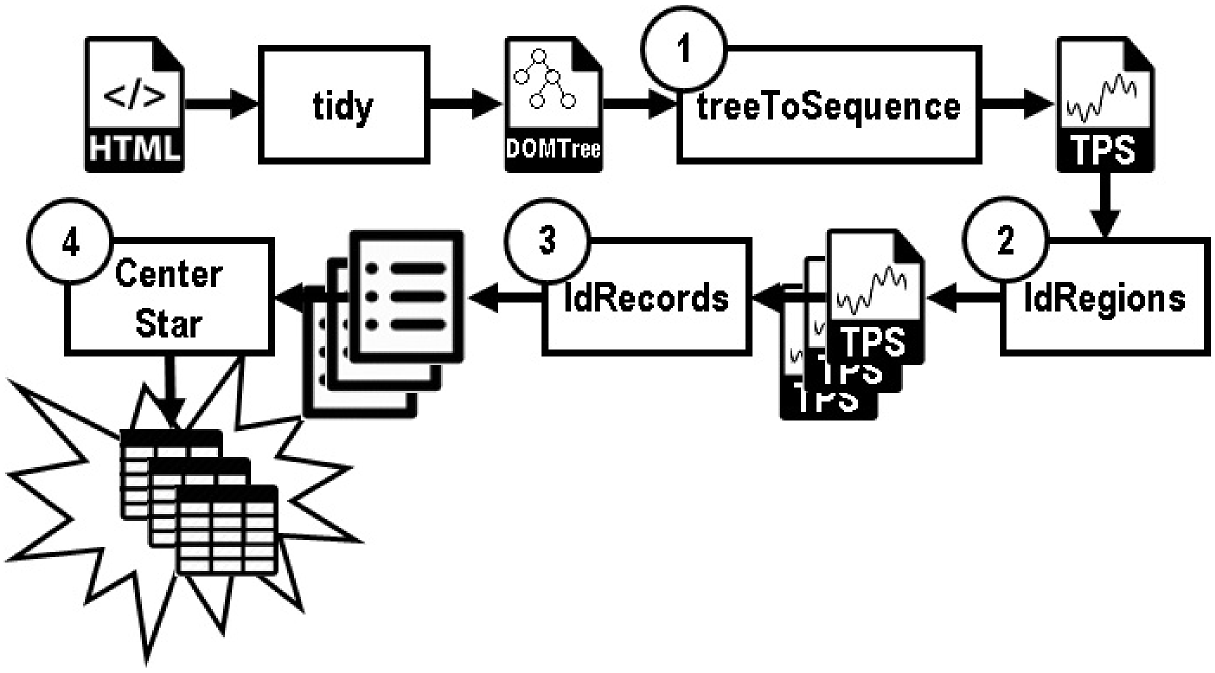


Figure 1 – Extraction system overall diagram.

1.1 FINDING PATTERNS IN A SEQUENCE

When the DOM tree is converted into its sequence representation, and plotted in a graph, the patterns of structured data become evident (see Figure 2 for instance). That said, the research area concerned with signal processing has been dealing with pattern detection and recognition long before the web became what it is today (e.g., Fast Fourier Transform – FFT(COOLEY; TUKEY, 1965) algorithm as published in 1965) and as a consequence it is much more developed in that aspect, so it is only logical to employ some of these consolidated techniques once we succeed converting the problem of record extraction into a signal processing problem. The web page depicted in Figure 2, as well as its sequence, will be used to illustrate various concepts and details of our approach throughout this thesis.

The adopted sequence representation enables the use of signal processing techniques that, otherwise, would probably not be possible. The areas of the sequence (i.e., the subsequences) that we are concerned about are those that show a repeating pattern, so we segment the sequence into its cyclic subsequences, which represent structured data regions, using the finite difference (Definition 3.10). Within each of this cyclic subsequences we use its power spectrum (Definition 3.7) to find their main period (which represents the size of the records) and frequency (which represents the number of records within that data region). Once we have the main period (record size) and frequency (record count) we use this information to break down the data region into records that are subsequently aligned.

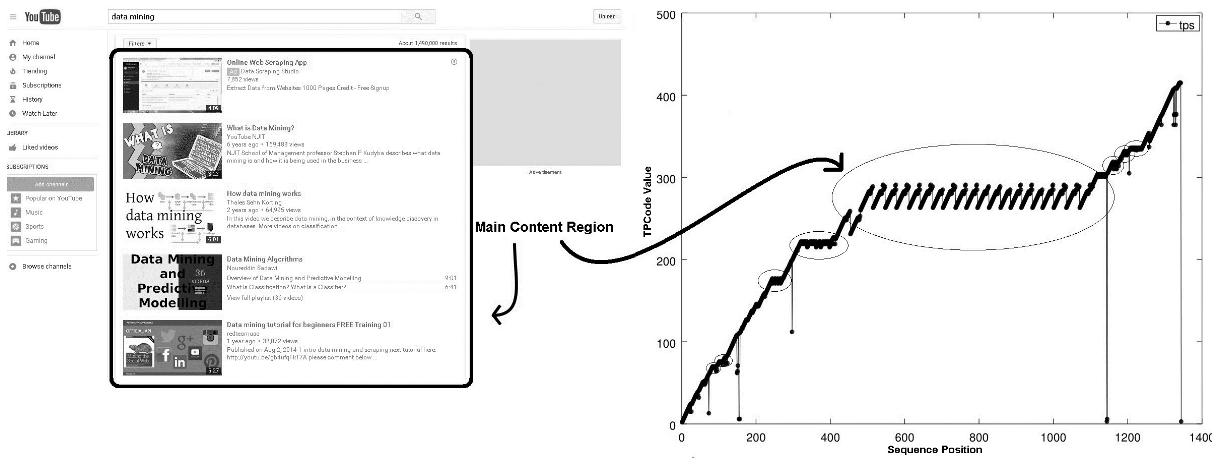


Figure 2 – How structured content appears in the sequence (structured regions encircled).

1.2 PROBLEM DEFINITION

Given an HTML document and its respective DOM Tree converted to sequence representation called **doc** we want to identify the set **reg** of subsequences of **doc** containing contiguous records with similar structure and size or, equivalently, the set **reg** of subsequences which exhibit cyclic behavior.

Given a structured subsequence **seq** belonging to **reg** we want to identify how many records, lets call this quantity **n**, there are in **seq** and the set **p** of their respective positions. Once we know **n** and **p** we can break **seq** into **n** records and align their fields.

1.3 DEALING WITH NOISE

At a first glance one might think that, in order to extract structured data, it should be enough to detect its underlying structure and be done with it. That, alone, is already a somewhat difficult task, but the problem with this approach is that among structured data there is also structured noise: information that is well organized but that we do not care about (i.e., document template, ads, menus, headers, footers and so on). We have found, in the course of our research, that a significant amount of works in this area neglect this fact, considering all structured data as content. In our work we took that into account and demonstrated to be possible to distinguish noise from content using only syntactic information (i.e., features from the structure of a document). We must be able to identify content, distinguish it from noise, so we do not end up with an unusable, bloated database, full of unimportant information (i.e., noise). According to Gibson et al. (2005) and our own findings (VELLOSO; DORNELES, 2013) between 40%-50% of a web document corresponds to noise (menus, template, ads). This amount is more than enough to completely compromise extraction precision.

So, how can we distinguish between content and noise? We could not find in the literature, nor did we reach a deterministic and closed form way of solving this problem, for this reason we have decided to characterize (create conjectures for the features) content and noise the best we could and try out machine learning models to approximate a satisfactory solution.

Supervised machine learning learns a function that maps an input to an output based on several labeled examples. This mapping can be used for classification or regression, depending on the problem (in our case we have modeled it as a classification problem). There are several learning algorithms, some models are learned using gradient descent algorithm (partial derivative of error function), other through the use of information entropy (e.g., decision tree models), optimization techniques (e.g., SVM), among others. Due to this diversity of methods and models it is very important, and productive, to use a well-known and tested framework (e.g., scikit-learn (PEDREGOSA et al., 2011)).

One can also use a combination of several machine learning models with the objective of attaining better results than any other method alone. This technique is called "Ensemble Learning". We have investigated two types of ensembles if our thesis:

- Soft Voting Ensemble: a collection of models, each pondered by a weight, vote for the answer/prediction;
- Stacking Ensemble: the outputs of several models are used as input features of a new model.

We have tackled this problem in our thesis by analyzing several supervised machine learning models for structured content detection. Our investigation considered eight machine learning techniques (Logistic Regression, Gaussian Naive Bayes, k Nearest Neighbours, Support Vector Machine, Extra Trees, Gradient Boosting, Voting and Stacking Ensembles) and all possible combinations of features within each approach to find the one that suited best in each case and at the same time investigate feature importance. We have conducted a fairly extensive statistical and empirical analysis of the features, demonstrating their importance and effectiveness solving the problem at hand.

1.4 COMPUTATIONAL EFFICIENCY

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, specifically, feasibility means, most of the time, space and time complexity. With that in mind we have devoted some time and attention to the optimization of our extraction approach. Our implementation seems to be limited by the time complexity ($O(nlogn)$) of the Fast Fourier transform (FFT). We use the FFT to detect the size and the number of record within a structured region of a web page. That seems to be a hard limit to beat, since it is not know whether or not the FFT can be performed in less time, but nonetheless, in our specific case, we have managed to overcome it by avoiding unnecessary computation and circumventing other issues that emerged as a consequence of this optimization. By doing that we have lowered the time complexity from $O(nlogn)$ to $O(n)$ (more details on that at Section 4.2 and Subsection 5.2.4).

1.5 CONTRIBUTIONS

The contributions presented in this thesis are the following:

1. a novel insight on how the structure of a web document can be seen: as a periodic signal. To the best of our knowledge there is nothing alike in the literature and the results depict both its efficiency and effectiveness;
2. a novel web page segmentation technique;
3. a new and efficient way of checking if a given region of a web page is structured or not. Such a framework, that is independent of the extraction process (i.e., one does not need to extract the entire document), is useful to avoid processing portions of the document that are unpromising for record extraction, clustering, indexing or any other end;
4. a more efficient approach, when compared to the state-of-the-art, that is due to the use of efficient signal processing algorithms. Most existing works rely mainly on dynamic programming (e.g., edit distance), clustering algorithms and/or full page rendering and hence are less efficient;
5. a small set of features, extracted from structured data regions, that can be used to classify such a region as content or noise using supervised machine learning models.

During the course of our research three papers were published:

1. "Extracting records from the web using a signal processing approach", published at CIKM'17, rated EA1 by CAPES(VELLOSO; DORNELES, 2017);
2. "Web Page Structured Content Detection Using Supervised Machine Learning", published at ICWE'19, rated EB1 by CAPES(VELLOSO; DORNELES, 2019);
3. "Optimized Extraction of Records from the Web Using Signal Processing and Machine Learning", published at SBBD'20, rated EB2 by CAPES(VELLOSO; DORNELES, 2020).

1.6 ORGANIZATION

The rest of the thesis is organized as follows. In Chapter 2, we review the work related to structured extraction; in Chapter 3, we present some definitions required to better understand the proposed approach for the problem; in Chapter 4, we outline the proposed approach; in Chapter 5, we detail our experimental setup as well as our framework, datasets and results and; in Chapter 6, we conclude and present some of our ideas for future work.

2 RELATED WORK

The task of extracting records from web pages has been approached from many angles. Some authors proposed *ad hoc* solutions while others tried broader approaches for the problem.

We have grouped the papers into four categories:

1. extraction from specific sources (Section 2.1);
2. approaches based on general observation about structured data (Section 2.2);
3. approaches that use a sequence representation similar to the one we use in our work (Section 2.3);
4. surveys on structured information (Section 2.4).

2.1 EXTRACTION FROM SPECIFIC SOURCES

The papers under this category carry out the extraction of records from very specific sources like data formatted with `<table>`, `<ul/ol/etc.>` or even specific templates. Although these approaches may seem to be limited, at least at a first glance, they actually yield sound results, as documented in Michael J Cafarella et al. (2008b), Wu et al. (2012) and Michael J Cafarella (2012), and these results can then be used to leverage subsequent extractions and other applications. Such quality of results are due to the large amount and heterogeneity of data in the web. From that fact we get an important premise when dealing with web scale data: recall can be traded off for precision. That is what those approaches are all about, they tackle very specific situations (sacrificing recall) in a very specialized way (ensuring high precision).

In Michael J Cafarella et al. (2008a) and Jingjing Wang et al. (2012) the extraction is done solely over data formatted using the `<table>` tag. In (CAFARELLA, Michael J et al., 2008a) they extract only tables with schema information in line/column header, using a logistic regressor to identify if a table has schema information or not, no alignment is needed since the data are already in tabular format. In Jingjing Wang et al. (2012) header/schema detection is done using heuristics instead of a supervised classifier and if a header is not found they try to create one from previously known headers. Both works build a taxonomy from extracted schema and reuse it to improve future extractions (sort of a feedback loop) and leverage other applications.

In Elmeleegy et al. (2011) and Chu et al. (2015) the extraction is done over data formatted using list tags (e.g., `<ul/ol/etc.>`). Compared to table extraction there is the additional difficulty of finding out how to aligning the list items into columns (i.e.„ identifying the fields). In Elmeleegy et al. (2011) each list item is split in multiple positions, generating several candidate tables. The objective is to identify and align the fields of each table row by maximizing a set of consistency scores, defined by the authors, in a refinement process. In Chu et al. (2015) the item splitting is done globally, considering the consistency of every list item during the process. The authors have modeled the problem as an optimization problem and employed dynamic programming to solve it.

In Zhang et al. (2013) the data is extracted from a single specific template: singe page top-k items (e.g., top 10 restaurants in France, etc.). The authors have used the taxonomy generated by Wu et al. (2012) and a Conditional Random Fields (CRF) classifier to identify these pages by their title and with the information acquired from this first step (the subject and number of items in the list) a set of candidate tables is generated and ranked, using heuristics, for extraction.

In Qiu et al. (2015) supervised machine learning is used to detect tables and lists in detail pages containing product specifications from e-commerce sites. This structured information is then extracted as attribute-value pairs.

In Guo et al. (2019) records are extracted from search engines results exploiting the redundancy present in multiples result pages from the same search. The information gathered from the search results is used to leverage extraction from detail pages.

2.2 EXTRACTION BASED ON GENERAL OBSERVATION

The approaches under this category were designed from more general observations about structured data. These proposals try to solve the problem in a broader sense (which is a much harder problem to solve) grounded on general observations about structured content. Some approaches make use of visual information, rendering the entire document in a browser engine before processing it. Although each of these works represent advances in the field of structured extraction, most of them do not provide, like their *ad hoc* counterparts do, readily usable results (i.e., end-to-end production grade results), but they do serve as stepping stones for future works to catch on. For instance, some works focus on record extraction only, disregarding alignment, others focus on main data region detection and so on.

The works in Crescenzi et al. (2001), Arasu and Garcia-Molina (2003) and Kayed and Chang (2009) all describe a site oriented technique that exploits the similarities between documents using the same template. Several pages with the same template are used to build a template model that is applied in the extraction of subsequent pages with the same template. In short: what remains constant is template; what changes is the data. In Arasu and Garcia-Molina (2003), according to the authors, they overcame some of the limitations found in Crescenzi et al. (2001) like scalability. In Kayed and Chang (2009) they use tree alignment to construct a more general template model and they claim to achieve higher precision than Arasu and Garcia-Molina (2003).

In Bing Liu et al. (2003), Zhai and Bing Liu (2005), Bing Liu and Zhai (2005), Jindal and Bing Liu (2010), Shi et al. (2015) and Wai et al. (2017) the authors propose a technique based on tree edit distance and tree alignment. The records in a web page are considered to be similar to each other and contiguous to one another. The records are detected when there is similarity between neighbouring subtrees in the DOM. In Zhai and Bing Liu (2005) and Simon and Lausen (2005) the same algorithm is used and visual information, from the rendered document, is employed to improve extraction results. In Bing Liu and Zhai (2005) visual information is also used and the DOM is traversed post-order and item alignment occurs in a bottom-up fashion. In Jindal and Bing Liu (2010) the authors have tackled the problem of extracting records that contains nested lists. They collapse list items into a single node to avoid false negative when comparing two adjacent subtrees containing lists of different sizes. In Shi et al. (2015) the authors extended the work in Bing Liu et al. (2003) and Zhai and Bing Liu (2005) for detecting similar adjacent subtrees in the DOM, but they use more strict heuristics to detect record similarity in conjunction with tree edit distance. In Wai et al. (2017) the authors extended the work in Bing Liu et al. (2003) with deep learning to improve extraction results from online forums.

In Cortez et al. (2010) pre-existing knowledge bases in specific domains, containing a set of attribute-values on that domain, are used to perform unsupervised record extraction from semi-structured text. The text is segmented into substrings more likely to represent the occurrence of an attribute.

In Cai et al. (2003), Wei Liu et al. (2009) and Grigalis (2013) the web page is rendered in a browser and visual information, such as screen coordinates, is used in the extraction process. In Cai et al. (2003) a method to outline the general structure of a web page is proposed and this method is used in Wei Liu et al. (2009) to detect main content region and perform the actual extraction of records. In Grigalis (2013) visual information is used to enrich tag path strings, adding more details to them, besides the path alone, allowing for a more accurate clustering of tag paths that takes visual information into account.

2.3 USING A SEQUENCE REPRESENTATION

The papers under this category use an alternative document representation. Instead of using the DOM tree as document representation, the web page is transformed to a sequence of tag paths. Such representation enables the use of algorithms targeted at strings and sequences (e.g., suffix tree/array (UKKONEN, 1995; MANBER; MYERS, 1993), FFT (COOLEY; TUKEY, 1965), Lempel-Ziv compression (ZIV; LEMPEL, 1977), etc.), which are vast and well studied (GUSFIELD, 1997), unleashing new possibilities in this research field. Some of these algorithms were already used in this research area (e.g., Suffix Tree in Xie et al. (2012) and Fang et al. (2018)), the FFT is used here in this work and in Chapter 6 we suggest the use of LZ compression for record extraction as a promising and very interesting possibility.

In Miao et al. (2009) the DOM is converted to a sequence of tag paths, similar to what we do in our work, and the elements of the sequence are grouped together using a spectral clustering algorithm. Elements with the same repeating pattern are clustered together, extracted as a data region and aligned as records. This is the first work we have found using sequence notation.

In Xie et al. (2012) and Fang et al. (2018) the DOM is transformed to a sequence of tag paths and a suffix tree of this sequence is built and used to find repeating subsequences. A set of heuristic rules is applied to avoid false positives and the remaining subsequences are aligned.

In Velloso and Dorneles (2013) the DOM is converted to a sequence of tag paths and the sequence is recursively sliced into disjoint subsequences (i.e., subsequences formed by disjoint sets of symbols). The goal of this work is to detect the main region of a web page with structured content. The largest disjoint subsequence is considered to be the main region, all the rest is discarded as noise. The extraction occurs at region level, not at record level.

2.4 SURVEYS ABOUT STRUCTURED EXTRACTION

Past surveys (SLEIMAN; CORCHUELO, 2012; FERRARA et al., 2014; SCHULZ et al., 2016; VARLAMOV; TURDAKOV, 2016; ROLDÁN et al., 2019) on the subject of structured extraction have reached the same conclusion about the state-of-the-art: existing works fail to definitively solve the problem, i.e., it is still an open research problem. All those surveys also agree about the problem encountered when one tries to compare two approaches: i) there is no up to date standard dataset/baseline; ii) most implementations are not publicly available and; iii) there is no standard procedure and/or framework to perform comparison. We can see, from these observations, that there is still a lot to do in this field, despite all those years of research.

2.5 RELATED WORK COMPARISON

We present, in Table 2, a comparison between approaches we have considered to be relevant in this research area. They are grouped by type and listed in chronological order. The comparison was carried out with respect to 5 criteria, enumerated below. Cells in “green” color comply with the criteria and cells in “red” color do not.

1. “KB”: is the approach based on some sort of predefined knowledge base?
2. “Rules/HTML”: does the approach use heuristic rules or is it dependent on the HTML syntax?

3. “Single Page”: is the approach able to carry out extraction over single pages or does it need several pages to do so?

4. “Render”: does the approach need to render the page in a browser engine?

5. “Noise”: does the approach care about filtering out noise from the extraction?

During the development of our research we have aimed at being independent from HTML syntax and avoiding the use of knowledge bases and strict heuristic rules. We believe that by attending these goals we have attained a higher degree of generality in our approach and ensured it will not be short lived. The columns “KB” and “Rules/HTML” carry out the comparison with this respect.

Our approach extracts repeating patterns from single pages and, because of that, it needs at least two records to be there for extraction. There are pros and cons when we choose to extract data from a single page or from several pages. For instance, our approach can not be used to extract data from detail pages. Although in Bing Liu and Chen-Chuan-Chang (2004) it is argumented that extracting from a single page is a main advantage and that makes some sense, since one does not need to crawl, download and preprocess several documents, to infer site template, prior to the extraction. The column “Single Page” carries out this comparison.

The column “Render” indicates whether the approach makes use of visual information by rendering the document in a browser engine. This practice, although it may improve extraction, incurs in a very high computational cost. It remains to be seen if it is worth or not.

The column “Noise” indicates if the approach considers all structured data as content or if it attempts to filter out structured noise from the output, preventing the extraction of information like menus, template structure, etc. Most works do not approach this matter and those who do, do so implicitly (i.e., it is intrinsic to the approach).

Another important criterion of comparison is the time complexity of each work. Unfortunately, only a minority of papers do some analysis of time complexity. For this reason we have not included this information in our table.

We can see in Table 2 that the requirements we have established for our research (i.e., no KB use, no *ad hoc* rules, single page processing, no rendering and account for noise) are not met by any of the related work. Those requirements were not randomly chosen, but because they are desirable properties of an extraction approach. We note that all “specific sources” works make use of *ad hoc* rules, for example, while all the others fail to meet at least one of the requirements. Our work tries to fill this gap (i.e., attain all requirements) while maintaining high quality state-of-the-art results and low computational complexity.

Table 2 – Related work comparison.

Paper	KB	Rules/HTML	Single Page	Render	Noise
Extraction from specific sources					
Michael J Cafarella et al. (2008a)	no	yes	yes	no	no
Elmeleegy et al. (2011)	yes	yes	yes	no	no
Jingjing Wang et al. (2012)	no	yes	yes	no	no
Zhang et al. (2013)	no	yes	yes	no	yes
Chu et al. (2015)	yes	yes	yes	no	no
Qiu et al. (2015)	no	yes	yes	no	no
Guo et al. (2019)	no	yes	no	no	yes
Extraction based on general observation					
Crescenzi et al. (2001)	no	no	no	no	no
Arasu and Garcia-Molina (2003)	no	no	no	no	no
Bing Liu et al. (2003)	no	no	yes	no	no
Cai et al. (2003)	no	yes	yes	yes	no
Zhai and Bing Liu (2005)	no	no	yes	yes	no
Bing Liu and Zhai (2005)	no	no	yes	no	no
Kayed and Chang (2009)	no	no	no	no	no
Wei Liu et al. (2009)	no	yes	yes	yes	no
Jindal and Bing Liu (2010)	no	no	yes	no	no
Cortez et al. (2010)	yes	yes	yes	no	yes
Grigalis (2013)	no	no	yes	yes	no
Shi et al. (2015)	no	yes	yes	yes	no
Wai et al. (2017)	no	yes	yes	yes	no
Using a sequence representation					
Miao et al. (2009)	no	no	yes	no	no
Xie et al. (2012)	no	yes	yes	no	no
Velloso and Dorneles (2013)	no	no	yes	no	yes
Fang et al. (2018)	no	yes	yes	no	no
Our work	no	no	yes	no	yes

3 DEFINITIONS AND PRELIMINARIES

In this section we define some concepts for the sake of better understanding our approach and the motivation behind each of these concepts.

Definition 3.1 ***(Tag Path)*** *is a string describing the path from the root node of the DOM tree to every other node in the tree. The path is enriched with style definitions.*

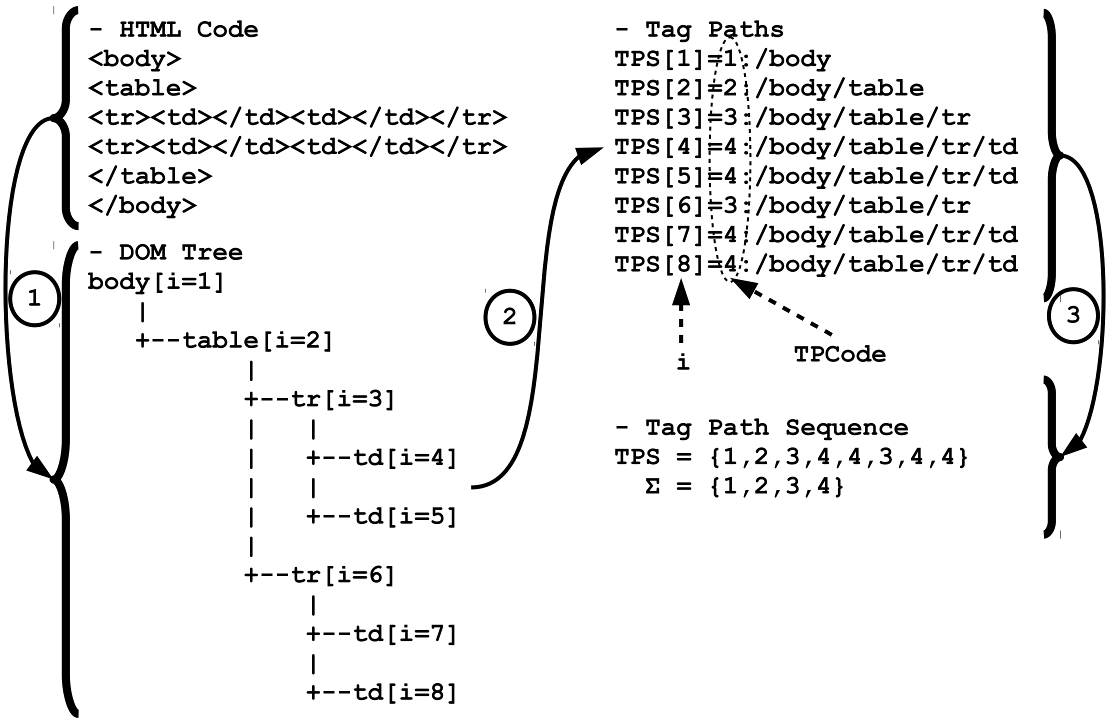
For example: "`html/body/table/tr/td/#text`". To better characterize each path we include style definitions of every node in the path, like this: "`html/body/table/tr class=tbrow/td class=tbcell/#text bgcolor=red`".

Definition 3.2 ***(Tag Path Code – TPCode)*** *is a numeric ascending code assigned to every different tag path string encountered in the tree, in order of appearance. If a given path has occurred in the past, it is assigned the same code as before. The paths are built in depth first order.*

Figure 3 shows an example of this definition. Here we refer to TPCode as "symbol" and a set of TPCodes as "alphabet".

Definition 3.3 ***(Tag Path Sequence – TPS)*** *is a sequence of TPCodes in the same order as they were built from the DOM tree.*

Figure 3 shows the resulting TPS for an HTML snippet as well as the set of TPCode used for that the sequence. In this thesis we also refer to TPS as simply "sequence".



```
- HTML Code
<body>
<table>
<tr><td></td><td></td></tr>
<tr><td></td><td></td></tr>
</table>
</body>

- DOM Tree
body[i=1]
    |
    +--table[i=2]
           |
           +--tr[i=3]
           |    |
           |    +--td[i=4]
           |    |
           |    +--td[i=5]
           |
           +--tr[i=6]
                |
                +--td[i=7]
                |
                +--td[i=8]
```

```
- Tag Paths
TPS[1]=1 /body
TPS[2]=2 /body/table
TPS[3]=3 /body/table/tr
TPS[4]=4 /body/table/tr/td
TPS[5]=4 /body/table/tr/td
TPS[6]=3 /body/table/tr
TPS[7]=4 /body/table/tr/td
TPS[8]=4 /body/table/tr/td

- Tag Path Sequence
TPS = {1,2,3,4,4,3,4,4}
  Σ = {1,2,3,4}
```

Figure 3 – tag path sequence built from HTML code snippet.

Definition 3.4 ***(Coefficient of Variation – CV)*** *is a statistical measure of data dispersion and is defined as the ratio between the standard deviation and the mean as shown in Equation 1 (EVERITT, 2010).*

$$CV = \frac{\sigma}{\mu} \tag{1}$$

We use a measure of dispersion to find out how well distributed a symbol is in a sequence and we chose the CV, specifically, because it is a **standardized** measure of dispersion. In our approach we need to compare the CV of symbols from several different sequences against the same threshold, so a standardized measure is needed. For example, consider the sequence $tps(1..12) = [1, 2, 3, 4, 5, 4, 5, 4, 5, 6, 4, 5]$ and the symbol $s = 4$ of this sequence; the symbol occurs at positions $[4, 6, 8, 11]$ in the sequence and the distances between adjacent occurrences are $[6-4 = \mathbf{2}, 8-6 = \mathbf{2}, 11-8 = \mathbf{3}]$, so the CV for symbol "*s*" is equal to $\frac{\sigma([2,2,3])}{\mu([2,2,3])} = 0.24744 = 24.744\%$. The more evenly distributed a symbol is, along the sequence, the lower the CV and vice versa.

Definition 3.5 ***(Discrete Fourier Transform – DFT)*** *is a mathematical transformation that decomposes a discrete sequence into its frequency components (Equation 2).*

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

Definition 3.6 ***(Power Spectrum Density – PSD)*** *is the squared absolute value of the DFT as defined in Equation 3.*

$$PSD = |X_k|^2 \tag{3}$$

The result of the DFT is complex-valued, the real part represents the amplitude and the imaginary part represents the phase. To evaluate the contribution of a frequency component in the overall sequence we use the power spectrum density (PSD), which is real-valued, as defined in Equation 3. Here, we may refer to the PSD as simply "spectrum". Figure 5 shows a discrete sequence (a) and its corresponding PSD standardized to *z-score* (b). We use the PSD to detect record count (main frequency) and record size (main period) in structured regions. A structured region is a cyclic subsequence (or periodic) and, as such, its PSD displays a higher peak at the frequency that corresponds to the main period (as we can see in Figure 5b). This kind of analysis is made possible by the decomposition of the signal into a summation of **weighted** frequencies.

Definition 3.7 ***(Fast Fourier Transform – FFT)*** *is an efficient implementation of the DFT that yields the same result.*

The naïve implementation of the DFT has $O(n^2)$ time complexity whereas the FFT has $O(nlogn)$ time complexity. For more details about DFT, FFT and discrete signal processing in general we refer the reader to Oppenheim et al. (1989), which covers the subject in great depth, since this is not the scope of this paper.

Definition 3.8 ***(Z-Score)*** *or standard score (EVERITT, 2010), is a standardization method with respect to the data's mean and standard deviation, as defined by Equation 4, where x is the data being converted. The z-score means the distance, in number of standard deviations, from the mean.*

$$z = \frac{x - \mu}{\sigma} \tag{4}$$

We use the z-score to convert the PSD of all structured regions to a common ground, so they can be validated against the same threshold. This is needed because each structured region has a different range of values and varying sizes, so standardization is required if they are to be compared against a common threshold.

Definition 3.9 ***(Sequence Contour)*** *is actually the upper contour of a sequence. In other words, the contour is the maximum value seen so far in the sequence. It can be thought of as the skyline of the sequence.*

The contour is important, in our context, because it is flat throughout a structured region as illustrated in Figure 4a. Function `contour` of algorithm 2 (Lines 8–17) details the construction of a sequence's contour.

Definition 3.10 ***(Finite Difference)*** *is the difference between adjacent values of a discrete sequence as defined in Equation 5.*

$$d[n] = x[n] - x[n-1] \tag{5}$$

We use the finite difference, in conjunction with contour (Definition 3.9), to break down a sequence into cyclic regions. By computing the difference (Figure 4b) of the sequence's contour (Figure 4a) we find the intervals where the difference is zero and consider them to be candidate data regions that are submitted to subsequent analysis and validation to verify if they are indeed structured. This process is explained in depth in Subsection 4.1.2.

Definition 3.11 ***(Structured Region)*** *is a region of the document that contains contiguous structured data (either content or noise) and, because of its structured nature, when converted to a TPS, exhibits a cyclic behaviour.*

This cyclic behaviour is a consequence of structure: the records are contiguous and have similar structure so the TPCs forming the structured region's TPS will repeat throughout the sequence, in cycles. This is illustrated in Figure 2 where a document containing 20 SRRs (search result records) is converted to its TPS representation and we can see that each record becomes a cycle in the encircled main content region.

4 RECORD EXTRACTION

In this chapter we outline our approach in detail. In Section 4.1 we detail how structure is detected in a web page; in Section 4.2 we show how we managed to reduce the time complexity of our approach from $O(nlogn)$ to $O(n)$ and; in Section 4.3 we detail the features we used to train our content/noise classifier.

4.1 STRUCTURED DATA DETECTION AND EXTRACTION

We present here, in this section, the details of our proposal to tackle the problem of automatically detecting structured data within web pages. The technique is subdivided in four main steps, namely: sequence representation, region identification, record identification and record alignment. Each step is detailed in Subsections 4.1.1, 4.1.2, 4.1.3 and 4.1.4 respectively.

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 representation.

Figure 1 depicts the entire process of record extraction proposed here. 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 splits the TPS into several structured regions; Step 3 identifies the record boundaries within each region and; Step 4 aligns the records in tabular form.

4.1.1 Sequence Representation

Most of the works in this research field use the DOM tree to represent the documents subjected to record extraction. We choose, instead, an alternative representation: a sequence of tag paths. This representation allows us to see the web page as a sequence instead of a tree, enabling us to make use of algorithms targeted at sequences. Such a representation was used before in (MIAO et al., 2009; XIE et al., 2012; VELLOSO; DORNELES, 2013).

The translation process from DOM tree representation to tag path sequence is depicted in Figure 3. The HTML code is converted to a DOM tree in Step 1; the DOM tree is converted to a sequence of tag paths in Step 2 and; in Step 3 the TPS is built by assigning TPCodes to each tag path.

Algorithm 1 presents the TPS build process. It is basically a depth first search that assembles the path from the root of the tree to every other node and assigns a code to each distinct path found. To help distinguish different regions of the document, during the region identification step, we have added style definitions to the tag paths, similar to what was done in Grigalis (2013) with the difference that here we use class and in-line style attributes instead of rendering the entire document, avoiding the computational cost associated with such a task.

Algorithm 1 Builds a tag path sequence from a DOM tree

```
1: procedure TREETOSEQUENCE(node,tp,tps by ref.)
2:     $tp \leftarrow concat(tp, \text{“/”}, node.\text{tag}, node.style)$
3:     if $tp \ni tagPathMap$ then
4:         $tagPathMap \leftarrow tagPathMap \cup \{tp\}$
5:         $tagPathMap[tp].code \leftarrow tagPathMap.size$
6:     end if
7:     $tps \leftarrow concat(tps, tagPathMap[tp].code)$;
8:     for each child of node do
9:         $treeToSequence(child, tp, tps)$
10:    end for
11: end procedure
```

Description of Algorithm 1: procedure `treeToSequence` is initially called with the tree's root node, an empty tag path string and an empty sequence: `treeToSequence(root, ‘“’, tps = ‘“’)`. In Line 2 the current tag path string is assembled; in Lines 3–6 the tag path code is retrieved, if it is a recurring path, otherwise a new, ascending, code is assigned to it and the path string is stored for future reference; In Line 7 the current TPCode is appended to the TPS and; In Line 9 the procedure is called recursively for every child of the current node. When the procedure returns, the full TPS is stored in parameter `tps` which was passed by reference.

4.1.2 Region Identification

Before extracting the records themselves we first isolate the regions that contain structured data. It is simpler and easier to extract records from a delimited region that is known to contain structured data than it is to do so from the entire document. The reason for this is that structured regions are cyclic, i.e., since the records have similar structure and are contiguous then its tag path sequence will display a cyclic behaviour as illustrated in Figure 2.

We can see in Figure 2 that whenever there is structured content in the document (i.e., repeating records with similar structure), be it the main content, menus, or anything else, the corresponding interval in the sequence displays a cyclic behaviour (the encircled subsequences in Figure 2). In other words, the sequence stabilizes during the structured portions. That happens because the records look alike and so their paths and styles repeat themselves. We took advantage of this observation to devise Algorithm 2 that isolates the structured segments of the sequence.

Algorithm 2 Identifies structured regions in a document

Input: tag path sequence
Output: a set of structured regions

```
function IDREGIONS(tps)                                          ▷ Main Function
    $tpsContour \leftarrow$ CONTOUR($tps$)
    $regions \leftarrow$ SEGMENT($tpsContour$)
    $regions \leftarrow$ MERGEREGIONS($regions$)
    $regions \leftarrow$ FILTERREGIONS($regions$)
    return $regions$
end function
function CONTOUR(tps)                                            ▷ Subroutine
    $maxHeight \leftarrow 0$
    for $i \leftarrow 1..length(tps)$ do
        if $tps[i] > maxHeight$ then
            $maxHeight \leftarrow tps[i]$
        end if
        $contour[i] \leftarrow maxHeight$
    end for
    return $contour$
end function
function SEGMENT(contour)                                        ▷ Subroutine
    $diff \leftarrow difference(contour)$
    $regions \leftarrow \emptyset$
    $start \leftarrow end \leftarrow 1$
    for $i \leftarrow 1..length(diff)$ do
        if $diff[i] \neq 0$ then
            if $start \neq end$ then
                $regions \leftarrow regions \cup [start, end)$
            end if
            $start \leftarrow end \leftarrow i + 1$
        else
            $end \leftarrow end + 1$
        end if
    end for
    return $regions$
end function
function MERGEREGIONS(regions)                                   ▷ Subroutine
    $merged[1] \leftarrow regions[1]$
    $j \leftarrow 1$
    for $i \leftarrow 2..regions.count()$ do
        $\Sigma_{prev} \leftarrow alphabet(merged[j])$
        $\Sigma_{curr} \leftarrow alphabet(regions[i])$
        if $\Sigma_{prev} \cap \Sigma_{curr} \neq \emptyset$ then
            $merged[j] \leftarrow concat(merged[j], regions[i])$
        else
            $j \leftarrow j + 1$
            $merged[j] \leftarrow regions[i]$
        end if
    end for
    return $merged$
end function
function FILTERREGIONS(regions)                                  ▷ Subroutine
    for each region in regions do
        $angCoef \leftarrow linearRegression(region)$
        if $|angCoef| > threshold$ then
            $regions \leftarrow regions - \{region\}$
        end if
    end for
    return $regions$
end function
```

Description of `idRegions` function: this is the main function that calls the other subroutines. It receives a TPS as input and returns the detected structured regions as output;

Description of `contour` **function:** this function is pretty straightforward, it receives the TPS as input, iterates over the sequence (Line 10), computes its upper contour (i.e., the maximum value seen so far) in Lines 11–15 and then; returns the sequence's contour (Line 16);

Description of `segment` **function:** this function receives the sequence's contour as input, computes its finite difference (Line 19), as in Definition 3.10; assembles a set with all intervals where the finite difference is zero-valued (Lines 20–31) and then; returns this set of possibly structured regions (Line 32);

Description of `mergeRegions` **function:** this function receives, as input, a set of possibly structured regions; iterates over this set (Line 37), merging the adjacent elements with intersecting alphabets (Lines 38–45) and then; returns the merged set of possibly structured regions (Line 47);

Description of `filterRegions` **function:** this function receives, as input, a set of possibly structured regions; iterates over this set (Line 50); computes the linear regression for each region in the set (Line 51); if the current region has an angular coefficient above the threshold (Line 52) it is removed from the set (Line 53) and finally; the remaining set of regions is returned (Line 56).

Algorithm 2 starts by computing the contour of the input sequence, in function `contour`, and its finite difference, in function `segment`, using this to segment the document into several regions. The idea behind function `segment` is that the finite difference of the contour will be zero whenever the contour stabilizes (i.e., when it remains constant during an interval). As we can see in Figure 4a there are flat (constant) intervals in the sequence's contour and consequently the first difference will be zero during those intervals, as shown in Figure 4b.

The resulting regions are then merged together according to their alphabets (i.e., set of symbols that form the sequence). If two adjacent regions share a common alphabet, they are merged into a single region, if their alphabet is disjoint (i.e., they share no common symbols) they are kept apart.

At last, to ensure that the identified regions contain structured content, they are filtered out according to their angular coefficient. This filtering step is necessary because when using the contour and its difference to segment the sequence, it is possible that some of the identified regions are spurious. If a region is indeed cyclic, its linear regression will yield a small angular coefficient. If it is not cyclic, then it is either increasing or decreasing and in both cases the linear regression will yield a much greater angular coefficient.

The computational complexity of functions *contour* and *segment* are linear in the size of the input sequence. For functions *mergeRegions* and *filterRegions* the complexity is linear in the number of regions detected, which in turn have a worst case

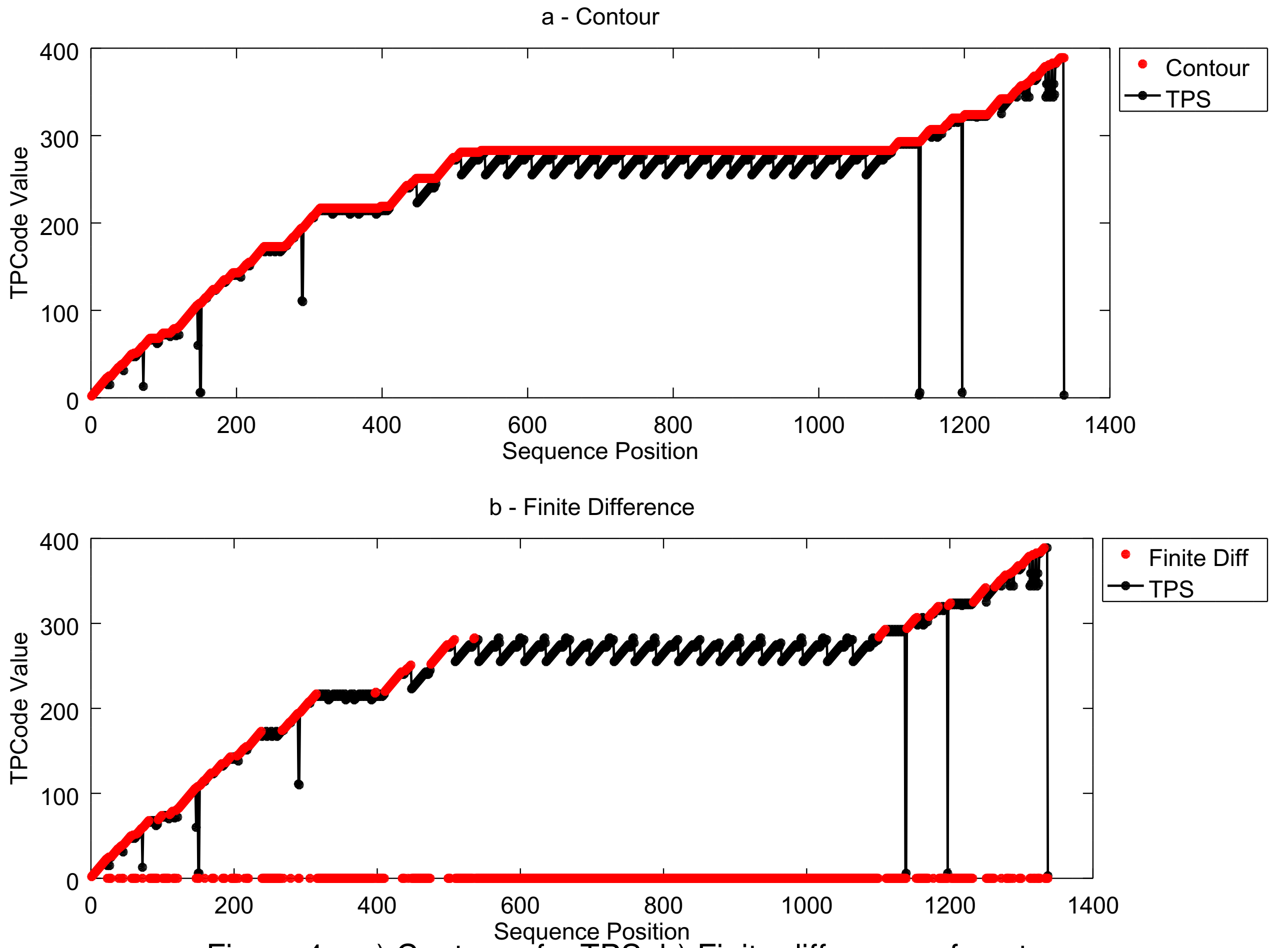


Figure 4 – a) Contour of a TPS; b) Finite difference of contour.

proportional to the size of the sequence, so the overall time complexity for function *idRegions*, in the worst case, is $O(n)$ where n is the size of the sequence.

4.1.3 Record Identification

For each region identified in the previous step we now try to extract records out of it. To do so, we analyze the tag paths that appear in the sequence against the region's spectrum (Definition 3.6). The spectrum is calculated using the FFT (COOLEY; TUKEY, 1965) (Definition 3.7). The FFT factors the sequence into its frequency components, so cyclic sequences will exhibit higher peaks in frequencies that correspond to the main period of the sequence as we can see in Figure 5 (the main content region of Figure 2). Figure 5a shows the subsequence that represents a structured region of the TPS and, hence, displays a cyclic behaviour and a linear regression with a small angle. In this case the linear regression displays an inclination of only 0.15 degrees (almost horizontally flat), if the region were unstructured (i.e., an increasing or decreasing subsequence) its angular coefficient would be much higher (e.g., 45 degrees of inclination); Figure 5b shows the subsequence's PSD with a highlighted frequency peak. The frequency peak is located at position 20 of the spectrum, which corresponds exactly to the number of records present in this region and if we divide the size of the region by the number of records we get the average record size (approximately 31 in this case). In short, this frequency peak represents the record count (or number of cycles in the subsequence) and its corresponding period (i.e., the subsequence length divided by the frequency) is equivalent to the average record size.

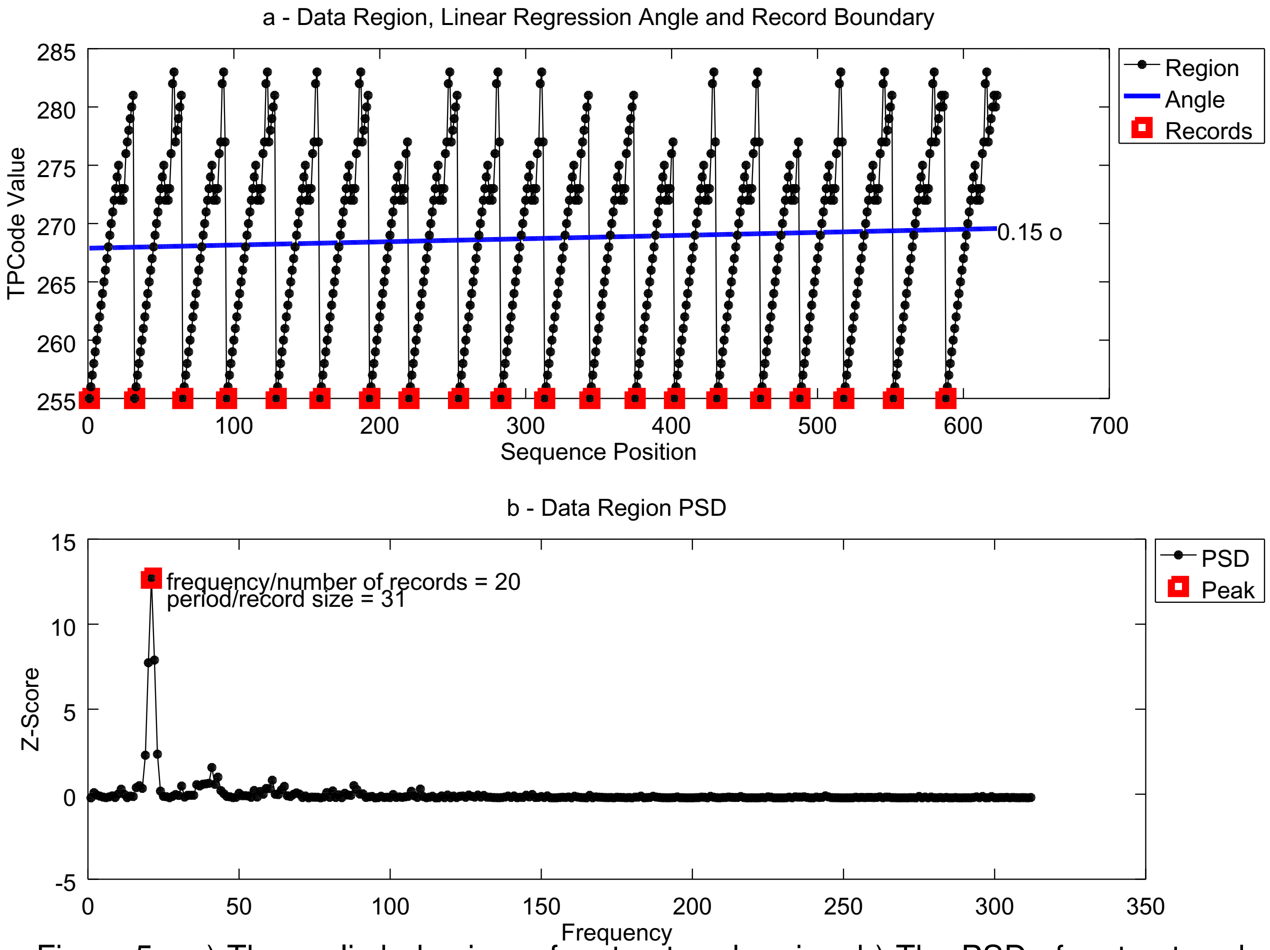


Figure 5 – a) The cyclic behaviour of a structured region; b) The PSD of a structured region.

To decide if a certain TPCode represents a record boundary within the region, we check how well distributed along the subsequence each TPCode is, in increasing order, using the coefficient of variation (CV for short – Definition 3.4) of the distance between its occurrences, e.g., if a given tag path occurs at positions {10, 20, 30}, the distances between consecutive occurrences are {20 – 10 = **10**, 30 – 20 = **10**} and the CV is equal to 0.0% since all distances are the same. In order to determine if a tag path is well distributed a threshold must be set. Since the regions are cyclic, the number of different tag paths that form the region's sequence is much smaller when compared to the size of the sequence. This is due to inherent repetitions and so it is not prohibitive to check the CV of every symbol in the region's alphabet. Figure 5a exhibits the record boundaries detected for that region (highlighted with squares). In this case the TPCode with value 255 matches the spectrum and CV constraints as we shall see. TPCode 255 occurs 20 times at positions {1, 32, 65, 95, 129, 159, 193, 220, 254, 283, 313, 344, 375, 402, 431, 461, 488 518, 552, 588} and the distance between adjacent occurrences are {31, 33, 30, 34, 30, 34, 27, 34, 29, 30, 31, 31, 27, 29, 30, 27, 30, 34, 36} so the average record size, for this TPCode, is 30.895, the standard deviation is 2.6435 and the CV is equal to 0.085564. If we validate this information against the region's spectrum we'll have a match because both the number of occurrences (frequency) and the average size (period) match the peak encountered in the region's spectrum at position 20 and the CV for this TPCode is only about 8.5564% (a low CV means little variation in record size).

Cross validating TPCode variation against the TPS spectrum is more robust than doing the inverse (i.e., selecting the highest peak in the spectrum and then searching for a corresponding TPCode). This is because the signal may be noisy and the spectrum may contain artifacts and errors due to windowing (e.g., spectral leakage) and frequency resolution limitation, and so it is not guaranteed that the highest peak will in fact correspond to the correct period. Figure 6 shows an example of this situation where the highest peak in the spectrum is not the correct period, but nonetheless there is a significant peak that corresponds to the correct period. Figure 6a shows a structured region with a cyclic behaviour, much like Figure 5a, except that, this time, the highest frequency peak in Figure 6b does not correspond to the correct period, but still, there is a significant frequency peak that matches the correct period and record count.

For those tag paths that are well distributed along the sequence, we search the corresponding range of the spectrum for a significant peak. If such a peak is found we subdivide the region accordingly, extract the records and end the process. To determine if a peak is significant we convert the spectrum to *z-score* (Definition 3.8) and set a threshold measured in number of standard deviations.

Algorithm 3 along with Figure 5 shows in more detail the process of identifying the records within a region.

Description of Algorithm 3: function `IdRecords` receives, as input, a structured region; it computes the PSD (Definition 3.6, Equation 3) of the input sequence using the FFT (Definition 3.7 – Line 3); iterates over the region's alphabet, in ascending order (Line 5); for each symbol in the alphabet its CV is calculated (Line 9), as defined in Definition 3.4, and checked against a threshold (Line 10); if it is bellow the threshold (meaning the current symbol is well distributed along the sequence) then the PSD is scanned in the range around the number of candidate records (Line 12 – `recCount` is the frequency we are validating against the sequence's spectrum) and; if a considerable peak is found in this range the function returns the positions of the current symbol (Line 13).

With respect to `idRecords`' time complexity, the overall complexity is dominated

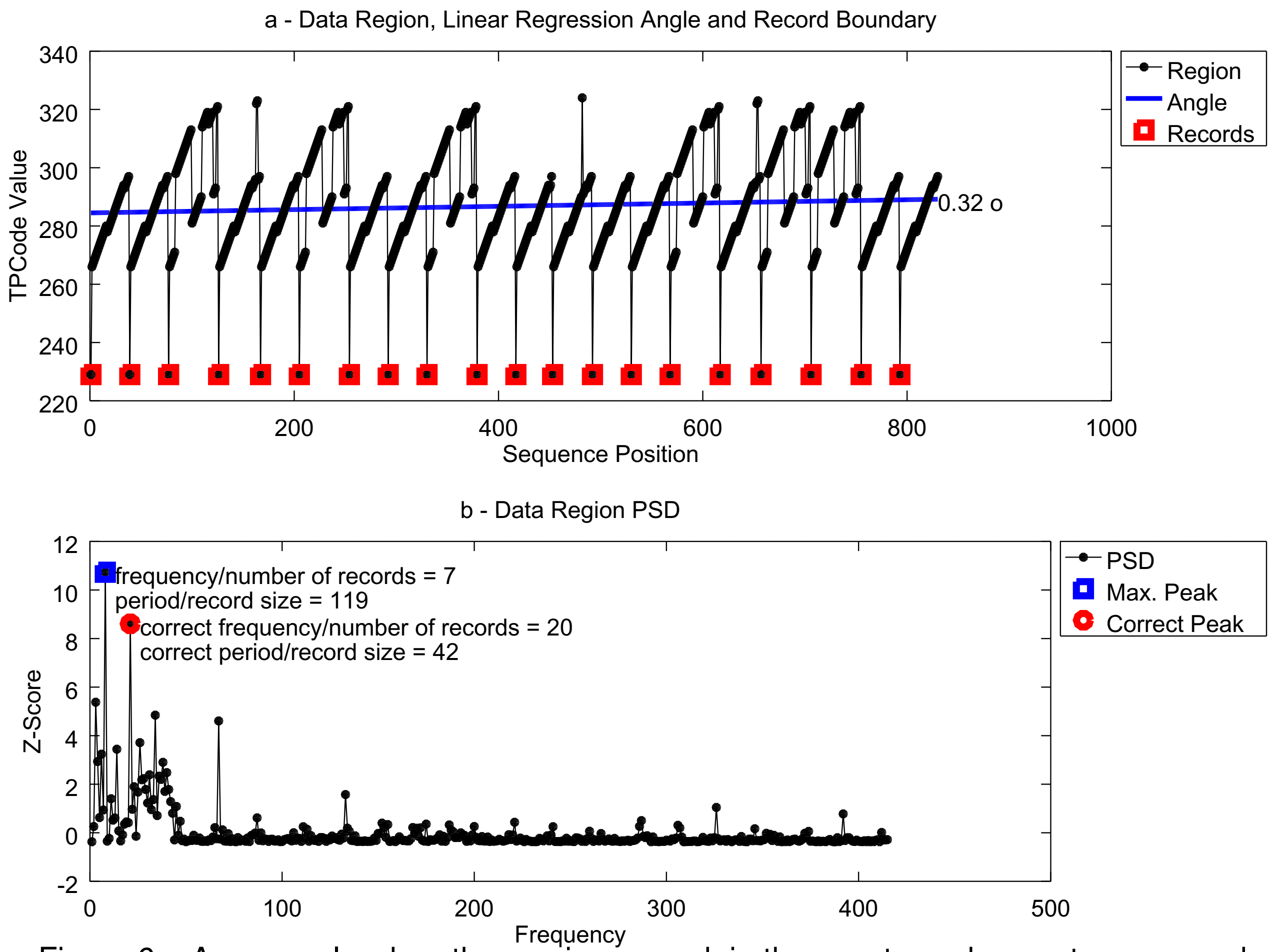


Figure 6 – An example when the maximum peak in the spectrum does not correspond to the correct period.

by Line 3, which is $O(nlogn)$ (due to FFT algorithm) on the size of the input region. The improvement of this time complexity is the subject of Section 4.2.

After this step we have, at our disposal, all the features used to classify structured data as content or noise. At this point we call the ML classifier (detailed in Section 4.3) to filter out noise data so this records are not forwarded for subsequent alignment.

4.1.4 Record Alignment

Once the records are identified they need to be correctly aligned (i.e., their fields must the matched in the best possible way) so they can be extracted in tabular form.

The problem of optimal multiple sequence alignment is known to be NP-Hard (ELIAS, 2006), so we are forced to adopt approximate solutions for this problem. One such solution, known as *Center Star* (GUSFIELD, 1993), has a polynomial time complexity and guaranteed error bound. This algorithm uses the edit distance to align two sequences and works by inserting "spaces" in the sequences when a misalignment is found. It also makes no assumptions about the data being aligned (i.e., it is general) and runs in time proportional to $O(k^2n^2)$, where k is the number of records to be aligned

Algorithm 3 Locates record boundaries in a region

Input: a region's TPS
Output: a set of record starting positions

```
 1: function IdRecords(region)
 2:     $signal \leftarrow region - \mu(region)$                    ▷ remove DC
 3:     $PSD \leftarrow abs(FFT(signal))^2$
 4:     $\Sigma \leftarrow sort(alphabet(region), ascending)$
 5:     for each $symbol$ in $\Sigma$ do
 6:         $recordPositions \leftarrow find(region == symbol)$
 7:         $recSize \leftarrow difference(recordPositions)$
 8:         $recCount \leftarrow length(recordPositions)$
 9:         $CV \leftarrow \frac{\sigma(recSize)}{\mu(recSize)}$
10:         if $CV < CVthreshold$ then
11:             $interval \leftarrow [recCount - 2..recCount + 2]$
12:             if $PSD[interval] > numSD \cdot \sigma(PSD)$ then
13:                 return $recordPositions$
14:             end if
15:         end if
16:     end for
17:     return $\emptyset$
18: end function
```

and *n* is the size of the records. The multiple alignment score (i.e., the sum of pairs edit distance) for the approximate solution is guaranteed to be at most a factor of 2 worst than the optimal solution's score and in practice can be better than this for similar strings, according to Gusfield (1993).

Table 3 shows the resulting alignment, using *Center Star* method, for the records extracted from the region depicted in Figure 5a and Table 18, in Appendix B, shows the actual data extracted (simplified). The values in the cells are TPCodes, so each represents a distinct node in the DOM tree and the empty cells are the spaces introduced by *Center Star* algorithm during sequence alignment.

The alignment of records, by itself, deserves a more in depth study. It is still an open, and very important, problem related to web record extraction. As observed in Elmeleegy et al. (2011), there is no single correct answer when it comes to record alignment and this constitutes an additional difficulty when evaluating alignment correctness.

It is important to note that the previous step of record identification is parameterized by the CV of record size and this parameter directly affects the quality of the alignment in the following way: the lower the CV the better will be the alignment at the expense of recall, i.e., we can filter out records too different in size (by setting a lower CV threshold) because such records are more prone to misalignment and that degrades the quality of the extraction. As stated in Gusfield (1993), similar sequences

Table 3 – *Center Star* record alignment for Figure 5a region

rec #	fields (TPCodes)														
#1	255	256	258	259	262	266	270	272	270	270	274				278
#2	255	256	258	259	262	266	270	272	270	270		280		274	278
#3	255	256	258	259	262	266	270	272	270	270	274	280	274		
#4	255	256	258	259	262	266	270	272	270	270	274	280		274	278
#5	255	256	258	259	262	266	270	272	270	270	274	280		274	
#6	255	256	258	259	262	266	270	272	270	270	274	280		274	278
#7	255	256	258	259	262	266	270	272	270	270	274				
#8	255	256	258	259	262	266	270	272	270	270	274	280		274	278
#9	255	256	258	259	262	266	270	272	270	270		280		274	
#10	255	256	258	259	262	266	270	272	270	270	274	280		274	
#11	255	256	258	259	262	266	270	272	270	270	274				278
#12	255	256	258	259	262	266	270	272	270	270	274				278
#13	255	256	258	259	262	266	270	272	270	270	274				
#14	255	256	258	259	262	266	270	272	270	270		280		274	
#15	255	256	258	259	262	266	270	272	270	270	274	280		274	
#16	255	256	258	259	262	266	270	272	270	270	274				
#17	255	256	258	259	262	266	270	272	270	270	274	280		274	
#18	255	256	258	259	262	266	270	272	270	270	274	280		274	278
#19	255	256	258	259	262	266	270	272	270	270	274	280	274	278	278
#20	255	256	258	259	262	266	270	272	270	270	274	280	274		

yield better alignment results. The structure of the records detected by our algorithm is necessarily similar, otherwise the data region would not be cyclic, but the size of the records can vary as a consequence of optional and disjoint fields that may be present. So, if we enforce a low CV we can also guarantee that the records will have similar size improving overall record similarity.

Another measure taken, in order to improve runtime and alignment quality, is to prune the records before aligning them. Since we are not constrained here by the hierarchy imposed by the DOM's tree structure, we can remove intermediate nodes that do not contain information, easing the alignment process and improving both its quality and running time (since the input size is diminished). For example, we can remove `<div>` nodes (which contain no information, but are part of the record structure) and align only the record's `text` nodes.

4.2 OPTIMIZING STRUCTURE DETECTION

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 transform, as explained in Subsection 4.1.3, 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 or not 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 5 we detected some regularity in $value = 255$, it is evenly distributed along the x-axis, and it occurs twenty times in the sequence, 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 $O(nlogn)$ 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 (using z-score from Definition 3.8). 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 7 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 than 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 6). 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 the PSD's average value (Equation 7). 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.

$$\sum |x[n]|^2 = \sum |X[f]|^2 \tag{6}$$

In Equation 6, $x[n]$ is the structured region (the signal) and $X[f]$ is the region's PSD.

$$E(PSD) = \frac{\sum |x[n]|^2}{N} \tag{7}$$

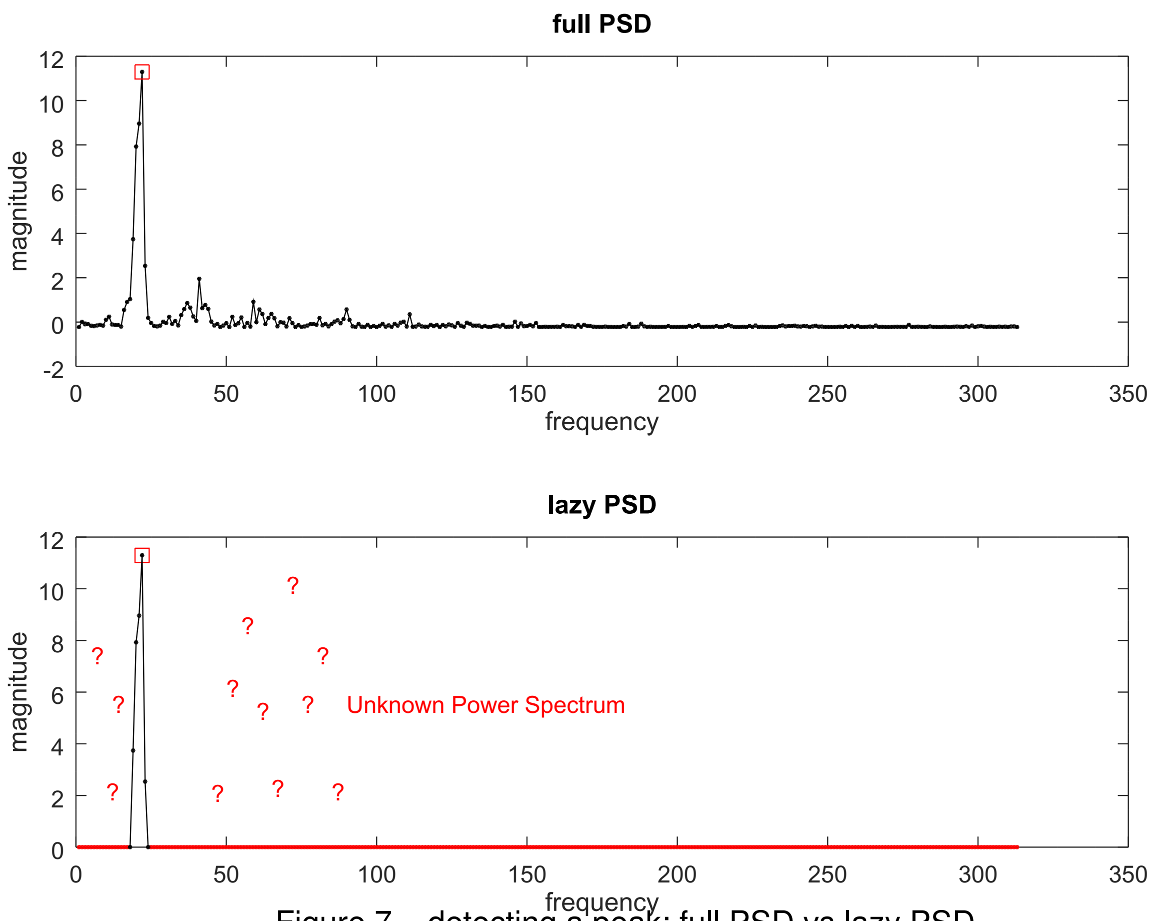


Figure 7 – detecting a peak: full PSD vs lazy PSD.

In Equation 7, $E(PSD)$ is the PSD's average value, $x[n]$ is the structured region and N is the region's length.

Putting it all together, we no longer have 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 2 and 3) as well as the PSD's average (using Equation 7). With this 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 Subsection 5.2.4 and in Figure 15). If we allow the number of computed coefficients to grow proportional to input size, then our time complexity would degrade to $O(n^2)$.

Conceptually there is no difference between the two approaches (full PSD vs single 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 proved 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 8, 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 pre-computed; lazy PSD construction takes $O(n)$ because that is where we compute the signal mean, using Equation 7, and looking up a coefficient is done in $O(n)$, using Equations 2 and 3.

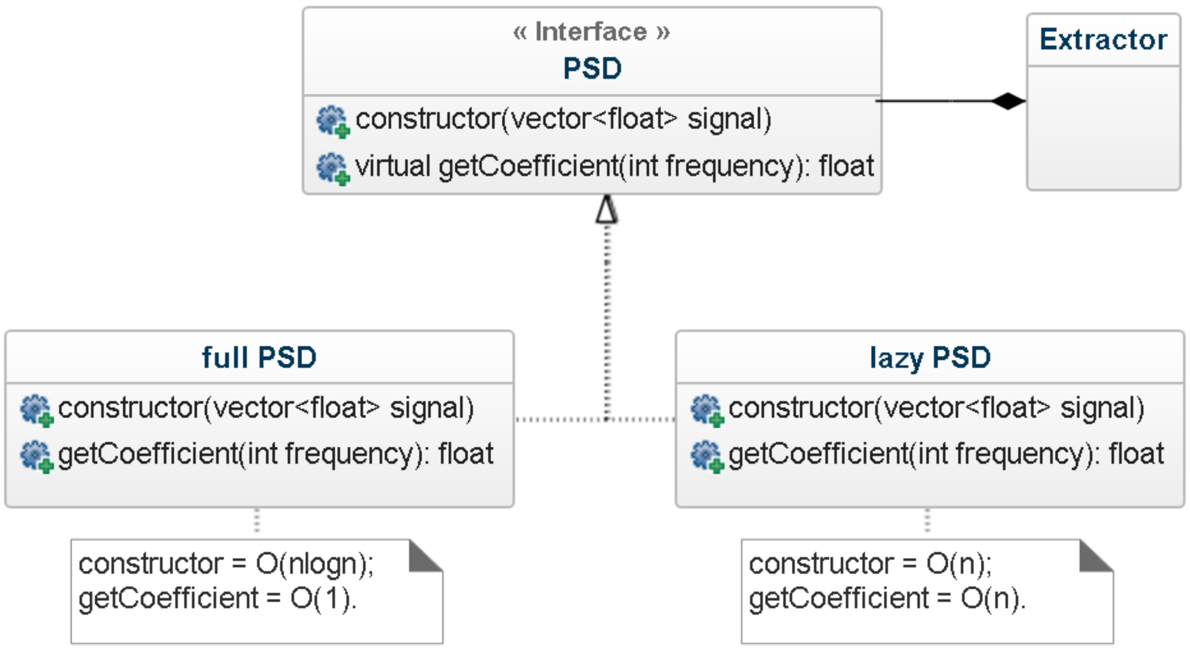

« Interface »
PSD
constructor(vector<float> signal)
virtual getCoefficient(int frequency): float
Extractor
full PSD
constructor(vector<float> signal)
getCoefficient(int frequency): float
lazy PSD
constructor(vector<float> signal)
getCoefficient(int frequency): float
constructor = O(nlogn);
getCoefficient = O(1).
constructor = O(n);
getCoefficient = O(n).

Figure 8 – 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 naïve Fourier transform algorithm (i.e., a direct implementation of Equation 2) 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 real time DTMF tone detection in telephone lines using low end microcontrollers that can not perform FFT in real time, but are capable of computing single coefficients in real time. 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.

4.3 CONTENT DETECTION

In order to distinguish which structured regions are content and which are noise we consider six region features, extracted from the document sequence, such as: region **size**, **center** position, **horizontal** position, **vertical** position, region **range** and **record** proportion (record count vs record size). All features refer to the document sequence, as opposed to the DOM tree as other works do. The region size and range are the size and range of the subsequence that represents a given structured region; all position features are relative to the document sequence and record proportion is retrieved from the spectral analysis of the region's subsequence.

These features were chosen because we believe (that is our hypothesis) they characterize the problem well and thus, can be helpful solving it. We purposefully have limited ourselves to **syntactical** features, as commented in Chapter 1, in order to keep our classifier as general as possible (i.e., independent from domain, content, semantics, etc.). As an extra, they can also be easily acquired from the extraction process. We will discuss each feature (size, positions, range and record proportion) in Subsections 4.3.1, 4.3.2, 4.3.3 and 4.3.4 respectively.

4.3.1 Size Feature

The region size feature is a real number, between 0 and 1, that represents the size of the region relative to the entire document, i.e., the percentage of the document occupied by the region.

The idea behind this feature is that if a web document was designed with the purpose of depicting a specific content, then this content (the reason the document was created in the first place) should occupy a considerable portion of the document. That is, our conjecture is that the likelihood of a region being content (and not noise) is directly proportional to its size.

Figure 9 shows an example where the entire document sequence has size equal to 1,336 and the main content region subsequence has size 618. The size feature in this case, using Equation 8, is equal to $\frac{618}{1{,}336} = 46.25\%$.

$$sizeFeat = \frac{regionSize}{sequenceSize} \tag{8}$$

4.3.2 Position Features

The region position features are actually comprised of three position features: center, horizontal and vertical positions. All three are real numbers between 0 and 1. The **center position** represents the distance from the center of the region to the center of the document; the **horizontal position** is the distance from the center of the region

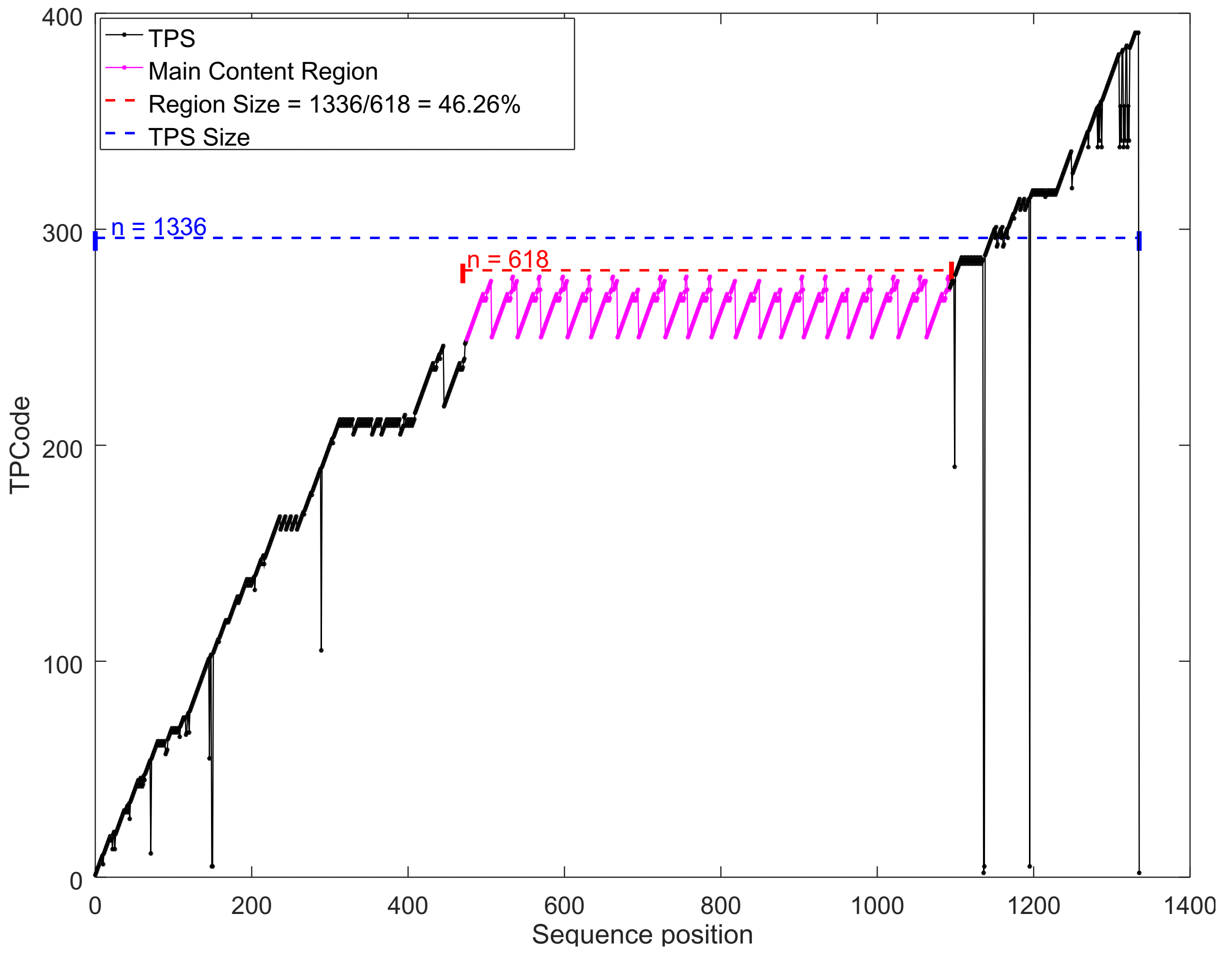


Figure 9 – Size feature example.

to the end of the document and; the **vertical position** is the distance from the vertical center of the region to the maximum value of the sequence.

With respect to the center position, the maximum possible distance is equal to half sequence size (e.g, when a region has size one and sits at the start/end of the document). The value of this feature is a percentage representing how close a region is from the center of the document (i.e., it is the distance complement). The rationale of our conjecture for this feature is similar to the size feature (Subsection 4.3.1): the closer a region is to the center of the document, the higher the probability it refers to real content.

Figure 10 shows an example where the document center (dark blue dashed line) is at position 667 (this is the maximum distance allowed) and main content region subsequence center (red dashed line) is at position 783, at a distance of 117 from document center (green dashed line). The value of this feature, using Equation 9, is equal to $1 - \frac{117}{667} = 82.46\%$

$$centerPositionFeat = 1 - \frac{|regionCenter - sequenceCenter|}{sequenceCenter} \quad (9)$$

With respect to the vertical and horizontal position, we believe they are needed

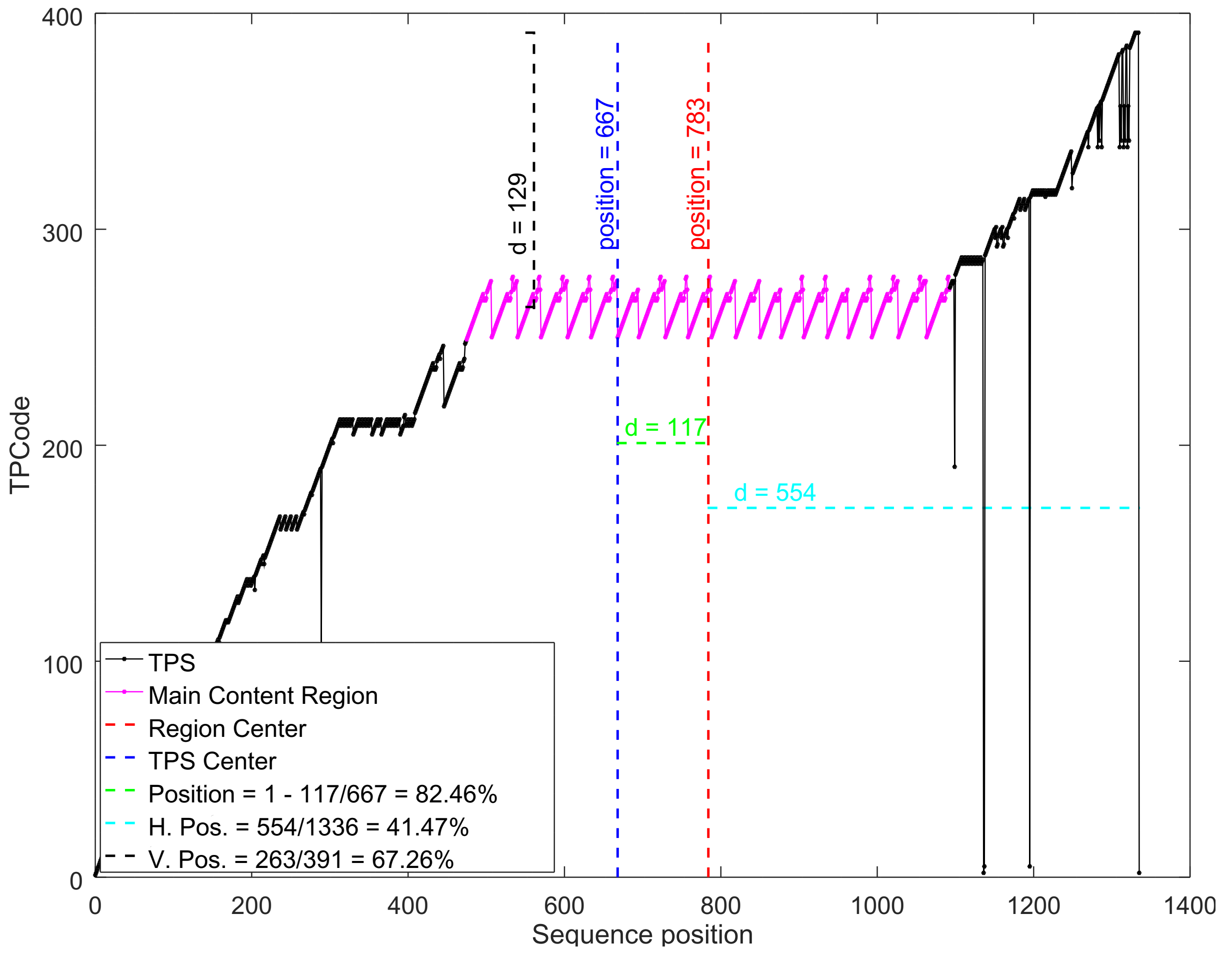


Figure 10 – Position feature example.

to provide a better indication of a region's position, **especially** when a document has no structured content (only structured noise), in this situation, **due to the absence of structured content**, a noise region will be closer to the center of the sequence and further away from its extremes. If we were concerned only about documents with structured content, these two features would, probably, be of little value to us.

Figure 10 shows how the horizontal and vertical positions are calculated. The horizontal position is the distance from the center of the region to the end of the sequence (light blue dashed line). Using Equation 11, the value of the horizontal position, in this example, is equal to $\frac{554}{1,336} = 41.47\%$.

$$horizPositionFeat = \frac{sequenceSize - regionCenter}{sequenceSize} \tag{10}$$

The vertical position (black dashed line) is the distance from the vertical center of the region (i.e., its average value) to the vertical end of the sequence (i.e., its maximum value). Using Equation 11, the value of the vertical position, in this example, is equal to $\frac{263}{391} = 67.26\%$.

$$vertPositionFeat = \frac{avg(region)}{max(sequence)} \tag{11}$$

Throughout this text we will refer to these three features simply as “center”, “horizontal” and “vertical” features.

4.3.3 Range Feature

The range feature is a real number, between 0 and 1, that represents the percentage of the region range relative to the entire sequence. It is orthogonal to the Size Feature (Subsection 4.3.1): it is vertical instead of horizontal. The region range is simply the maximum value found in the sequence (or subsequence) minus the minimum value. The full sequence range is equivalent to its maximum value.

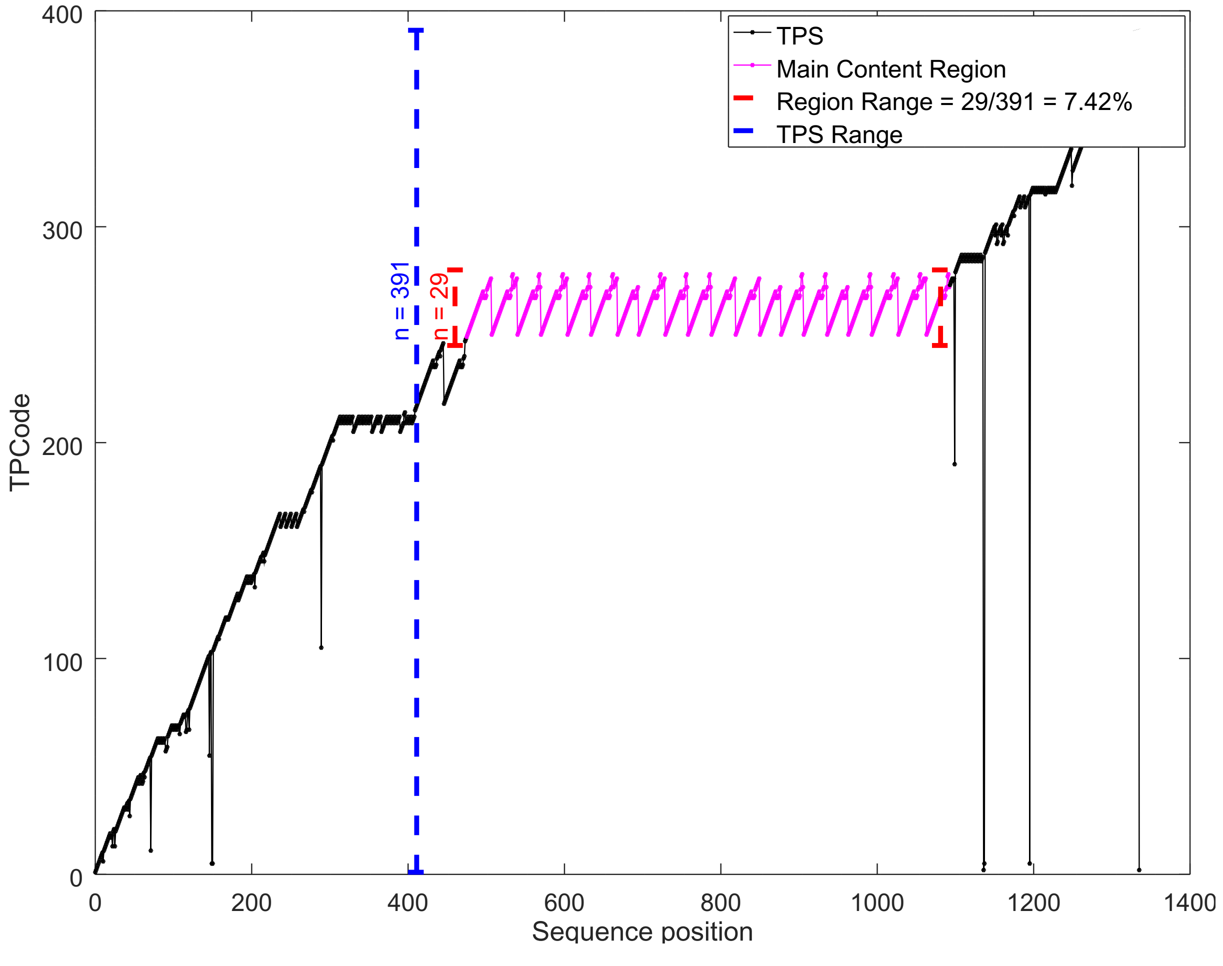


Figure 11 – Range feature example.

Figure 11 shows an example where region range is equal to 29 and document range is equal to 391. The value of this feature, using Equation 12, is equal to $\frac{283-254}{391} = \frac{29}{391} = 7.42\%$ of document range.

$$rangeFeat = \frac{regionRange}{max(sequence)} \tag{12}$$

4.3.4 Record Feature

We use the ratio between the number of records and their average size as a feature to indicate if a region is content or noise. We hypothesize that the lack of proportion[1] between these two measures (record count and record size) indicates noise and, conversely, the closer they are from one another the more likely the region is content. We calculate this value as shown in Equation 13.

$$recRatioFeat = \frac{min(numRecs, recCount)}{max(numRecs, recCount)} \tag{13}$$

The value of this feature is also a real number between 0 and 1, since the denominator in Equation 13 is always greater or equal to the numerator.

These two measures are obtained as documented in Subsection 4.1.3, using the region's power spectrum density (PSD) (OPPENHEIM et al., 1989). Figure 12b shows the PSD of the main content region in Figure 12a and its detected record count and average record size. The sequence in Figure 12a is the extracted main content region from the sequence depicted in Figure 2. In this example the number of records (represented by the red peak in Figure 12b) detected is 20 and their average size is 31. Therefore the value of the record feature, using Equation 13, is equal do $\frac{20}{31} = 64.51\%$. Throughout this text we will refer to this feature as "record" feature.

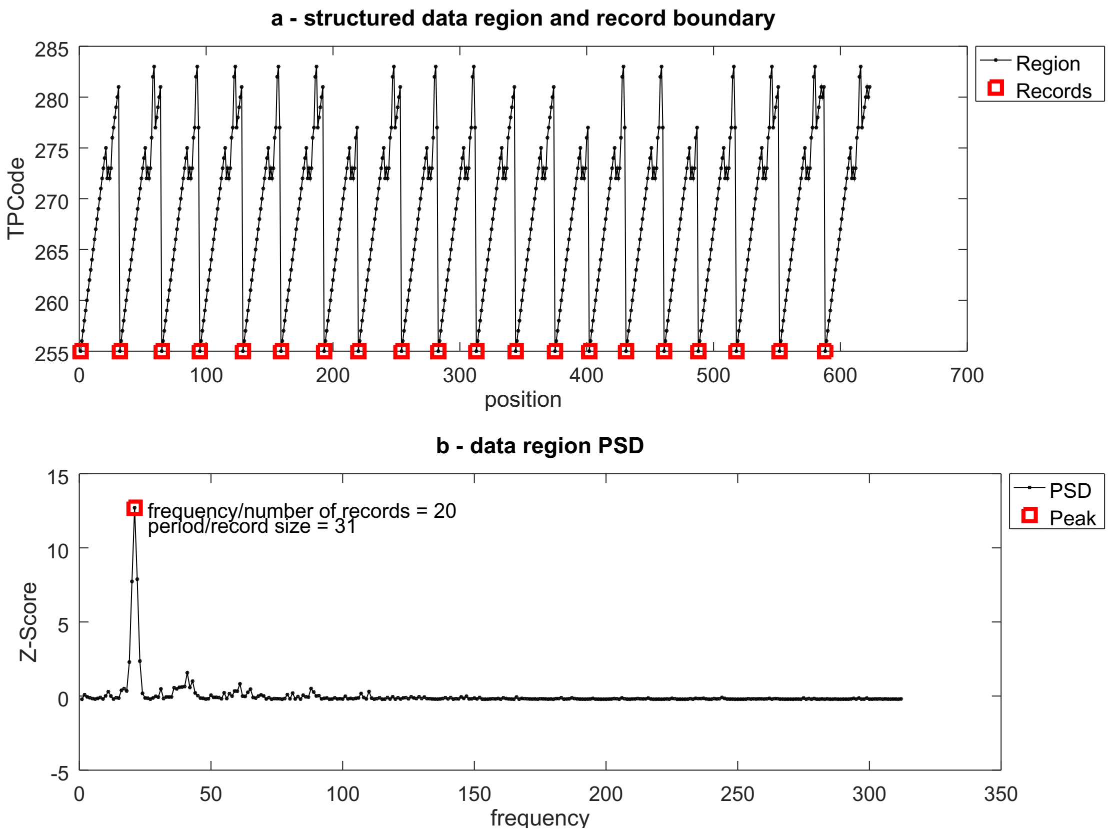


Figure 12 – Record count & size feature example.

[1] i.e., a lot of small records or few large records

5 EXPERIMENTAL EVALUATION

In this chapter we describe, in Section 5.1, our experimental setup, datasets and features used and, in Section 5.2, we show the analysis and results obtained from the extraction, classification and runtime experiments.

5.1 EXPERIMENTAL SETUP

In this chapter we present our experimental setup. In Subsection 5.1.1 we detail the datasets used in the experiments and; in Subsection 5.1.2 we detail how we have analyzed the features used to train the content classifier as well as the models themselves.

We have implemented our approach using a two tier software architecture where the core is coded in C++ and the presentation layer (graph plotting, result manipulation / display, logging, timing, etc.) in embedded Lua[1] or Python. For HTML parsing we have linked our system against libtidy[2]. Our implementation is publicly available at GitHub[3] and we encourage and support the scientific community to build on top of it, run independent experiments, implement other techniques for the sake of comparison and so on. In Appendix B there is a simple usage example of our framework.

5.1.1 Datasets

We used three datasets for comparison. One of them is commonly used for benchmarking and another was compiled by ourselves with pages collected from top largest internet companies[4] websites with more up to date templates from various domains (the documents were collected between 2017 and 2019). Dataset #1 was proposed by (YAMADA et al., 2004) and consists of 5 pages per site and 51 sites randomly sampled out of $114,540$ pages. This dataset was used, among others, by (LIU, B. et al., 2003; MIAO et al., 2009; GRIGALIS, 2013) which, in our opinion, are relevant works in the field, specially when we consider the requirements of being unsupervised, efficient, effective and HTML syntax independent. Dataset #2 was assembled and used in (GRIGALIS, 2013) and consists of 3 pages per site and 10 sites hand picked by the author. Dataset #3 (ours) consists of 266 HTML documents with structured content from various domains (news, banking, hotels, car rental, tickets, electronics), plus 61 documents **without** structured content, adding up to 327 documents. The documents without structured content were added to the dataset to investigate the behaviour of the ML content classifiers in the presence of this type of input. We use only one page per

[1] `https://www.lua.org/home.html` accessed 2020-10-06.
[2] `http://tidy.sourceforge.net/libintro.html` accessed 2020-10-06.
[3] `https://github.com/rpvelloso/webMining.git` accessed 2020-10-06.
[4] pages mostly drawn from sites owned by companies listed here: `https://en.wikipedia.org/wiki/List_of_largest_Internet_companies` accessed 2020-10-06.

site to avoid introducing bias towards specific sites and/or templates[5]. Table 4 presents summarized information about all datasets and Table 5 shows some details about our own dataset.

Table 4 – Summarized dataset information.

Dataset	#1	#2	#3
Sites	51	10	327
Pages per site	5	3	1
Average records	21	22	21
Total records	1052	218	7076

Table 5 – Dataset #3 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 acknowledge the fact that the size of our dataset is relatively small. The reason is that all regions, from every extracted document, have to be manually labeled as content or noise in order to be used in supervised machine learning. To compensate this limitation we kept the dataset as diverse as possible: every document comes from a different web site, with different template and content. All documents were collected from production web sites of real-world companies (e.g., Booking, Google, Amazon, Wikipedia, etc.). We believe that this diversity contributes to the overall representativeness of our dataset, making this study relevant. Also, all pages were collected recently, to guarantee they are all using modern and up to date templates.

5.1.2 Feature Analysis

In this section we detail the experiments we conducted using machine learning techniques with the features from Section 4.3. We have characterized the dataset used in the experiments with respect to its statistical properties, features correlation, etc. The objectives of this experiments are to determine the parameters and the subset of features which are important in this classification problem and measure the classification performance in terms of precision, recall, accuracy and F-Score.

We have considered, in our study, the following machine learning techniques: Gaussian Naive Bayes (GNB), Logistic Regression (LR), k Nearest Neighbours (kNN),

[5] this is possible because our extraction approach works with single page input

Gradient Boosting (GB), Extra-trees (EXT), Support Vector Machine (SVM), Voting Ensemble (VOT) and Stacking Ensemble (STCK). The voting and stacking ensembles are heterogeneous ensembles and are built from combinations of all the other models (GNB, LR, kNN, SVM, GB and EXT). These experiments were conducted using scikit-learn (PEDREGOSA et al., 2011) framework. For the gradient boosting we used XGBoost (CHEN; GUESTRIN, 2016) and for the ensembles we used MLxtend's (RASCHKA, 2018) implementation.

We have also evaluated the behavior of unsupervised machine learning, namely 1D optimal k-means clustering (WANG, H.; SONG, 2011), in a setting where all documents are known to contain structured content, so we can exploit this fact and leave supervised methods aside. We use the same features from Section 4.3 all combined into a single score. Since all features were built in such a way that the higher its value the greater the chance of being content and the lower the value the greater the chance of being noise, we can combine them all using simple multiplication so we end up with a single score directly proportional to the likelihood of being content (or inversely proportional to the likelihood of being noise).

We have conducted experiments to determine the best combination of parameters and features, within each approach, for solving the problem of distinguishing noise from content. To do so we ran a grid search for each algorithm with all feature combinations. We did so because the number of all possible combinations, for six features, is not prohibitive (only $2^6 - 1 = 63$ in total). After that we applied grid search, again, to select the best parameters for each algorithm.

5.2 RESULTS

Here we present our results. In Subsection 5.2.1, we show how the unsupervised content detection approach performed in a dataset where all documents contain structured content; in Subsection 5.2.2, we analyze how the supervised content detection approach performs and how it behaves in the presence of noisy documents; in Subsection 5.2.3, we show the results of our approach and compare it with other works and; in Subsection 5.2.4, we evaluate the runtime performance of our approach with the optimization proposed in Section 4.2

5.2.1 Unsupervised Content Detection

In this Subsection we show the results obtained in content detection using an unsupervised technique: 1D optimal k-means clustering(WANG, H.; SONG, 2011). We have combined all features into a single score because we are attributing meaning to it: content has a higher score than noise. This relative comparison, in a multidimensional space, becomes less clear. We have opted for this particular implementation, specifically, because it is deterministic (and, by consequence, reproducible) and it finds an optimal solution for the clustering problem.

We perform the clustering “locally”, that is, we cluster the scores of all structured regions within one document and we exploit the fact that the values of all the features we use are directly proportional to the likelihood of a region being content (that is the hypothesis on top of which we tailored each feature in Section 4.3). Based on this, we cluster the regions of a document into two groups, we discard the cluster with lower centroid (this is the noise cluster) and keep the other one (this is the content cluster).

By clustering the regions within a single document, as opposed to clustering the regions of a collection of documents, we keep our approach more scalable and easier to parallelize, since each document can be completely processed independently from the rest. The disadvantage of this method is that we can only use it in documents that we know, beforehand, to contain structured content. If we can not guarantee this, than we are better off using a supervised approach. If we process a document that has no structured content (only textual content and structured noise), some of its structured noise will be forcibly and erroneously tagged as content, considerably decreasing precision and thus rendering the extraction process useless.

In Table 6 we show the results we got from our unsupervised classifier. We can see that precision and recall are somewhat satisfactory (not excellent though), provided we guarantee the documents do contain at least one structured content region. We believe this scenario to be feasible so it is interesting to have an unsupervised alternative that works well for this task. Also, we can see a significant drop in precision when we add documents without structured content. This happens because structured noise gets classified as content, causing precision to degenerate rapidly. If we keep adding documents with unstructured content, the drop in precision only gets worst.

Table 6 – Results using k-means clustering in the structured documents from dataset #3.

Dataset	Precision	Recall	F-Score
Only Structured (dataset #3)	85.66%	84.35%	85.00%
Structured+Unstructured (full dataset #3)	72.94%	84.35%	78.23%

5.2.2 Supervised Content Detection

In this Subsection we analyze how the trained models perform at content detection and how they behave in the presence of noise. We also analyze feature statistics and their relative importance to this problem. For these results we have used our own dataset (see Tables 4 and 5) to generate training data for our models.

Figure 13 shows a scatter plot of each feature, separately, with respect to the target class (content vs noise), it gives a rough idea of how content and noise are intertwined within each feature. We also provide pair plots in Appendix C to help visualizing our dataset, particularly the kernel density estimation of content versus noise, in Figure 18, which depicts a clear distinction between both classes. Table 7 shows the features relative importance according to two different criteria (ANOVA and χ^2), both yielding the same results. Table 9 shows mean, coefficient of variation (CV), skewness and kurtosis for all features with respect to the target class. Table 8 shows the correlation between all features. We see that “size” vs “position”, “size” vs “range” and “position” vs “range” have a stronger correlation compared to others, this fact reflected in feature selection for some models and these same three features (size, position and range), according to Table 7, are also the most important ones.

Table 7 – Feature importance (vs class)

feature	χ^2	ANOVA
Size	51.6426225	487.50116318
Center	25.8025951	260.93679423
Range	23.1719961	232.44608713
Record	4.71168623	26.59572956
Vertical	1.16942710	12.38250104
Horizontal	0.433793117	3.63505065

Table 8 – Feature correlation

	Size	Record	Range	Horiz.	Vert.
Content & Noise					
Center	**0.63**	0.08	0.45	-0.11	0.01
Size		0.08	**0.68**	-0.02	0.11
Record			0.08	0.04	0.03
Range				-0.01	0.08
Horizontal					**0.85**
Content					
Center	0.58	-0.11	0.21	-0.37	-0.07
Size		-0.10	0.53	-0.12	0.12
Record			-0.04	0.04	-0.04
Range				-0.03	0.08
Horizontal					0.65
Noise					
Center	0.25	-0.01	0.22	-0.15	-0.11
Size		-0.12	0.39	-0.15	-0.11
Record			-0.08	0.01	0.01
Range				-0.13	-0.09
Horizontal					0.89

We have used 5-fold cross-validation to evaluate the performance of each model with respect to precision, recall, accuracy and F-Score. The average result of 200 runs is shown in Tables 10 and 11.

Table 9 – Feature statistics

Feature	Mean	CV	Skewness	Kurtosis
Content & Noise				
Size	0.27	0.87	0.92	2.90
Center	0.57	0.51	-0.18	1.71
Range	0.11	1.12	1.98	7.46
Record	0.40	0.68	0.52	2.15
Vertical	0.59	0.40	-0.49	2.42
Horizontal	0.55	0.47	-0.26	2.14
Content				
Size	0.44	0.49	0.28	2.47
Center	0.74	0.27	-0.81	2.95
Range	0.19	0.74	1.49	5.51
Record	0.46	0.61	0.22	1.89
Vertical	0.63	0.24	-0.80	3.58
Horizontal	0.57	0.26	-0.45	3.63
Noise				
Size	0.12	0.94	2.32	9.37
Center	0.41	0.65	0.58	2.16
Range	0.05	1.40	4.37	26.83
Record	0.34	0.74	0.81	2.74
Vertical	0.56	0.53	-0.15	1.69
Horizontal	0.53	0.61	-0.06	1.44

When we omit the documents without structured content we get the results shown in Table 10. The model with the best performance, in our experiments, was the Logistic Regression (LR), with 93.57% F-Score. In this application we should prioritize precision (the web is vast and full of noise), with that in mind kNN performed a little better, with almost 94% precision. So, in a controlled environment, where we can guarantee the input will always contain structured content, the features we elected were enough to achieve very good results with relatively simple models (kNN and LR). There is, probably, no need to use more elaborate approaches and/or ensembles in this setting.

Table 10 – Results using dataset containing only structured content documents.

Model	Precision	Recall	Accuracy	F-Score
LR	93.30%	**93.85**%	**93.02**%	**93.57**%
GNB	91.97%	90.55%	90.62%	91.26%
kNN	**93.83**%	92.21%	92.59%	93.01%
SVM	93.60%	92.20%	92.37%	92.90%
EXT	91.88%	91.87%	91.23%	91.88%
GB	90.75%	90.14%	89.74%	90.44%
VOT	92.41%	92.20%	91.71%	92.31%
STCK	92.97%	92.20%	92.06%	92.59%

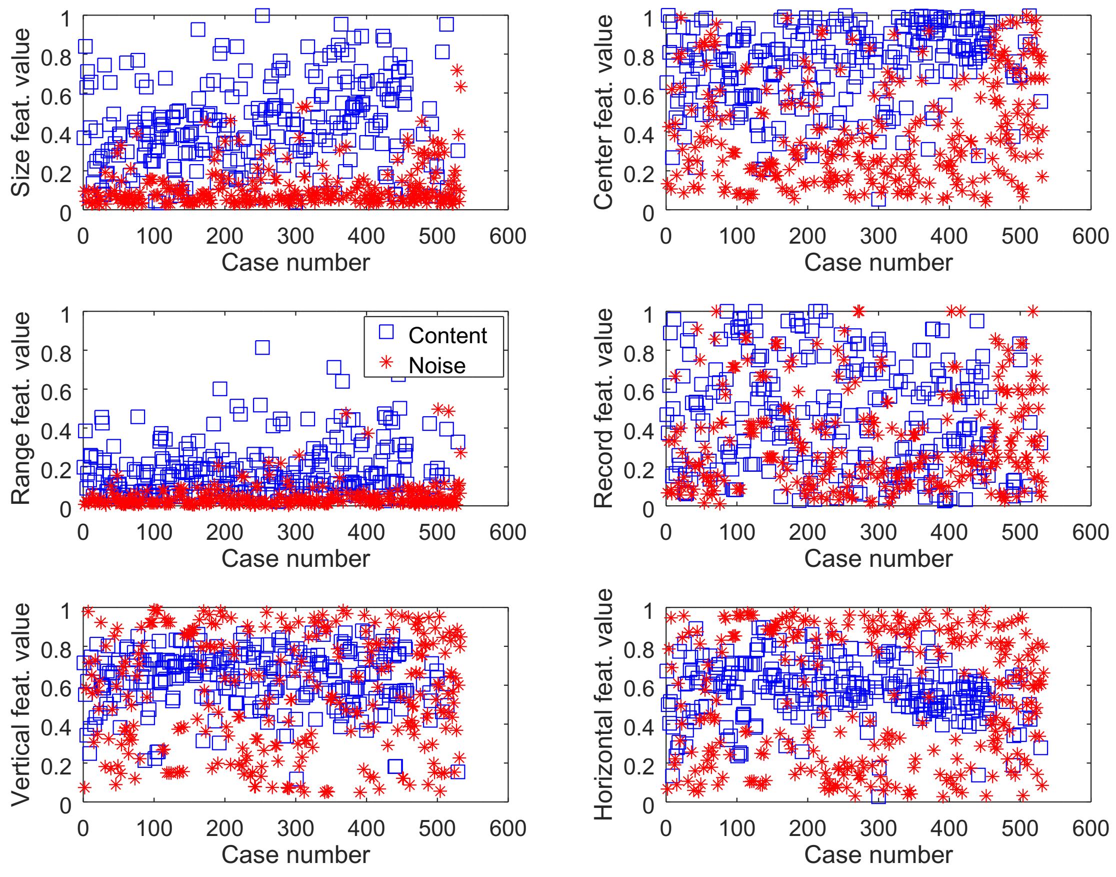


Figure 13 – Input dataset features: content vs noise.

When we consider the full dataset, including the documents without structured content, we get the results shown in Table 11. As expected there is a drop in F-Score (column "Drop" in Table 11). The amount of unstructured documents corresponds, roughly, to 18% of the entire dataset (see Table 5) and yet we were able attain negative variations in F-Score lower than 1%, for this reason we consider this results to be very significant as they show it is possible, using the proposed approach, to identify noise no matter the content is structured or not. The most prominent drop, ironically, occurs with Logistic Regression (which performed best with only structured content documents). Gradient Boosting (GB), although not the best performing model in either setting, showed the lowest impact in F-Score. Another interesting result we see is that all ensembles (VOT, STCK and GB) have relatively low drop in F-Score (the lowest ones, in fact), with the exception of ExtraTrees (EXT) ensemble, which is the second largest. For this setting, where we have no guarantee that the documents have structured content (only that they may have structured noise), the best option would be the Voting heterogeneous ensemble with F-Score of 91.47% and largest precision (above 90%).

For the sake of precision score we have investigated the false positives and concluded that they are borderline cases, i.e., they are structured noise regions that we can not clearly classify, even when we visually inspect the document's TPS (without considering semantics, of course). These region's features are in a gray area, half way

Table 11 – Results using complete dataset (including unstructured documents).

Model	Precision	Recall	Accuracy	F-score	Drop
LR	87.97%	92.13%	89.45%	90.00%	**-3.57**%
GNB	89.69%	88.99%	89.47%	89.34%	-1.92%
kNN	89.72%	90.56%	90.02%	90.14%	-2.87%
SVM	89.51%	92.12%	90.77%	90.79%	-2.11%
EXT	88.80%	88.43%	88.71%	88.61%	**-3.27**%
GB	90.13%	88.97%	89.65%	89.55%	**-0.89**%
VOT	**90.45**%	**92.52**%	**91.33**%	**91.47**%	-1.38%
STCK	89.93%	91.81%	90.73%	90.86%	-1.73%

between content and noise. Fortunately, these cases seem to be a minority. The hyperparameters used in every model are documented in Appendix A, these parameters were found using grid search.

Table 12 – Best set of features p/algorithm.

Model	Size	Center	Range	Record	Ver.	Hor.
LR	✓	✗	✓	✓	✗	✓
GNB	✓	✓	✓	✓	✓	✓
EXT	✓	✓	✓	✓	✓	✓
SVM	✓	✗	✓	✓	✓	✓
kNN	✓	✓	✓	✓	✓	✓
GB	✓	✓	✓	✓	✓	✓

We show in Table 12 the best performing features for each model, for the sake of documentation and reproducibility. Almost all models used all features, this shows that all features are relevant to the problem and somehow contribute to the solution. The exceptions are the Logistic Regression and Support Vector Machine. The Logistic Regression achieved the best results without “center” and “vertical” features, and SVM without “center” feature, that is probably due to the high correlation with other features as shown in Table 8: “center” feature has a considerable correlation with “size” and “vertical” feature with “horizontal”.

5.2.3 Record Extraction

In this section, we discuss and compare the results of our research with other approaches found in the literature that we considered to be of relevance. The proposals we considered most relevant are those which are fully automatic, highly effective (i.e., high f-score), domain independent and computationally efficient. We have used in our comparisons published results since we were not given access to the authors' implementation. We are using, for comparisons, only the subset of structured content documents, since we are comparing with other approaches that are also using only structured content documents.

Table 13 – Results comparison using the dataset #1.

	Precision	Recall	F-Score
MDR(LIU, B. et al., 2003)	59.80%	61.80%	60.78%
TPC(MIAO et al., 2009)	90.40%	93.10%	91.73%
ClustVX(GRIGALIS, 2013)	99.81%	99.52%	99.66%
Ours (Unsupervised)	92.02%	94.11%	93.05%
Ours (LR)	95.45%	95.45%	95.45%
Ours (VOT)	93.02%	90.91%	91.95%

Table 13 compares precision, recall and f-score of our approach with three other state-of-the-art techniques, using dataset #1: MDR (LIU, B. et al., 2003), TPC (MIAO et al., 2009) and ClustVX (GRIGALIS, 2013). Our approach performs better than MDR and TPC and slightly worse than ClustVX, but on the other hand our technique does not need to render the entire page, as ClustVX does, in order to extract the records and we show how rendering impacts computational cost in Figure 14. As a matter of fact, our approach just parses the HTML and do not even process CSS definitions (all we care about is the `class` attributes in each tag). We have considered MDR to be of relevance here because it is one of the few approaches with available implementation and probably the first to achieve acceptable results (both in computational complexity and effectiveness). The experimental results were manually inspected, one by one, to check if the final alignment was acceptable (we use the word "acceptable" here, because there can be more than one correct alignment).

Table 14 – Results comparison using dataset #2.

	Precision	Recall	F-Score
ClustVX(GRIGALIS, 2013)	100.00%	99.50%	99.80%
Ours	100.00%	97.68%	98.83%

Table 14 compares only against ClustVX, using dataset #2. Again we have performed slightly worse, but close to a perfect result. This dataset is more up to date than dataset #1, with respect to document template and web programming practices, although it is rather small with only 10 sites.

Table 15 – Precision, recall and F-Score for dataset #3.

Precision	Recall	F-Score
93.30%	93.85%	93.57%

Table 15 presents the results obtained over our own dataset, assembled with documents from large internet companies' websites using modern and up to date layouts from various distinct domains, including: travel (e.g., booking.com, tripadvisor.com), real estate (e.g., airbnb.com), car rental and sales (e.g., rentalcars.com), search engines (e.g., google.com, yahoo.com), videos (e.g., youtube.com), movies (e.g., imdb.com), music (e.g., itunes.com), retail sales (e.g., aliexpress.com, walmart.com, amazon.com) and social networks (e.g., facebook.com, linkedin.com). The overall quality is pretty good with a precision of 93.30%, recall of 93.85% yielding an f-score of 93.57%. This high precision is a key factor when dealing with web scale data and it is quite an achievement when we consider data diversity and the fact that our approach is both automatic and computationally efficient, such attributes are necessary for feasibility.

We have mentioned that an important requirement when dealing with web scale is efficiency. On that subject ClustVX fails to deliver for two reasons: (1) it employs a browser engine to render the entire web page and issues several calls to the browser API during extraction and; (2) it clusters DOM tree nodes with similar "Xstrings" (a variation of a tag path) probably using edit distance (the paper is not clear about this issue) which has quadratic time complexity. Anyway, we chose to compare against this approach mostly because the reported results are near perfect and we wanted to show that our proposal achieves very similar results without incurring in unnecessary computational cost. To support our claim about the computational cost involved in document rendering, we have compared running time of our approach using an internal HTML parser versus a browser engine. The browser engine is controlled via *WebDriver*[6] interface in offline mode (to avoid computing networking latency as runtime).

Table 16 – Benchmarking for dataset #3.

	render	no render.	variation
total runtime (in secs)	89.871s	41.279s	217.72%
avg. runtime per page	0.281s	0.129s	
total size (# of nodes)	642,264	634,428	98.78%
avg. size per page	2,007	1,982	

We present in Figure 15 the running time behaviour of our implementation. We can see that the graph suggests a $O(n)$ time complexity with a coefficient of determination[7] of 85.75%. This behaviour is consistent with our complexity analysis. The alignment phases, although it has a greater time complexity, does not impact (asymptotically) the overall time complexity because their input size, compare to overall input size, is much smaller. Anyway, the alignment phase could be considered as a limiting factor/bottle neck of our approach when dealing with long records and/or a large number of data regions, but that is not the average behaviour we observed in practice.

Figure 14 also shows the running time comparison when using a browser engine to render the entire document. Our approach does not need the rendered pages in order to extract the records, we just did this comparison to illustrate the huge difference in running time to other approaches that do need to render the document. The processing

[6] `https://www.w3.org/TR/webdriver/`

[7] the coefficient of determination (R^2) measures how much data is explained by the fitted curve.

time is more than two times greater (89.871s rendering vs 41.279s no rendering), the change in input size (the browser engine generates a slightly different DOM tree than our built in HTML parser) is marginal, only 1.22% larger, and the achieved results are the same (precision, recall and f-score) as shown in Tables 15 and 16. In Figure 14 we can see, as well, both asymptotic behaviours. We can see the rendering running time adds a much greater constant factor to the whole process. In practice an approach like ClustVX, that needs to render the page in order to extract the records, incurs in a very large overhead (two times slower than our approach, in average) to obtain an f-score gain of only 4.21% (in dataset #1) and 0.97% (in dataset #2) over our approach. Put another way, let's assume we have to process an entire search engine index containing a sizable portion of all indexable content on the internet, would this trade-off be worth it?

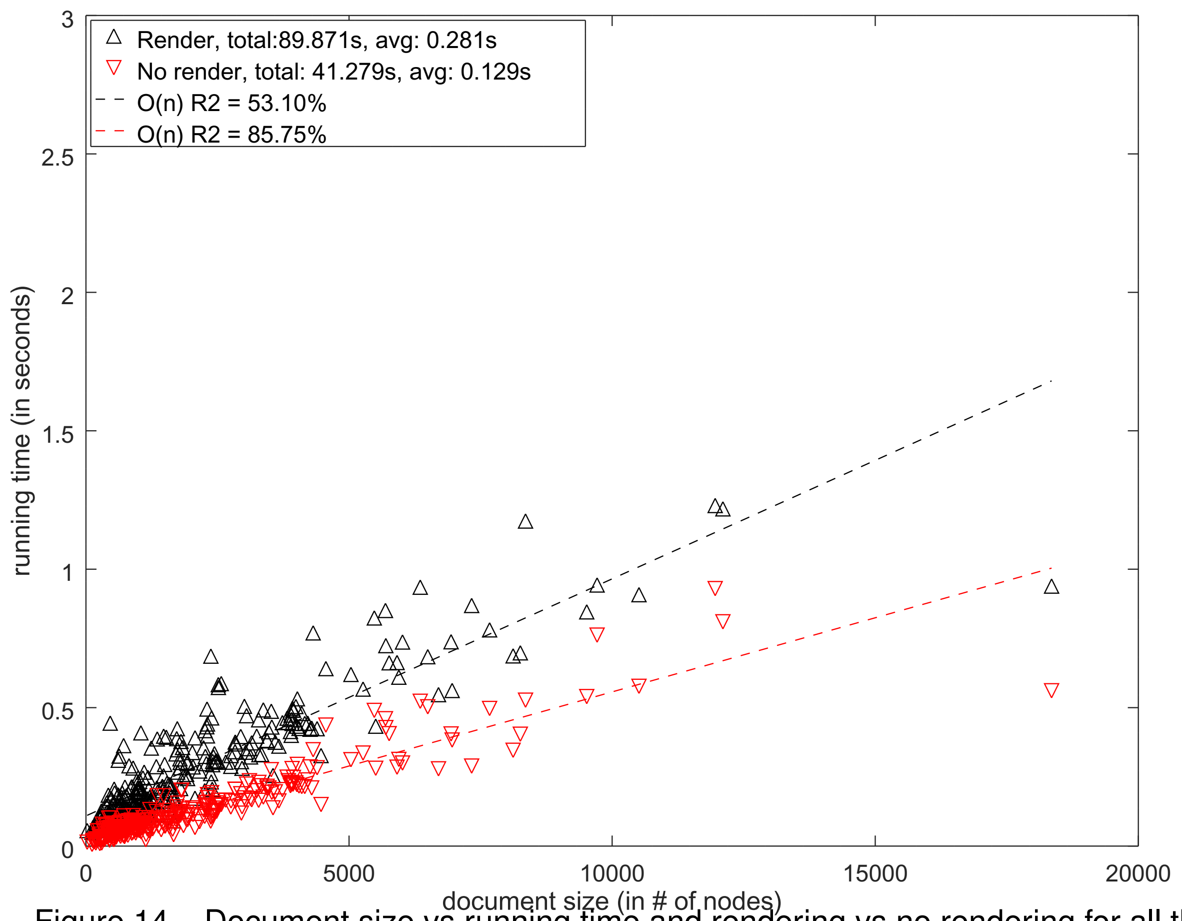


Figure 14 – Document size vs running time and rendering vs no rendering for all three datasets.

5.2.4 Optimization

We have outlined in Section 4.2 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.

In Figure 15 we have both approaches compared: 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.

We can see a 11.77% improvement in runtime when using lazy PSD, compared to full PSD computation: full PSD takes 46.788s to process our 327 documents dataset and lazy PSD takes 41.279s. The linear regression (dashed lines in the graph) is also adherent to the lazy PSD runtime (R^2 = 85.75%), 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.

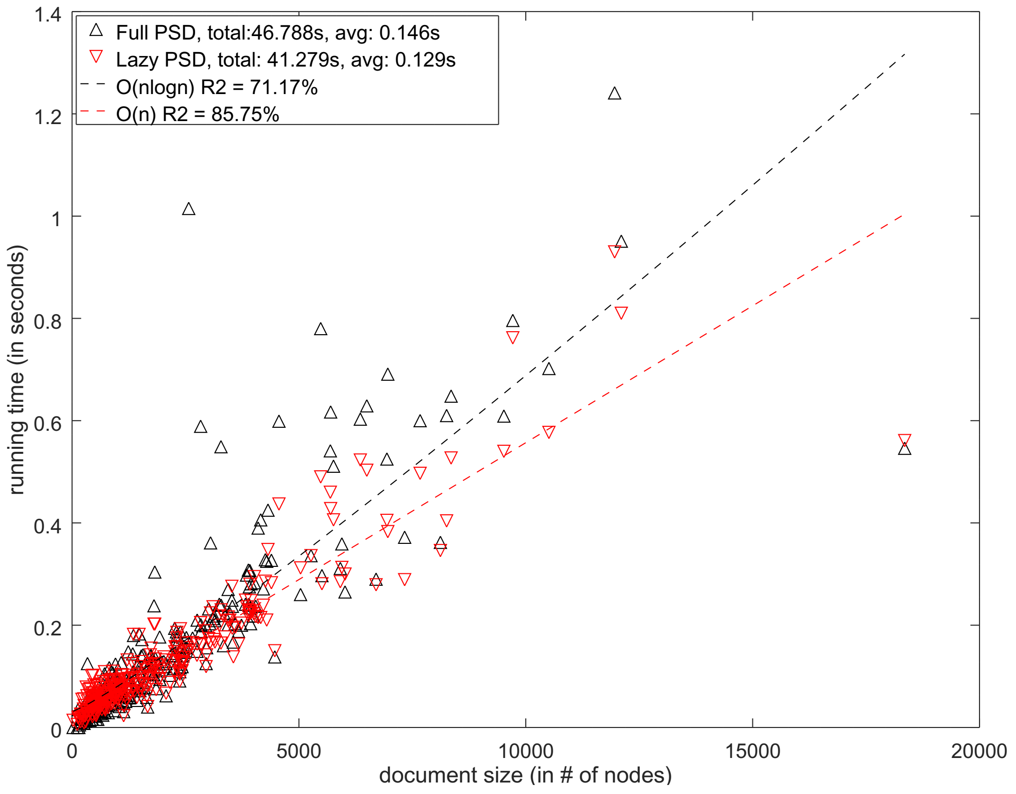


Figure 15 – Runtime improvement with lazy PSD.

The qualitative results presented in Subsection 5.2.3 have not changed when we applied the optimization proposed here, 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.

6 CONCLUSION

We have shown an approach for the problem of extracting records from semi-structured web pages based on the following insight: seeing the structure of a document as a cyclic signal. Our results are superior to the state-of-the-art, but we claim we have achieved it in a more efficient way as documented in Section 5.2. The structure detection in our approach can also be used in other areas, for example, when the target is unstructured data one can rule out structured noise (e.g., menus and advertising). In our work we have tried to exploit, as much as possible, the syntactical characteristics of semi-structured data, without resorting to more costly approaches (e.g., semantics, rendering, etc.). However, we believe that such techniques could be combined to improve the results even further.

Although the high efficiency and effectiveness of our technique are demonstrated by the results in Section 5.2, it is not an error free approach. We have assumed the records in a web document are contiguous, consistently formatted / structured and with approximate size. Although these assumptions are plausible, they are not universal and where they do not hold our approach fails. Nonetheless, when considering web scale data, this subset of all structured data is of considerable size and value.

In our research, through observation, we have come up with the conjectures depicted here, for each region feature and, through experimentation, we have confirmed these conjectures in two different situations: in a controlled setting (with 93.57% F-Score using Logistic Regression) and; in an open environment (with 91.47% F-Score using a heterogeneous voting ensemble). We believe these to be very good results, especially considering we are using only very basic information (size, position, etc.) to distinguish between content and noise and a direct ML approach. We have also demonstrated the relevance of these features to the problem by testing every possible combination of them.

In our evaluation we have successfully achieved our goals:

1. **efficient** extraction of structured data through the use of signal processing techniques: $O(n)$;
2. **effective** extraction of structured data through the use of signal processing techniques;
3. **effective** content classification through the use of machine learning techniques: 93.57% and 91.47% f-score.

Nonetheless, there is always room for improvements. Adding other features to the problem, perhaps using semantic features combined with the ones proposed here could yield some interesting results, especially when we consider borderline false positives where semantics could help improve precision even further. With an increased number of features though, testing all combinations becomes prohibitive, other approaches should then be employed (e.g., genetic algorithms) to find a good (maybe the best) combination of features. About the dataset size, more documents should be gathered, in the future, to improve confidence on our analysis and results, especially the relative performance of various models tested here.

In our opinion the major set backs in this research area are the unavailability of implementations and the absence of a common framework. Without the implementation of a particular technique we can not analyze individual results to gain insight about why it did or did not work in a particular case and we are unable to make a deeper and independent comparative analysis of published results. This kind of analysis speeds up research because it helps refine research assumptions faster, using previous works. Had implementations been available one could compare specific situations where one approach performs better than the other, otherwise we can only compare against published results (usually summarized precision and recall). Furthermore, implementing approaches proposed by others is also a complex matter. For starters, most papers do not provide full details and there is no guarantee that an independent implementation will match exactly the one used by the authors in their experiments.

On the other hand a common framework, for implementing extractors, reduces the variables involved in a comparison by standardizing the basic software infrastructure (e.g., same HTML parser used by everyone, as well as rendering engine and test beds). These problems were pointed out in Sleiman and Corchuelo (2012) and Ferrara et al. (2014) and would be mitigated by such a framework.

Our guidelines for future work are the following:

1. research alignment techniques: here we have used a general purpose and supra-linear (but still polynomial) sequence alignment algorithm, but we believe it is possible to devise a solution specifically for the problem of record alignment that is more effective and has a lower complexity bound. In Gfrerer et al. (2017) a parallel approach is proposed;

2. standardize datasets: the ideal test bed should consist of several randomly taken samples from a large scale search engine index with annotated ground truth. Such a dataset would enable the researcher to draw consistent conclusions from statistics (e.g., percentage of documents with structured content, “real” precision and recall). We haven’t found such a dataset in the literature (i.e., one that is annotated for record extraction). With reliable statistics the amount of this kind of structured information could be correctly estimated, similar to what was done in Michael J Cafarella et al. (2008a);

3. investigate the use of Lempel-Ziv (ZIV; LEMPEL, 1977) (LZ) in the context of pattern detection: LZ is a computationally efficient compression algorithm targeted

specifically at sequences, it works by looking for redundancies in the sequence, the redundancy (i.e., repeated data) can be seen as a recurring pattern (like contiguous records). We believe it is possible to exploit this algorithm in this context;

4. try out synthesizing training data and see if that improves content classifier's performance;
5. investigate the use of convolutional neural networks (CNNs) on this problem. We could, for example, try to detect the number of records (training a regressor) by feeding the TPS to a CNN and see if it can extract relevant features from the document sequence;
6. investigate if the document sequence can be treated and processed as a time-series with specific algorithms;
7. experiment with classification using other features of different nature (e.g., content and semantic features);
8. develop the extraction framework further: continue the development of the extraction framework, refine and refactor code, implement other extraction techniques, integrate a crawler, etc.

About the time complexity 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.

REFERENCES

ABADI, Daniel et al. The Beckman report on database research. **Communications of the ACM**, ACM, v. 59, n. 2, p. 92–99, 2016.

ARASU, Arvind; GARCIA-MOLINA, Hector. Extracting structured data from web pages. In: ACM. PROCEEDINGS of the 2003 ACM SIGMOD international conference on Management of data. [S.l.: s.n.], 2003. P. 337–348.

BALAKRISHNAN, Sreeram et al. Applying WebTables in Practice. In: CIDR. [S.l.: s.n.], 2015.

CAFARELLA, Michael J. **ACSDb Download**. [S.l.: s.n.], 2012. http://web.eecs.umich.edu/~michjc/data/acsdb.html.

CAFARELLA, Michael J; HALEVY, Alon; WANG, Daisy Zhe; WU, Eugene; ZHANG, Yang. Webtables: exploring the power of tables on the web. **Proceedings of the VLDB Endowment**, VLDB Endowment, v. 1, n. 1, p. 538–549, 2008a.

CAFARELLA, Michael J; HALEVY, Alon Y; ZHANG, Yang; WANG, Daisy Zhe; WU, Eugene. Uncovering the Relational Web. In: WEBDB. [S.l.: s.n.], 2008b.

CAFARELLA, Michael J.; HALEVY, Alon; MADHAVAN, Jayant. Structured Data on the Web. **Commun. ACM**, ACM, New York, NY, USA, v. 54, n. 2, p. 72–79, Feb. 2011. ISSN 0001-0782. DOI: 10.1145/1897816.1897839. Available from: http://doi.acm.org/10.1145/1897816.1897839.

CAI, Deng; YU, Shipeng; WEN, Ji-Rong; MA, Wei-Ying. Extracting content structure for web pages based on visual representation. In: SPRINGER. ASIA-PACIFIC Web Conference. [S.l.: s.n.], 2003. P. 406–417.

CHEN, Tianqi; GUESTRIN, Carlos. Xgboost: A scalable tree boosting system. In: ACM. PROCEEDINGS of the 22nd acm sigkdd international conference on knowledge discovery and data mining. [S.l.: s.n.], 2016. P. 785–794.

CHU, Xu; HE, Yeye; CHAKRABARTI, Kaushik; GANJAM, Kris. Tegra: Table extraction by global record alignment. In: ACM. PROCEEDINGS of the 2015 ACM SIGMOD International Conference on Management of Data. [S.l.: s.n.], 2015. P. 1713–1728.

COOLEY, James W; TUKEY, John W. An algorithm for the machine calculation of complex Fourier series. **Mathematics of computation**, JSTOR, v. 19, n. 90, p. 297–301, 1965.

CORTEZ, Eli; SILVA, Altigran S da; GONÇALVES, Marcos André; MOURA, Edleno S de. Ondux: on-demand unsupervised learning for information extraction. In: ACM. PROCEEDINGS of the 2010 ACM SIGMOD International Conference on Management of data. [S.l.: s.n.], 2010. P. 807–818.

CRESCENZI, Valter; MECCA, Giansalvatore; MERIALDO, Paolo, et al. Roadrunner: Towards automatic data extraction from large web sites. In: VLDB. [S.l.: s.n.], 2001. P. 109–118.

ELIAS, Isaac. Settling the intractability of multiple alignment. **Journal of Computational Biology**, Mary Ann Liebert, Inc. 2 Madison Avenue Larchmont, NY 10538 USA, v. 13, n. 7, p. 1323–1339, 2006.

ELMELEEGY, Hazem; MADHAVAN, Jayant; HALEVY, Alon. Harvesting relational tables from lists on the web. **The VLDB Journal - The International Journal on Very Large Data Bases**, Springer-Verlag New York, Inc., v. 20, n. 2, p. 209–226, 2011.

EVERITT, Brian S. **The Cambridge dictionary of statistics**. [S.l.]: Cambridge University Press, 2010. P. 89.

FANG, Yixiang; XIE, Xiaoqin; ZHANG, Xiaofeng; CHENG, Reynold; ZHANG, Zhiqiang. STEM: a suffix tree-based method for web data records extraction. **Knowledge and Information Systems**, Springer, v. 55, n. 2, p. 305–331, 2018.

FERRARA, Emilio; DE MEO, Pasquale; FIUMARA, Giacomo; BAUMGARTNER, Robert. Web data extraction, applications and techniques: A survey. **Knowledge-based systems**, Elsevier, v. 70, p. 301–323, 2014.

GFRERER, Christine; VAJTERŠIC, Marián; KUTIL, Rade. Parallel Algorithms to Align Multiple Strings in the Context of Web Data Extraction. In: EMERGENT Computation. [S.l.]: Springer, 2017. P. 525–578.

GIBSON, David; PUNERA, Kunal; TOMKINS, Andrew. The volume and evolution of web page templates. In: ACM. WWW. [S.l.: s.n.], 2005. P. 830–839.

GOERTZEL, Gerald. An algorithm for the evaluation of finite trigonometric series. **The American Mathematical Monthly**, JSTOR, v. 65, n. 1, p. 34–35, 1958.

GRIGALIS, Tomas. Towards web-scale structured web data extraction. In: ACM. PROCEEDINGS of the sixth ACM international conference on Web search and data mining. [S.l.: s.n.], 2013. P. 753–758.

GUO, Jinsong; CRESCENZI, Valter; FURCHE, Tim; GRASSO, Giovanni; GOTTLOB, Georg. RED: Redundancy-Driven Data Extraction from Result Pages? In: ACM. THE World Wide Web Conference. [S.l.: s.n.], 2019. P. 605–615.

GUSFIELD, Dan. **Algorithms on strings, trees, and sequences: computer science and computational biology**. [S.l.]: Cambridge university press, 1997.

GUSFIELD, Dan. Efficient methods for multiple sequence alignment with guaranteed error bounds. **Bulletin of mathematical biology**, Elsevier, v. 55, n. 1, p. 141–154, 1993.

JINDAL, Nitin; LIU, Bing. A generalized tree matching algorithm considering nested lists for web data extraction. In: SIAM. PROCEEDINGS of the 2010 SIAM International Conference on Data Mining. [S.l.: s.n.], 2010. P. 930–941.

KAYED, Mohammed; CHANG, Chia-Hui. FiVaTech: Page-level web data extraction from template pages. **IEEE transactions on knowledge and data engineering**, IEEE, v. 22, n. 2, p. 249–263, 2009.

LIU, Bing; CHEN-CHUAN-CHANG, Kevin. Editorial: special issue on web content mining. **Acm SIGKDD explorations newsletter**, ACM, v. 6, n. 2, p. 1–4, 2004.

LIU, Bing; GROSSMAN, Robert; ZHAI, Yanhong. Mining data records in web pages. In: ACM. PROCEEDINGS of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining. [S.l.: s.n.], 2003. P. 601–606.

LIU, Bing; ZHAI, Yanhong. NET–a system for extracting web data from flat and nested data records. In: SPRINGER. INTERNATIONAL Conference on Web Information Systems Engineering. [S.l.: s.n.], 2005. P. 487–495.

LIU, Wei; MENG, Xiaofeng; MENG, Weiyi. Vide: A vision-based approach for deep web data extraction. **IEEE Transactions on Knowledge and Data Engineering**, IEEE, v. 22, n. 3, p. 447–460, 2009.

MANBER, Udi; MYERS, Gene. Suffix arrays: a new method for on-line string searches. **siam Journal on Computing**, SIAM, v. 22, n. 5, p. 935–948, 1993.

MIAO, Gengxin; TATEMURA, Junichi; HSIUNG, Wang-Pin; SAWIRES, Arsany; MOSER, Louise E. Extracting data records from the web using tag path clustering. In: ACM. PROCEEDINGS of the 18th international conference on World wide web. [S.l.: s.n.], 2009. P. 981–990.

OPPENHEIM et al. **Discrete-time signal processing**. [S.l.]: Prentice hall Englewood Cliffs, NJ, 1989. v. 2.

PEDREGOSA, Fabian et al. Scikit-learn: Machine learning in Python. **Journal of machine learning research**, v. 12, Oct, p. 2825–2830, 2011.

QIU, Disheng; BARBOSA, Luciano; DONG, Xin Luna; SHEN, Yanyan; SRIVASTAVA, Divesh. Dexter: large-scale discovery and extraction of product specifications on the web. **Proceedings of the VLDB Endowment**, VLDB Endowment, v. 8, n. 13, p. 2194–2205, 2015.

RASCHKA, Sebastian. MLxtend: Providing machine learning and data science utilities and extensions to Python's scientific computing stack. **The Journal of Open Source Software**, The Open Journal, v. 3, n. 24, Apr. 2018. DOI: `10.21105/joss.00638`. Available from: `http://joss.theoj.org/papers/10.21105/joss.00638`.

ROLDÁN, Juan C; JIMÉNEZ, Patricia; CORCHUELO, Rafael. On extracting data from tables that are encoded using HTML. **Knowledge-Based Systems**, Elsevier, 2019.

SCHULZ, Andreas; LÄSSIG, Jörg; GAEDKE, Martin. Practical Web data extraction: are we there yet?-a short survey. In: IEEE. 2016 IEEE/WIC/ACM International Conference on Web Intelligence (WI). [S.l.: s.n.], 2016. P. 562–567.

SHI, Shengsheng; LIU, Chengfei; SHEN, Yi; YUAN, Chunfeng; HUANG, Yihua. AutoRM: An effective approach for automatic Web data record mining. **Knowledge-Based Systems**, Elsevier, v. 89, p. 314–331, 2015.

SIMON, Kai; LAUSEN, Georg. ViPER: augmenting automatic information extraction with visual perceptions. In: ACM. PROCEEDINGS of the 14th ACM international conference on Information and knowledge management. [S.l.: s.n.], 2005. P. 381–388.

SLEIMAN, Hassan A; CORCHUELO, Rafael. A survey on region extractors from web documents. **IEEE Transactions on Knowledge and Data Engineering**, IEEE, v. 25, n. 9, p. 1960–1981, 2012.

UKKONEN, Esko. On-line construction of suffix trees. **Algorithmica**, Springer, v. 14, n. 3, p. 249–260, 1995.

VARLAMOV, MI; TURDAKOV, D Yu. A survey of methods for the extraction of information from Web resources. **Programming and Computer Software**, Springer, v. 42, n. 5, p. 279–291, 2016.

VELLOSO, Roberto Panerai; DORNELES, Carina F. Automatic Web Page Segmentation and Noise Removal for Structured Extraction using Tag Path Sequences. **JIDM**, v. 4, n. 3, p. 173, 2013.

VELLOSO, Roberto Panerai; DORNELES, Carina F. Extracting records from the web using a signal processing approach. In: ACM. PROCEEDINGS of the 2017 ACM on Conference on Information and Knowledge Management. [S.l.: s.n.], 2017. P. 197–206.

VELLOSO, Roberto Panerai; DORNELES, Carina F. Optimized Extraction of Records from the Web Using Signal Processing and Machine Learning. **SBBD 2020**, 2020.

VELLOSO, Roberto Panerai; DORNELES, Carina F. Web Page Structured Content Detection Using Supervised Machine Learning. In: SPRINGER. INTERNATIONAL Conference on Web Engineering. [S.l.: s.n.], 2019. P. 3–18.

WAI, Fok Kar; YONG, Lim Wee; THING, Vrizlynn LL; POMPONIU, Victor. CMDR: Classifying nodes for mining data records with different HTML structures. In: IEEE. TENCON 2017-2017 IEEE Region 10 Conference. [S.l.: s.n.], 2017. P. 1862–1862.

WANG, Haizhou; SONG, Mingzhou. Ckmeans. 1d. dp: optimal k-means clustering in one dimension by dynamic programming. **The R Journal**, v. 3, n. 2, p. 29–33, 2011.

WANG, Jingjing; WANG, Haixun; WANG, Zhongyuan; ZHU, Kenny Q. Understanding tables on the web. In: SPRINGER. INTERNATIONAL Conference on Conceptual Modeling. [S.l.: s.n.], 2012. P. 141–155.

WU, Wentao; LI, Hongsong; WANG, Haixun; ZHU, Kenny Q. Probase: A probabilistic taxonomy for text understanding. In: ACM. PROCEEDINGS of the 2012 ACM SIGMOD International Conference on Management of Data. [S.l.: s.n.], 2012. P. 481–492.

XIE, Xiaoqin; FANG, Yixiang; ZHANG, Zhiqiang; LI, Li. Extracting data records from web using suffix tree. In: ACM. PROCEEDINGS of the ACM SIGKDD Workshop on Mining Data Semantics. [S.l.: s.n.], 2012. P. 12.

YAMADA, Yasuhiro; CRASWELL, Nick; NAKATOH, Tetsuya; HIROKAWA, Sachio. Testbed for information extraction from deep web. In: ACM. PROCEEDINGS of the 13th international World Wide Web conference on Alternate track papers & posters. [S.l.: s.n.], 2004. P. 346–347.

ZHAI, Yanhong; LIU, Bing. Web data extraction based on partial tree alignment. In: ACM. PROCEEDINGS of the 14th international conference on World Wide Web. [S.l.: s.n.], 2005. P. 76–85.

ZHANG, Zhixian; ZHU, Kenny Q; WANG, Haixun; LI, Hongsong. Automatic extraction of top-k lists from the web. In: IEEE. 2013 IEEE 29th International Conference on Data Engineering (ICDE). [S.l.: s.n.], 2013. P. 1057–1068.

ZIV, Jacob; LEMPEL, Abraham. A universal algorithm for sequential data compression. **IEEE Transactions on information theory**, IEEE, v. 23, n. 3, p. 337–343, 1977.

APPENDIX A – MODEL PARAMETERS

Table 17 – Grid Search result: algorithm parameters.

Model	Parameters
LR	*C* = 1.0 *penalty* = *L2* *warm_start* = *True*
GNB	none
EXT	*max_features* = 0.8 *criterion* = *entropy* *warm_start* = *False* *n_estimators* = 31 *bootstrap* = *True* *min_samples_leaf* = 5 *min_samples_split* = 0.02
SVM	*kernel* = *poly* *C* = 0.6 *gamma* = 0.188 *degree* = 3 *shrinking* = *True*
kNN	*weights* = *uniform* *p* = 3 *n_neighbors* = 13
GB	*min_child_weight* = 9 *subsample* = 0.60 *n_estimators* = 92
VOT	*clfs* = [*GNB*, *SVM*, *kNN*, *GB*] *weights* = [0.5, 1.0, 0.5, 0.5]
STCK	*classifiers* = [*GNB*, *SVM*, *kNN*, *GB*] *meta_classifier* = *LogisticRegression*

APPENDIX B – EXTRACTION FRAMEWORK USAGE EXAMPLE

Listings B.1 and B.2 show a small code snippet, in Lua and Python respectively, that parses an HTML document, extracts and displays the records found in it. The classes `DOM` and `DSRE` are implemented in C++ and their interfaces are exposed under the Lua and Python environments.

The Lua environment also has other utilities available to the user like: embedded SQLite to store extracted results and a WebDriver implementation to remote control FireFox and Chrome browsers.

Listing B.1 – Example of Lua source code calling C++ exposed interfaces.

```
extractFile = function(filename)
        -- loads and parses HTML into a DOM tree
        local dom = DOM.new(filename)

        -- instantiates an extractor
        local dsre = DSRE.new()

        -- sets z-score and CV thresholds
        dsre:setMinPSD(minZScore)
        dsre:setMinCV(minCV)

        -- extract records
        dsre:extract(dom)

        local regions = dsre:regionCount()

        -- iterates over regions
        for i=1,regions do
                local dr = dsre:getDataRegion(i-1)
                local rows = dr:recordCount()
                local cols = dr:recordSize()

                -- iterates over current region's
                -- rows and columns
                for r=1,rows do
                        local record = dr:getRecord(r-1)
                        for c=1,cols do
                                -- output field value separated by ';'
                                if (record[c] ~= nil) then
                                        io.write(record[c]:toString())
                                end
                                io.write(';')
                        end
                        io.write('\n')
                end
        end
        dsre:printTps()
        dom:printHTML()
end

extractFile('./test.html')
```

Listing B.2 – Example of Python source code calling C++ exposed interfaces.

```
import webMining as wm
from IPython.core.display import display, HTML
from tabulate import tabulate
from matplotlib import pyplot as plt
from matplotlib import patches
import requests
import numpy as np

doc = wm.DOM('./test.html')
dsre = wm.DSRE()

dsre.extract(doc)

regions = []
for reg in range(0, dsre.regionCount()):
region = dict()
dr = dsre.getDataRegion(reg)
region['content'] = dr.isContent()
region['sequence'] = dr.getSequence()
region['transform'] = dr.getTransform()
region['table'] = dr.getTable()
region['rows'] = len(region['table'])
region['start'] = dr.getStartPos()
region['end'] = dr.getEndPos()
regions.append(region)

content_string = ['noise', 'content']
for i, region in enumerate(regions):
        display('Region␣#' + str(i) + '␣-␣' +
                content_string[region['content']] +
                ',␣' + str(region['rows']) + '␣records')
        display(tabulate(region['table'], tablefmt='html'))

rows = len(regions) + 1
plt.figure(figsize=(15,15))
ax = plt.subplot(rows, 1, 1)
plt.title('document')
plt.plot(dsre.getSequence())
for i, region in enumerate(regions):

        xy = (region['start'], min(region['sequence']))
        width = region['end'] - region['start']
        height = max(region['sequence']) - min(region['sequence'])
        color = "green" if region['content'] else "red"
        ax.add_patch(patches.Rectangle(xy, width, height, color=color, fill=False))

        plt.subplot(rows, 2, (i+1)*2 + 1)
        plt.title('region␣#' + str(i) + '␣-␣' + content_string[region['content']])
        plt.plot(region['sequence'])
        plt.subplot(rows, 2, (i+1)*2 + 2)
        plt.title('region␣#' + str(i) + '␣-␣Fourier␣Transform')
        sequence_len = len(region['sequence'])
        transform_len = len(region['transform'])
        plt.plot(
        np.linspace(0, sequence_len, transform_len),
        region['transform'][0:transform_len])
        plt.axvline(region['rows'], min(region['transform']), max(region['transform']), c='r')
        plt.plot()
```

Table 18 – Output Example.

<img ...	7:27...	...	<img ...	Os Me...	de ...	2 dia...	9.817...	Futeb...	Inscr...		NOVO...		
<img ...	7:28...	...	<img ...	Boliv...	de ...	2 dia...	32.21...	GoalB...	bate...		NOVO...		Brasi...
<img ...	13:31...	...	<img ...	O BRA...	de ...	2 ano...	1.855...	bueno...	Repor...	.Muit...			Brasi...
<img ...	5:42...	...	<img ...	Ines ...	de ...	5 mes...	1.596...	ines ...	Ines ...	BBB ...	HD...		Brasi...
<img ...	10:05...	...	<img ...	PÃªna...	de ...	2 ano...	220.0...	Panth...	Dispu...	e Ho...			Brasi...
<img ...	9:06...	...	<img ...	Acord...	de ...	6 mes...	550.0...	Luiz ...	Video...	e no...	HD...		Brasi...
<img ...	12:52...	...	<img ...	Brasi...	de ...	4 mes...	77.67...	leand...	Acess...				
<img ...	14:18...	...	<img ...	Acord...	de ...	1 ano...	100.5...	Luiz ...	Video...	, em ...	HD...		Brasi...
<img ...	9:59...	...	<img ...	BRASI...	de ...	5 ano...	1.236...	wesle...	O NU...				BRASI...
<img ...	9:54...	...	<img ...	O dia...	de ...	6 ano...	2.018...	abelf...	Anima...	pelo...			Brasi...
<img ...	2:57...	...	<img ...	Os Go...	de ...	2 dia...	16.93...	Futeb...	Inscr...		NOVO...		
<img ...	49:06...	...	<img ...	Agora...	de ...	1 sem...	115.3...	fuuuh...	http:...		HD...		
<img ...	8:30...	...	<img ...	Resum...	de ...	1 ano...	366.8...	tassi...	Profe...				
<img ...	2:03:...	...	<img ...	JOGO ...	de ...	2 ano...	416.0...	Marki...	na F...				Brasi...
<img ...	7:14...	...	<img ...	Os Me...	de ...	2 sem...	2.754...	Futeb...	A Omn...	....			Brasi...
<img ...	3:18:...	...	<img ...	A His...	de ...	1 ano...	417.7...	Dylso...	SÃ©ri...				
<img ...	11:47...	...	<img ...	MELHO...	de ...	2 sem...	3.392...	Akile...	MELHO...	2 x ...			Brasi...
<img ...	4:01...	...	<img ...	Boliv...	de ...	2 dia...	1.488...	Repla...	Os Go...	- Am...	NOVO...		Brasi...
<img ...	2:44...	...	<img ...	GOLS ...	de ...	2 dia...	3.901...	Futeb...	GOLS ...	x BO...	HD...	NOVO...	BRASI...
<img ...	2:44...	...	<img ...	BolÃ...	de ...	2 dia...	974 v...	2012B...	BolÃ...	- Go...			Brasi...

APPENDIX C – FEATURES PAIR PLOTS

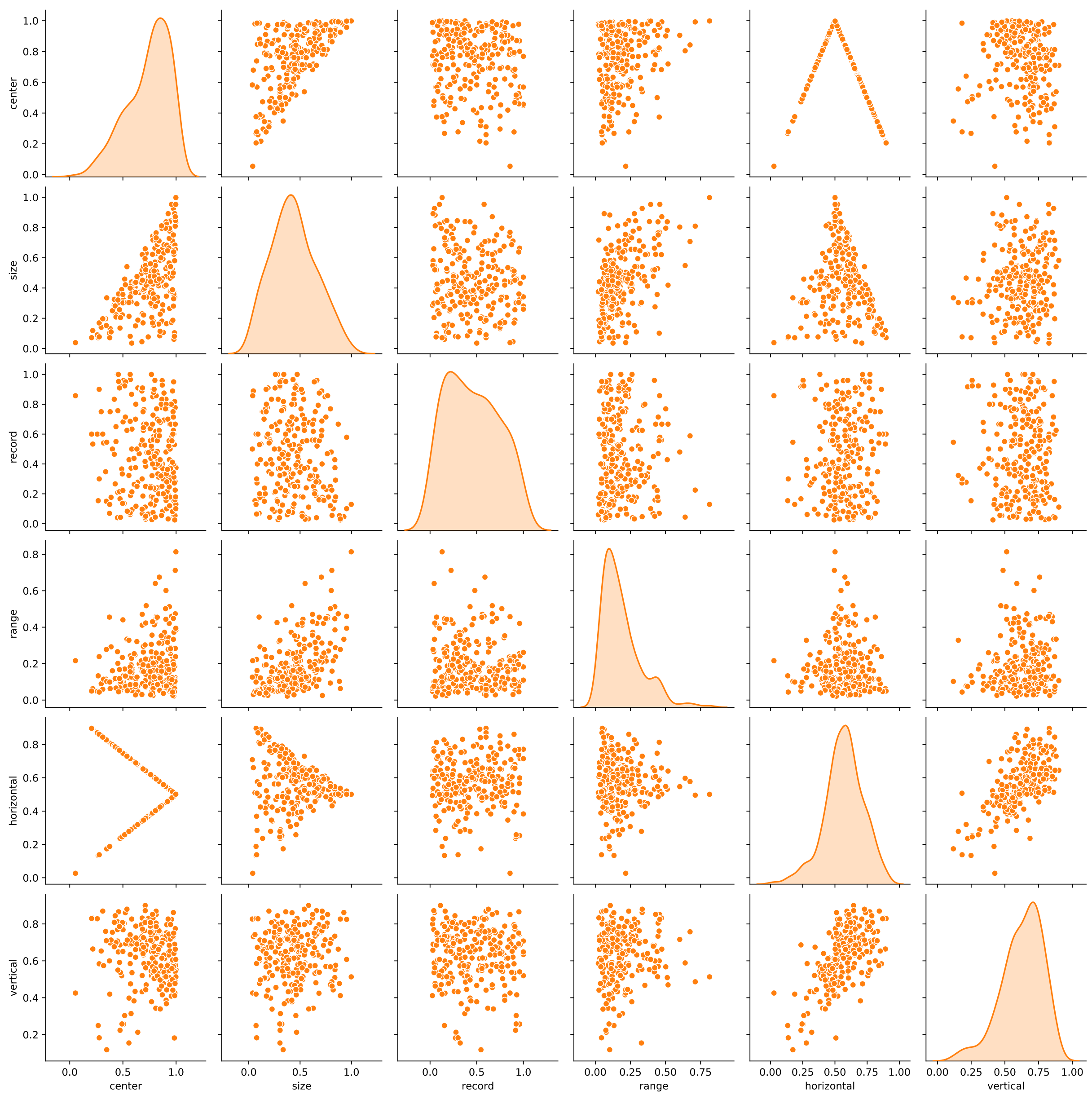


Figure 16 – Content pair plot and kernel density estimation.

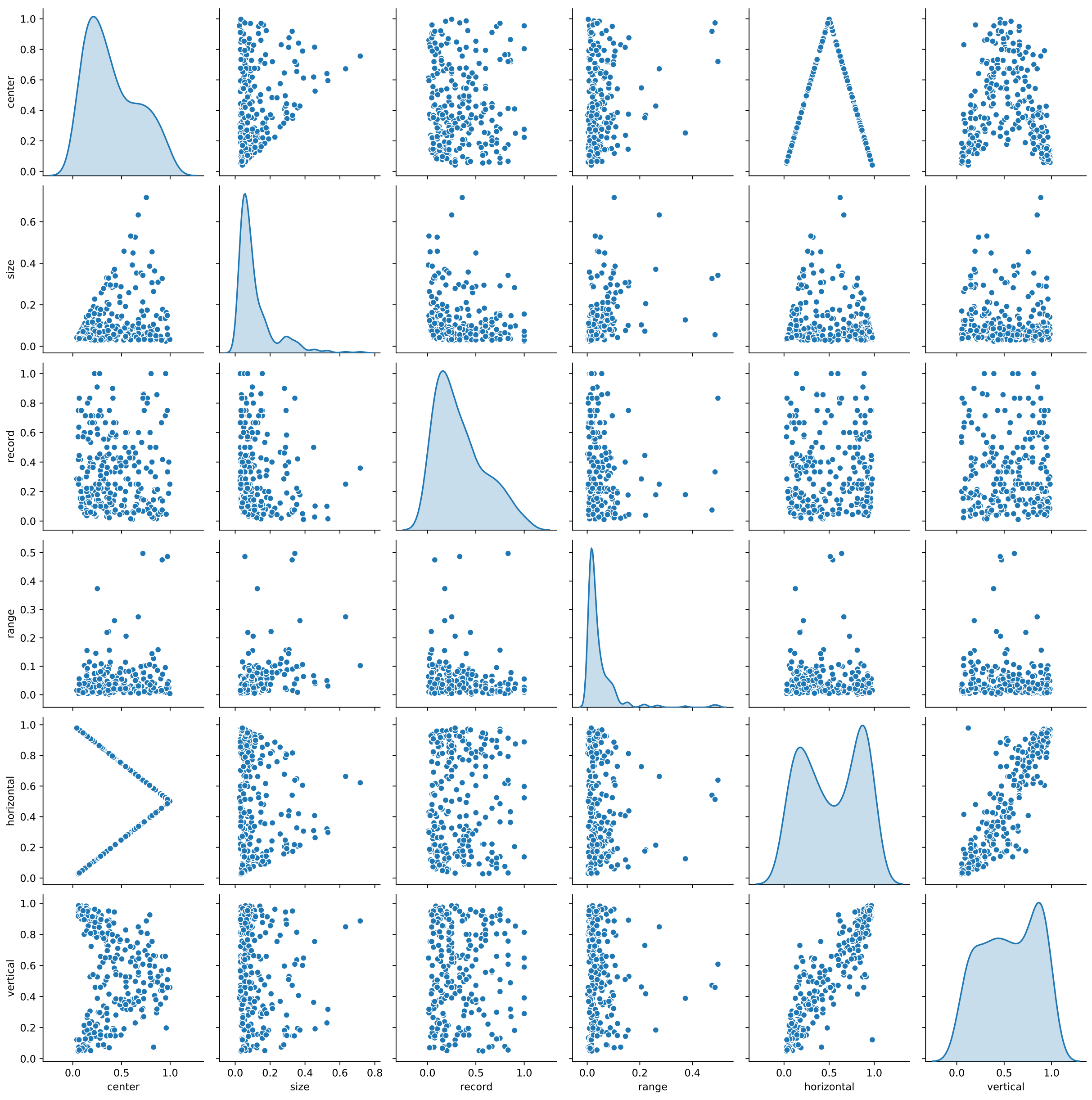


Figure 17 – Noise pair plot and kernel density estimation.

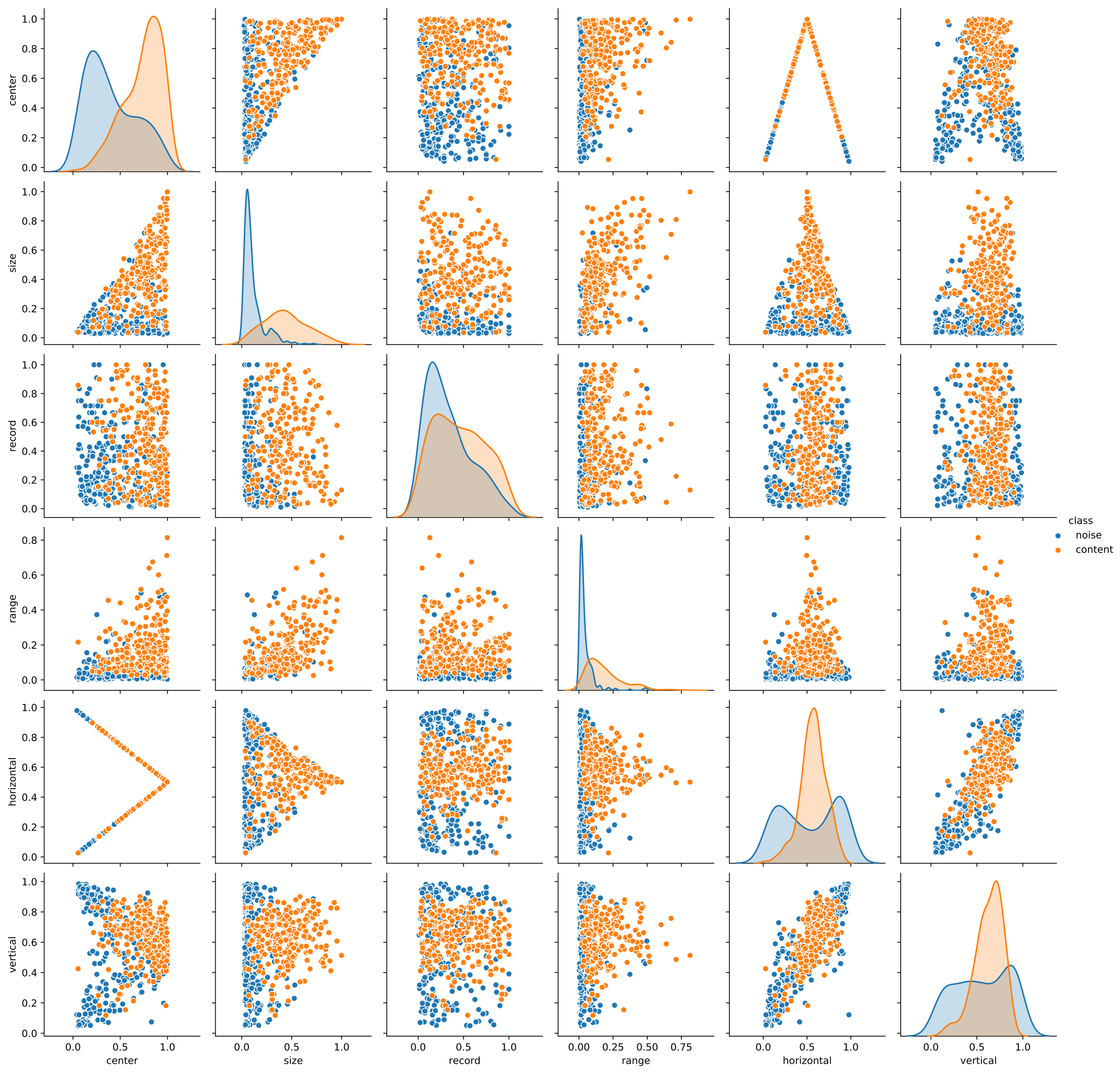


Figure 18 – Content vs. noise pair plot and kernel density estimation.