Web Page Structured Content Detection Using Supervised Machine Learning

Roberto Panerai Velloso(✉) and Carina F. Dorneles

Universidade Federal de Santa Catarina - UFSC, Florianopolis, SC, Brazil
rvelloso@gmail.com, dorneles@inf.ufsc.br

Abstract. In this paper we 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 specifically 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 a preexisting and publicly available extraction technique that processes web pages as a sequence of tag paths, thus the features are extracted from these sequences instead of the DOM tree. Besides comparing the performance between different models we have also conducted extensive feature selection/combination experiments. We have achieved an average F-score of about 93% in a controlled setting and 91% in an open setting.

Keywords: Web mining · Content detection · Noise removal · Record extraction · Structure detection · Information retrieval

1 Introduction

The web is an invaluable source of data and information about virtually any subject we can think of. Some of this information is made available to the public in a structured fashion (e.g., shopping items, news, search engine results, etc.), providing some level of organization that can be exploited and leveraged: one can use it to detect/find structured content in a document. But there are other parts of a document, besides the main content, that can have some sort of organization (template and menus, for instance), this organized "non-content" information adds noise to the extraction process, decreasing precision. How can we distinguish between them? This is the problem we tackle in this paper.

Extracting structured information, by itself, is an important task, but we must also 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 [5,16] between 40%–50% of a web document refers to noise (menus, template, ads), this amount is more than enough to completely

M. Bakaev et al. (Eds.): ICWE 2019, LNCS 11496, pp. 3–18, 2019.
https://doi.org/10.1007/978-3-030-19274-7_1

compromise extraction precision. Since structured data can exist in any kind of web document, whether main content is structured or not, we must be able to identify it correctly independently from its source, even when there is no structured content, i.e. we must be able to identify noise in a document whose main content is textual as well as in a document whose main content is structured. So, once we have extracted the structured content from a document, how can we classify it as content or noise? We could not find in the literature, nor did we reach a deterministic and closed form way of solving this problem (classify content and noise), for this reason we decided to characterize (create conjectures for the features) content and noise the best we could and try out machine learning models to approximate a solution. Our goal is to classify **structured data**, specifically, as content or noise, but without restricting ourselves just to **structured content** documents (i.e., we also want to detect structured noise in textual content documents). This is a desirable property in order to avoid manual intervention (selecting only certain types of documents for processing) for the web is completely heterogeneous in all aspects. But we also evaluated the models in a controlled setting, when the web pages are known beforehand to have structured content. This scenario is not unrealistic (e.g., a focused crawler that retrieves only search result records from e-commerce sites), although it does demands more manual intervention.

Previous attempts at this problem, such as [6,7,17,19], were targeted at textual content, their performance is measured in tasks such as clustering and classification of web pages, not in terms of records extracted. It is also not clear whether or not these approaches can be used with structured content, they might remove part of the content, believing it is noise, without affecting clustering, but this removal would most likely impair record extraction. Other attempts ([15,16]), targeted at structured content, can not be used in an open environment (i.e., an environment where we encounter any kind of content: textual, structured or hybrid), they assume only structured content will be processed. Although this is not completely unrealistic, it is also not as general, demanding more controlled environments of execution (i.e., more manual intervention needed).

In this paper, we analyze several possible 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. For the extraction phase we choose the method proposed in [15] because of the good quality of the results, its feasibility in a production environment (it is unsupervised and computationally efficient compared to other state-of-the-art approaches) and also because its source code is freely available for download, allowing reproducibility of results. This extraction method uses a signal processing approach to detect repetitive structural patterns in the document by means of stability and spectral analysis. The web page is converted to a sequence (or signal) representation, prior to extraction, and it is from this sequence that we derive the features used in our work.

The features proposed here are generated during the extraction phase. We just normalize their values, adding no overhead to the pipeline (once we have the model trained). We have attained 93% F-Score in a dataset consisting of 266 different HTML documents from various domains with structured content and 91% F-Score in a dataset with 327 different HTML documents, some with unstructured content (same 266 structured documents, plus 61 unstructured documents mostly from blogs, news, etc.). The novelty presented here lies in the nature of the features. We are using an alternative representation for the web documents, to the best of our knowledge this representation was first introduced in [11], and until now we have not found any content/noise detection proposal using features derived from this representation. Moreover, these features are automatically extracted, meaning that, once we have a trained classifier, no more human intervention is needed. These automatically acquired features contrast with [8], where human intervention is needed for feature extraction. At last, our investigation showed these features can be used to solve this problem effectively and efficiently using a simple and direct ML approach.

The rest of this paper is organized as follows: in Sect. 2, we present a brief review of related works; in Sect. 3, we reproduce some concepts needed for the understanding of our work; in Sect. 4, we describe and illustrate each feature used in solving the problem of content detection; in Sect. 5, we detail and discuss the experiments conducted; in Sect. 6, we analyze the results achieved and; in Sect. 7, we present our conclusions.

2 Related Work

In our research we have encounter quite a few proposals for web page noise removal, dating as far back as 1999 [9]. Most early works focused in textual content, where main applications were web page clustering and classification.

In [3,6,7,17] the main content of a web page is assumed to be textual, they might be a fit for a web page with structured content, but that is unlikely and we have no results published using these techniques for this purpose, as far as we know.

On the other hand, if we assume content is structured ([15,16]) we loose generality and became confined in this setting. An approach biased toward structured content works great in a controlled environment but can not perform well in an open environment where we may encounter textual and hybrid content as well, precision would drop drastically. There is a cost (usually manual intervention and usually high) associated with maintaining such a well controlled environment.

We also found many proposals that do not assume content to be structured or textual ([2,4,8,19,20]), but each has some limitation we intend to overcome in our work. In [4,19,20] several samples from the same template are necessary to train the model, and it only works for that particular template; in [8] human intervention is needed to define a priori rules; and in [2] predefined knowledge bases are required.

Much has changed, since this area of research has started, new applications have arisen, web development culture has changed, among other things. Due to the web's ever changing nature, any proposal based on too specific assumptions (content form, predefined knowledge, template specific, static heuristics rules, etc.) is deemed to rapidly become outdated.

3 Preliminaries

Since we are proposing a way of identifying structured content and noise, we need to build our work on top of an extraction technique. We chose the approach reported in [15] for two reasons: (i) the results reported are equivalent to other state-of-the-art approaches and; (ii) the computational complexity is lower, especially when compared to rendering-based approaches. This extraction technique uses an alternative representation for the web documents, a **tag path sequence** (or TPS), and here we detail this representation. The understanding of this alternate representation is needed because the features we use to classify content and noise are derived from it. For a more thorough explanation we refer the reader to the work in [15].

Definition 1. ***(Tag Path)*** *is a string describing the path from the root node of the DOM tree to another node in the tree. For example:* *"*`html/body/table/tr/td/#text`*".*

Definition 2. ***(Tag Path Code – TPCode)*** *is a numeric ascending code assigned to every different tag path string encountered in the document tree, in order of appearance. If a given path has already occurred, it is assigned the same code as before. The paths are built in depth first order. Figure 1 shows an example of this definition.*

Definition 3. ***(Tag Path Sequence – TPS)*** *is a sequence of TPCodes in the same order as they were built from the DOM tree. Figure 1 shows the resulting TPS for an HTML snippet as well as the set of TPCodes used in that sequence. In this paper we also refer to TPS as simply "sequence".*

The translation process from DOM tree representation to tag path sequence is depicted in Fig. 1. 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.

Figure 2 illustrates a real web page converted to its sequence representation with its structured regions encircled and its main content region highlighted. We will use this specific sequence and its main content region throughout Sect. 4 to characterize the features. The sequence was constructed according to the definitions in this section. Every point in the sequence (Definition 3) corresponds to a specific node in the DOM tree and has a TPCode value (Definition 2) that encodes the node's Tag Path (Definition 1).

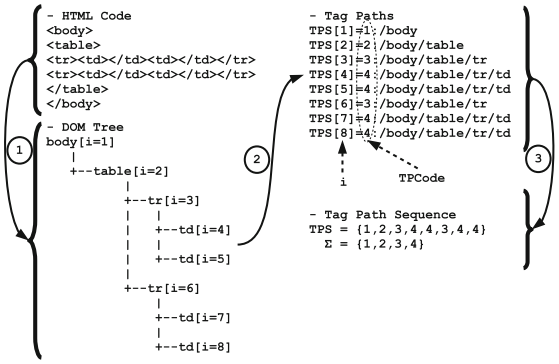

Fig. 1. Conversion of HTML snippet into a tag path sequence.

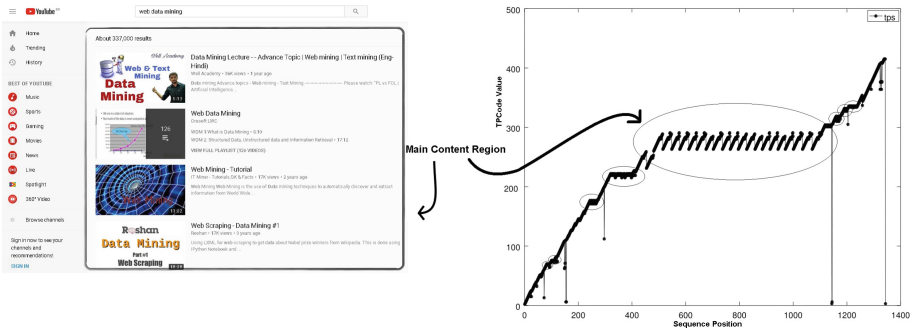

Fig. 2. Web page converted to sequence representation and its structured content.

Definition 4. ***(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 Fig. 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 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. As an extra, they can also be easily acquired from the extraction technique we are relying on. We will discuss each feature (size, positions, range and record proportion) in Subsects. 4.1, 4.2, 4.3 and 4.4 respectively.

4.1 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.

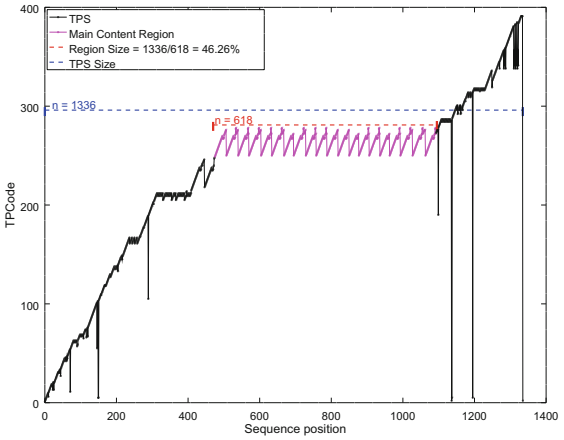

Fig. 3. Size feature example.

Figure 3 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 Eq. 1, is equal to $\frac{618}{1{,}336} = 46.25\%$.

$$sizeFeat = \frac{regionSize}{sequenceSize} \tag{1}$$

4.2 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 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 (Subsect. 4.1): the closer a region is to the center of the document, the higher the probability it refers to real content.

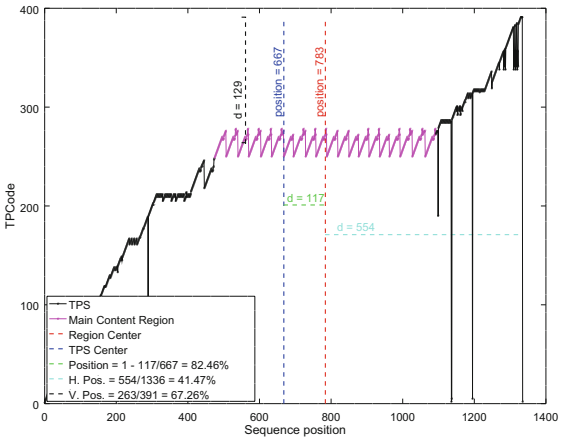

Fig. 4. Position feature example. (Color figure online)

Figure 4 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 Eq. 2, is equal to $1 - \frac{117}{667} = 82.46\%$

$$centerPositionFeat = 1 - \frac{|regionCenter - sequenceCenter|}{sequenceCenter} \tag{2}$$

With respect to the vertical and horizontal position, we believe they are needed to provide a better indication of a region's position, **especially** when a

document has no structured content (only structured noise), in this situation 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 4 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 Eq. 4, the value of the horizontal position, in this example, is equal to $\frac{554}{1{,}336} = 41.47\%$.

$$horizPositionFeat = \frac{sequenceSize - regionCenter}{sequenceSize} \tag{3}$$

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 Eq. 4, the value of the vertical position, in this example, is equal to $\frac{263}{391} = 67.26\%$.

$$vertPositionFeat = \frac{avg(region)}{max(sequence)} \tag{4}$$

Throughout this paper we will refer to these three features simply as "center", "horizontal" and "vertical" features.

4.3 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 analogous to the Size Feature (Subsect. 4.1) only 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.

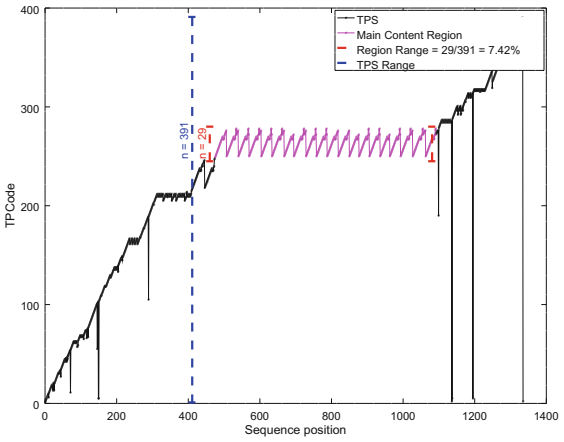

Fig. 5. Range feature example.

Figure 5 shows an example where region range is equal to 29 and document range is equal to 391. The value of this feature, using Eq. 5, is equal to $\frac{283-254}{391} = \frac{29}{391} = 7.42\%$ of document range.

$$rangeFeat = \frac{regionRange}{max(sequence)} \tag{5}$$

4.4 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 this 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 Eq. 6.

$$recRatioFeat = \frac{min(numRecs, recCount)}{max(numRecs, recCount)} \tag{6}$$

The value of this feature is also a real number between 0 and 1, since the denominator in Eq. 6 is always greater or equal to the numerator.

This two measures are obtained as documented in [15], using the region's power spectrum density (PSD) [12]. Figure 6b shows the PSD of the main content region in Fig. 6a and its detected record count and average record size. The sequence in Fig. 6a is the extracted main content region from the sequence depicted in Fig. 2. In this example the number of records (represented by the red peak in Fig. 6b) detected is 20 and their average size is 31, therefore the value of the record feature, using Eq. 6, is equal do $\frac{20}{31} = 64.51\%$

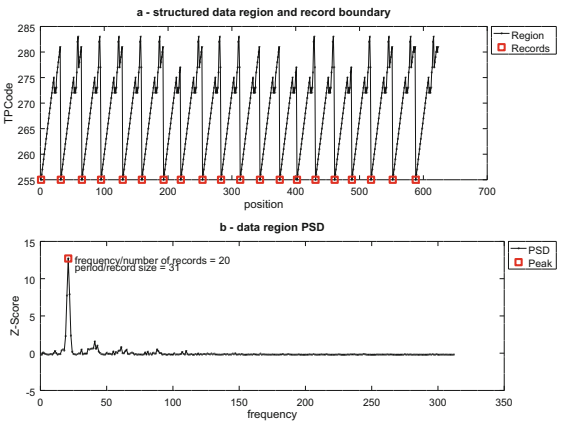

Fig. 6. Record count & size feature example.

Throughout this paper we will refer to this feature as "record" feature.

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

5 Experiments

In this section we detail the experiments we conducted using supervised machine learning techniques with the features from Sect. 4. We also characterize the dataset used in the experiments, 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), 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 [13] framework. For the gradient boosting we used XGBoost [1] and for the ensembles we used MLxtend's [14] implementation.

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. The feature set for each algorithm is documented in Table 8 (more details on that in Sect. 6).

For these experiments we have used a dataset consisting of 266 HTML documents with structured content from various domains (news, banking, hotels, car rental, tickets, electronics), plus 61 documents **without** structured content, totalizing 327 documents. The documents without structured content were added to the dataset to investigate the behaviour of the classifiers in the presence of this type of input. We use only one page per site to avoid introducing bias towards specific sites and/or templates[2]. These documents were processed using the technique proposed in [15], resulting in a total of 533 regions. We acknowledge 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. 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.

[2] This is possible because the extraction approach used also works with a single page input.

Table 1. Input dataset summary

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

Table 2. 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

That is our input dataset[3] used for training and cross-validation and it is summarized in Table 1. Figure 7 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. Table 2 shows the features relative importance according to two different criteria (ANOVA and χ^2), both yielding the same results. Table 4 shows mean, coefficient of variation (CV), skewness and kurtosis for all features with respect to the target class. Table 3 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 2, are also the most important ones.

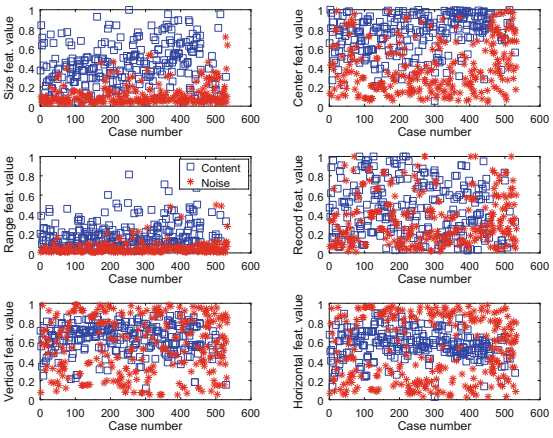

Fig. 7. Input dataset features: content vs noise.

[3] Our dataset is available at https://bit.ly/2D3lWFk.

Table 3. 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

Table 4. Feature statistics

Feature	Mean	CV	S kewness	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

6 Results

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 5 and 6.

When we omit the documents without structured content we get the results shown in Table 5. 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.

When we consider the full dataset, including the documents without structured content, we get the results shown in Table 6. As expected there is a drop in F-Score (column "Drop" in Table 6). The amount of unstructured documents corresponds, roughly, to 18% of the entire dataset (see Table 1) 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.

Table 5. 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%

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%).

Table 6. 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%

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 between content and noise. Fortunately, these cases seem to be a minority.

Table 7. Comparison w/other approaches

Algorit.	Prec.	Recall	F-Score	Acc.
BERyL [8]	n/d	n/d	90.00%	n/d
SIG [15]	92.02%	94.11%	93.05%	n/d
TPC [11]	90.40%	93.10%	91.73%	n/d
MDR [10]	59.80%	61.80%	60.78%	n/d
Our models				
LR	**95.45%**	95.45%	**95.45%**	94.80%
VOT	**93.02%**	90.91%	**91.95%**	90.91%

Table 8. Best set of features p/algorithm.

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

In Table 7 we have compared our results with other, state-of-the-art, approaches found in the literature. We have compared the performance of our classifiers in both settings (Logistic Regression from Table 5, trained only with structured content, and Voting Ensemble from Table 6 trained with our full dataset) with the results published in [8,11,15], using their reference dataset (from [18]) as our test data, except for [8] for which the dataset is not publicly available, in this case we compare only against raw published results. The work in [11,15] deals specifically with structured content, no assessment was made in the presence of noise as we did in Table 6. We can see, in Table 7, that our work outperformed the other approaches (MDR [10] by a large margin). We've outmatched BERyL, TPC and MDR even in the presence of unstructured content documents. BERyL uses hand tailored rules to extract relevant features to train the classifiers and we have outperformed its results with automatically extracted features. With respect to the SIG [15] approach, we have achieve superior results in a controlled setting and better precision in an open environment.

We show in Table 8 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 3: "center" feature has a considerable correlation with "size" and "vertical" feature with "horizontal".

7 Conclusion and Future Work

In our research, through observation, we have come up with the conjectures depicted here, for each 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 features. Moreover, we have also shown that our proposal effectively solves the problem and is superior to other state-of-the-art approaches found in the literature.

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.

References

1. Chen, T., Guestrin, C.: XGBoost: a scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD (2016)
2. Cho, W.T., Lin, Y.M., Kao, H.Y.: Entropy-based visual tree evaluation on block extraction. In: Proceedings of the 2009 IEEE/WIC/ACM, pp. 580–583. IEEE Computer Society (2009)
3. Fernandes, et al.: Computing block importance for searching on web sites. In: CIKM, pp. 165–174. ACM (2007)
4. Fernandes, et al.: A site oriented method for segmenting web pages. In: SIGIR, pp. 215–224. ACM (2011)
5. Gibson, D., Punera, K., Tomkins, A.: The volume and evolution of web page templates. In: WWW, pp. 830–839. ACM (2005)
6. Kohlschütter, C., Fankhauser, P., Nejdl, W.: Boilerplate detection using shallow text features. In: WSDM, pp. 441–450. ACM (2010)
7. Kohlschütter, C., Nejdl, W.: A densitometric approach to web page segmentation. In: CIKM, pp. 1173–1182. ACM (2008)
8. Kravchenko, A., Fayzrakhmanov, R.R., Sallinger, E.: Web page representations and data extraction with BERyL. In: Pautasso, C., Sánchez-Figueroa, F., Systä, K., Murillo Rodríguez, J.M. (eds.) ICWE 2018. LNCS, vol. 11153, pp. 22–30. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03056-8_3
9. Kushmerick, N.: Learning to remove internet advertisements. In: Proceedings of the Third Annual Conference on Autonomous Agents, pp. 175–181. ACM (1999)
10. Liu, B., Grossman, R., Zhai, Y.: Mining data records in web pages. In: SIGKDD, pp. 601–606. ACM (2003)
11. Miao, G., Tatemura, J., Hsiung, W.P., Sawires, A., Moser, L.E.: Extracting data records from the web using tag path clustering. In: WWW, pp. 981–990. ACM (2009)
12. Oppenheim, A.V., et al.: Discrete-Time Signal Processing. Prentice Hall, Englewood Cliffs (1989)

13. Pedregosa, F., et al.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
14. Raschka, S.: Mlxtend: providing machine learning and data science utilities and extensions to Python's scientific computing stack. J. Open Source Softw. **3**(24), 638 (2018). https://doi.org/10.21105/joss.00638. http://joss.theoj.org/papers/10.21105/joss.00638
15. Velloso, R.P., Dorneles, C.F.: Extracting records from the web using a signal processing approach. In: CIKM 2017 (2017)
16. Velloso, R.P., Dorneles, C.F.: Automatic web page segmentation and noise removal for structured extraction using tag path sequences. JIDM **4**(3), 173 (2013)
17. Vieira, K., et al.: A fast and robust method for web page template detection and removal. In: CIKM, pp. 258–267. ACM (2006)
18. Yamada, Y., Craswell, N., Nakatoh, T., Hirokawa, S.: Testbed for information extraction from deep web. In: WWW, pp. 346–347. ACM (2004)
19. Yi, L., Liu, B., Li, X.: Eliminating noisy information in web pages for data mining. In: SIGKDD, pp. 296–305. ACM (2003)
20. Zheng, S., Song, R., Wen, J.R., Wu, D.: Joint optimization of wrapper generation and template detection. In: SIGKDD, pp. 894–902. ACM (2007)