World Wide Web (2019) 22:1999–2015
https://doi.org/10.1007/s11280-018-0634-6

Large-scale holistic approach to Web block classification: assembling the jigsaws of a Web page puzzle

Andrey Kravchenko[1]

Received: 11 April 2018 / Revised: 29 June 2018 / Accepted: 21 August 2018 /
Published online: 12 September 2018

Abstract
Web blocks are ubiquitous across the Web. Navigation menus, advertisements, headers, footers, and sidebars can be found almost on any website. Identifying these blocks can be of significant importance for tasks such as wrapper induction, assistance to visually impaired people, Web page topic clustering, and Web search among a few. There have been several approaches to the problem of Web block classification, but they focused on specific types of blocks, trying to classify all of them with one single set of features. In our approach each classifier has its own unique extendable set of features, with the features being extracted through the declarative-based BER_yL language, and the classification itself is done through application of machine learning to these feature sets. In our approach we propose to take a holistic view of the page where all block classifiers in the classification system interact with each other, and accuracies of individual classifiers get improved through this interaction. The holistic approach to Web block classification is implemented through a system of constraints in our block classification system BER_yL. The evaluation results with the holistic approach applied to the BER_yL classification system achieve higher F_1 results than for individual non-connected classifiers, with the average F_1 of 98%. We also consider the distinction between classification of domain-independent and domain-dependent blocks and propose a large-scale solution to the problem of classification for both of these block types.

Keywords Web block classification · Constraint satisfaction · Declarative programming · Machine learning

Dedication: To the late Professor Stanislav V. Klimenko with gratitude for his devotion to science and his students.

This work was supported by the ESPRC programme grant EP/M025268/1 "VADA: Value Added Data Systems – Principles and Architecture".

This article belongs to the Topical Collection: *Special Issue on Web Information Systems Engineering 2017*
Guest Editors: Lu Chen and Yunjun Gao

✉ Andrey Kravchenko
andrey.kravchenko@cs.ox.ac.uk

[1] Department of Computer Science, University of Oxford, Oxford, England

1 Introduction

A Web page is a giant puzzle, consisting of multiple blocks carrying different semantic and functional meanings, such as main content, navigation menu, advertisement, footer, header, and sidebar. Whilst it is easy for a human to distinguish these semantic and functional roles, it is a major challenge for the machine as it can only process HTML code and CSS layouts.

There are several important applications of Web block classification including automatic and semi-automatic wrapper induction [10, 11, 29, 32], assisting visually impaired people with navigating the website's internal content [14], help in the task of mobile Web browsing [26, 31], ad removal, Web page topic clustering [25], and the ubiquitous task of a Web search [4, 30]. Web block classification is also important for improving the accessibility of Web resources. In particular, adequate segmentation, which takes into account logical consistence and semantic role of elements on a Web page [21] plays an essential role in adaptive technologies and Web automation [2, 9].

In this manuscript we present a novel holistic approach to Web block classification. The motivation for this is to increase the accuracy of classifiers for blocks of different types located on the same page by considering these blocks in one single system and employing their mutual interactions. We also use the BER$_y$L system, which allows to define highly tailored feature sets for individual classifiers, whilst other state-of-the-art approaches aim to classify all blocks in their respective systems with one single set of features.

We provide an example of how a holistic view of a page can enhance the accuracy of classification for individual blocks as well as the overall performance of BER$_y$L in Figure 1.

Example 1 A system consists of two classifiers – headers and navigation menus – and the navigation menu classifier has much higher accuracy than the header classifier. The individual classifiers determine the block highlighted in blue at the top of Figure 1 correctly

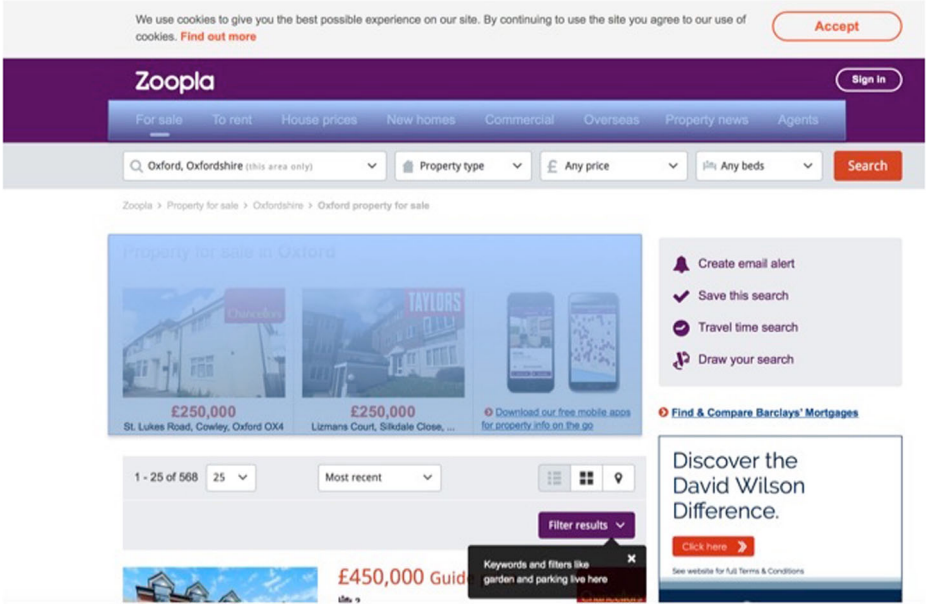

Figure 1 Mutual interaction between the block classifiers

as a navigation menu and the block highlighted in the middle as a header. However, our knowledge tells us that, in all domains, a navigation menu is located either in the header, just below the header, or in the sidebar. This constraint is violated by the given classifications and, therefore, because we trust the navigation menu classifier more, we can remove the classification of the header from the output as a False Positive.

The main contributions of our approach are the following.

1. We provide a holistic view of the page which gives a coherent perspective of the way it is split into Web blocks and the way these blocks interact.
2. The holistic view of the page is implemented through a system of mutual constraints.
3. We significantly enhance the BER$_y$L system for Web block classification [20] and provide a new use case for this system by employing global feature and template repositories and domain-specific knowledge encoded through logic-based rules to define the system of constraints.

2 Related work

A considerable amount of research has been undertaken in the field of Web block classification [3, 5, 6, 13, 15, 16, 18, 22–24, 26, 28, 29, 32]. However, most of the approaches taken in this research have attempted to classify a relatively small set of domain-independent blocks with a limited number of features, whereas we aim to develop a unified approach to classify both domain-dependent and domain-independent blocks. Furthermore, none of the papers with which we are familiar discussed the extensibility of their approaches to new block types and features.

Although most of these machine learning-based approaches require a significant amount of training, they usually reach a precision level below 70% [13, 15, 18, 22, 26]. The highest value of F_1 that any of these machine learning-based approaches reach (apart from [12] that achieves the F_1 of 99% but for a single very specific block type of "next" links and [19] that achieves the overall classification rate of 94% but for basic Web form elements within the transport domain and requires a complex training procedure) is around 85% [26].

The BER$_y$L system [20] achieves much higher levels of precision and recall. This can partly be explained by the fact that other approaches classify different blocks with the same set of features, whereas BER$_y$L employs individual feature sets for different types of blocks.

Finally, to our knowledge none of the state-of-the-art approaches considered a holistic view of Web pages, with classifiers for different block types interacting with one another in a single block classification system.

Although most of the current block classification approaches are based on machine learning methods, to our knowledge there is no universal approach to domain-dependent and domain-independent block classification, apart from the BER$_y$L system [20] discussed in more detail in Section 4.

Challenges of the Web block classification problem The challenges of the current Web block classification approaches can be divided into three main categories:

1. they cannot address a large variety of blocks;
2. domain-independent methods [16, 24, 26] run in performance limitations;
3. domain-dependent methods [3, 22, 29] have proven too costly in terms of supervision.

A detailed analysis of three seminal state-of-the-art classification methods [16, 22, 26] showed that none of these approaches employ domain-specific knowledge, that they all use a single set of features, and that they cannot perform fine-grained block classification. It is partially due to these reasons that state-of-the art approaches cannot achieve the necessary accuracy of classification on a wide range of blocks.

As discussed above, the failure of state-of-the-art classification systems in terms of covering diverse sets of block types can be partially explained by the fact that they all attempt to classify these blocks with one single set of features.

The BER$_y$L system presented in [20] addresses most of these challenges, in particular it meets the following requirements:

1. it covers a diverse range of domain-independent and domain-dependent blocks;
2. it achieves acceptable precision and recall results for each individual block in the classification system, and maximises the overall performance of the classification system;
3. it is adaptive to new block types and domains.

We therefore base our holistic approach to Web block classification on the BER$_y$L framework and implement the system of constraints in the BER$_y$L language. We discuss the contributions of the BER$_y$L system, as well as how we employ them in the holistic approach to Web block classification in Section 4.

3 Problem definition

Parts of Web pages are represented in the following as DOM tree fragments. We assume the readers are familiar with what a DOM tree is, so omit the detailed definition.

Definition 1 Let $\mathbb{BT}$ be a set of block types and $\mathbb{L}$ the corresponding set of labels. Given a Web page P with associated annotated DOM T_P, the problem of *Web block classification* is the problem of finding a mapping M from the set of sub-trees $T'_1, \ldots, T'_n$ of T_P to sets of labels from $\mathbb{L}$, such that each sub-tree is labelled with, at most, one label per block type, e.g. for each $BT \in \mathbb{BT}$ and sub-tree T_i, it must hold that $|M(T_i) \cap \mathbb{L}_{BT}| \leq 1$.

The above definition establishes the general classification problem. To apply standard classification algorithms, we need to map this general problem to a specific problem with a suitable feature space.

Example 2 Consider the Web page presented in Figure 2 and the corresponding DOM tree combined with the CSS representation of the page and textual annotations of DOM elements (e.g. numeric annotation or navigation-menu specific annotation, such as words ”Sale” or ”Rent” in case of navigation menus from the Real Estate domain). For this particular page we identify **(1)** a navigation menu that corresponds to the sub-tree rooted at the <ul> node, **(2)** a pagination bar that corresponds to the sub-tree rooted at the node and two sub-block “next” links of the pagination bar, **(3)** a numeric ‘next’ link, and **(4)** a non-numeric ‘next’ link corresponding to two <a> leaf nodes.

Unfortunately, individual classifiers only go so far. A key challenge in Web block classification is that the classifications of different block types affect each other; for example,

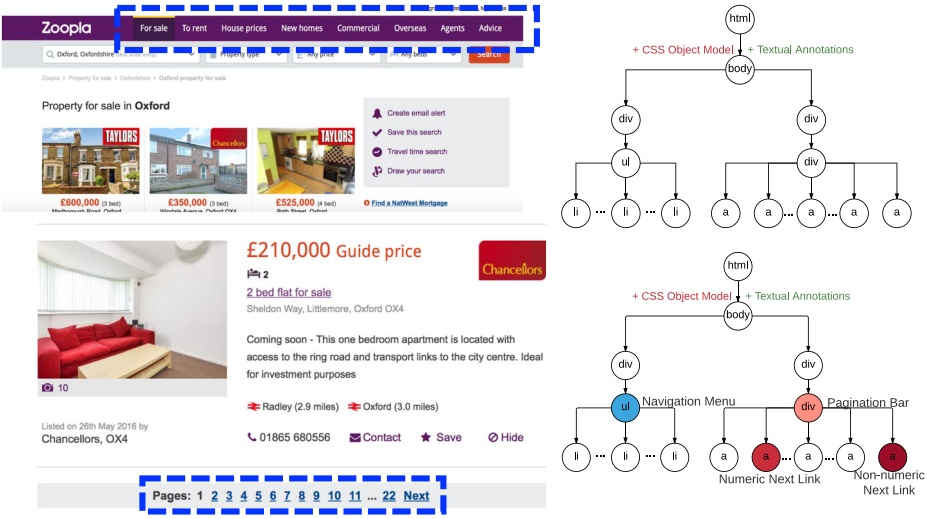

Figure 2 Navigation menu and pagination bar Web blocks identified on a Web page

if we determine that footers and sidebars are mutually exclusive and that both may be useful in determining the list of the main content paragraphs. Therefore we need to enhance a system for classification of individual block types, such as BER$_y$L [20], with a holistic approach to Web block classification (as explained in Section 1). The core mechanism of the holistic approach is a set of constraints to which the set of all Web block classifications of DOM nodes on a page must conform. BER$_y$L ensures that, if any of these constraints are violated, the existing classification is refined until all Web block classifications conform to the constraints.

It is now necessary to define a system of holistic constraints which indicates if a block classification is likely to be reasonable.

Definition 2 Let T_p be an annotated DOM and $\mathscr{C}$ a classification for T_P over a set of given block types $\mathbb{BT}$. Then, a *constraint* $\xi(T_P, \mathscr{C})$ over $\mathscr{C}$ and T_P is a first-order formula over three types of atoms:

– equality atoms between first-order variables and/or constants;
– atoms over annotated DOM relations, such as CHILD or CSS;
– atoms over the classification relation CLASS which relates a DOM node to each of the class labels from $\mathscr{C}$.

A non-empty set of such constraints over T_P and $\mathscr{C}$ is also referred to as a *system of constraints* $\Xi(T_P, \mathscr{C})$. We extend the notion of satisfaction from first-order formulae to constraints and systems of constraints in the obvious way.

We give examples of the actual constraints used by the BER$_y$L system in Section 4.

Definition 3 Let T_p be an annotated DOM, $\mathscr{C}$ a classification for T_P over a set of given block types $\mathbb{BT}$, and $\Xi(T_P, \mathscr{C})$ a corresponding system of constraints. Then $\mathscr{C}$ is a *valid classification* for Ξ if Ξ is satisfied by T_P and $\mathscr{C}$, i.e. if all constraints in Ξ are satisfied.

It is often the case that a system of constraints is not satisfied by a given set of classifications. We then need to find a subset of the given classification which satisfies all constraints in Ξ with that subset fulfilling some *optimisation criteria* and for which the system of constraints is satisfied. We use a combined optimisation criterion: first, we optimise the number of labels assigned to DOM nodes (i.e. the assignment size of the classification) and, in the case of ties, the relative coverage of the classified nodes on the page. The relative *coverage* of a set of nodes N in an annotated DOM T_P is the average of **(1)** the percentage of nodes occurring in at least one of the sub-trees rooted in nodes in N among all element nodes in T_P and **(2)** the percentage of the area covered by at least one node in N among the whole area of the page. We remove the DOM node with the minimal coverage ratio from the set of candidate DOM nodes on the intuition that this DOM node and its corresponding subtree contain less information about the page than other nodes in the set. We generalise the relative coverage from a set of nodes to a classification in the obvious way.

Definition 4 Let T_P be an annotated DOM, $\mathscr{C}$ a classification for T_P over a set of given block types $\mathbb{BT}$, and $\Xi(T_P, \mathscr{C})$ a corresponding system of constraints. The *maximal valid classification* problem is the optimisation problem of finding the classification $\mathscr{C}'$ for T_P over $\mathbb{BT}$ such that:

1. if n is a DOM node and $l \in \mathscr{C}'(n)$, then $l \in \mathscr{C}(n)$;
2. $\mathscr{C}'(n)$ has maximal assignment size among all classifications for which (1) holds;
3. $\mathscr{C}'(n)$ has the maximal relative coverage of T_P among all classifications for which (1) and (2) hold.

The underlying decision-related problem for the maximal valid classification is NP-complete. Suppose there are k classifications in the original set $\mathscr{C}$. We then need to in the worst case consider all 2^k subsets of the original set and find a subset of the maximal cardinality. This is equivalent to SAT problem where the k classifications are k variables and the true and false assignments are whether the classification is in the subset or not. We start with the full set $\mathscr{C}$ where all assignments are true and then consider all subsets of this set. SAT is reducible to 3-SAT which is an NP-complete problem [7, 17]. Therefore the maximal valid classification problem is also NP-complete. In Section 4 we introduce a heuristic approach to this problem of maximal valid classification that relaxes the problem from being NP-complete to a lower running time complexity, which makes it feasible to solve on Web pages containing up to thousands of DOM nodes. In Section 4 we also present concrete examples of how the holistic approach is implemented through the BER$_y$L system and evaluate how it improves the precision and recall of the system's constituent classifiers in Section 5.

4 Approach

Suppose our system supports classifiers for a set of different block types. How do we build a set of constraints that increases the accuracy for some classifiers whilst not dropping them for the others?

4.1 The BER$_y$L system for Web block classification

As discussed in Sections 1 and 3, our holistic approach to Web block classification is based on the BER$_y$L system for feature extraction and block classification [20]. We briefly

describe the main contributions of BER$_y$L and its approach to feature extraction with the Datalog-driven BER$_y$L language.

BER$_y$L addresses the challenges discussed in Section 3 through the following contributions. Contributions 1-2 and 3-4 refer to the quality of classification and generation aspects respectively.

1. It employs domain-specific knowledge to enhance the accuracy and performance of both domain-dependent (e.g. price, location, and floor plan images in the Real Estate domain) and domain-independent (e.g. navigation menus) classifiers.
2. It provides a global feature repository which allows users of BER$_y$L to easily add new features and block classifiers.
3. It encodes domain-specific knowledge through a set of logical rules; for example, that the navigation menu is always at the top or bottom of (and rarely to the side of) the main content area of the page for the Real Estate and Used Cars domains. Alternatively, for example, for the Real Estate domain, the floor plans can only be found in the main content area of the page.
4. It implements the global feature repository through baseline global features and template relations used to derive local block-specific features. BER$_y$L also uses textual annotations from the global annotation repository as features for Web block classification in order to further enhance the accuracy of individual classifiers.

BER$_y$L is a framework and an instrument for block classification. An expert defines the priority of each of the three phases. The first phase is related to feature engineering. This gives an expert the possibility to enrich the classifier with more features in case the second phase is required (sometimes the first phase, i.e. a set of rules for block classification, is enough). With additional features a simpler classification model or algorithm can be applicable. Regarding the second phase the expert can apply algorithms and models of different complexity, from decision tree algorithms to neural networks or more sophisticated approaches with collective classification. Currently BER$_y$L supports a limited set of machine learning algorithms, such as decision trees and support vector machines. Based on our experience we found out that introducing more complex features from the theory of topology and various spatial relations can require considerably less complex classification models. For instance, in our experiments related to the classification of navigation menus, footers, and headers, the use of a subset of topological relations and directions has a high performance for simpler algorithms such as C4.5 [27].

By default the machine learning phase has all the quantitative features which can be obtained from the DOM tree and CSS object model. These include coordinates, width, height, colour, font style, font size, and so on. This data might be enough for automatic classification, however it will require an expert to properly select a more sophisticated and advanced classification model. In case machine learning can tackle phase 3 (a system of constraints between block classifications), the latter can be omitted.

Sometimes it is easier to write a set of rules for feature extraction rather than find the best representative feature set for a particular classifier. In our system we can control which features go into the classifier, i.e. feature engineering allows to use a richer set of features. The feature engineering phase assists the classifier with providing a richer set of complex features which can be used in the training phase. We are by all means not limiting the classifier to the set of features we get by applying the rules but rather extend it, e.g. consider the vertical alignment feature for elements in the navigation menu, rather than just the coordinates of these elements, and leaving it to a deep learning model to find and automatically derive all features relevant for the navigation menu classifier.

4.2 The BER$_y$L language

BER$_y$L provides a tool to the users of the system that allows them to define powerful BER$_y$L component models in an easy and intuitively understandable, declarative way. This BER$_y$L language is a version of Datalog [1, 8] with a few extensions and is one of the key modules of our framework.

The BER$_y$L language is a dialect of Datalog that includes safe and stratified negation, aggregation, arithmetic, and comparison operators, denoted as Datalog$^{\neg,\mathsf{Agg},\mathsf{ALU}}$. The usual restrictions apply, i.e. all variables in a negation, arithmetic operation, aggregation, or comparison must also appear in some other (positive) context. Further, there may be no dependency cycles over such operators. In the context of the BER$_y$L system, this language is used in a certain way:

1. there is a number of input facts for representing the annotated DOM, as well as the output of previous components, such as the classification facts;
2. each program carries a distinguished set of output predicates and only entailed facts for those predicates are ever returned.

The BER$_y$L language also uses a number of syntactic extensions to ease the development of complex rule programs. The most important ones are *namespaces*, *parameters*, and the explicit distinction of *output predicates*. Parameters get instantiated to concrete values through the instantiation operator ($\mapsto$). All other intensional predicates are temporary ("helper") predicates only. For the purpose of brevity, we omit the precise definitions of these extensions.

Specifically, there are two classes of components and component types which are used heavily in the BER$_y$L language programs: components representing **(1)** relations and **(2)** features. These components operate on the universe $\mathscr{U}$ of DOM nodes and atomic values appearing in the DOM of a page. We typically use relations to either **(a)** distinguish sets of nodes, **(b)** relate sets of nodes to each other, or **(c)** attach additional information to nodes.

Declarative approach to feature extraction BER$_y$L uses a declarative approach to feature extraction wherever possible since that **(1)** allows to combine it with relations and global features which provide a succinct representation of the current feature set. This, in turn, allows to simplify the definition of new features through the employment of existing global features or relation instantiations and to learn new features by automatically finding the right combination of parameters for relation instantiations. **(2)** It is also much easier to learn Datalog predicates automatically than to learn procedural language programs, which is likely to come in useful in the large-scale block classification phase of BER$_y$L when we will have to infer new features automatically. In other cases which require the use of efficient libraries and data structures or intense numerical computation (e.g. features acquired from image processing), we employ a procedural approach for feature extraction implemented through Java.

Web block classification It does not seem feasible to solve the block classification problem through a set of logic-based rules, as it is often the case that there are many potential features which can be used to characterise a specific block type. However, only a few play a major role in uniquely distinguishing this block type from all others. Some of these features are continuous (e.g. the block's width and height), and it can be difficult for a human to manually specify accurate threshold boundaries. Hence, BER$_y$L uses a machine learning (ML) approach to Web block classification.

4.3 Holistic approach to Web block classification in BER$_y$L

BER$_y$L tries to derive a holistic interpretation of a Web page from individual classifiers. As such, it uses constraints to align the output of different classifiers with each other, find possible conflicts, and resolve them. As described in Section 3, classifiers are applied to features of DOM nodes (identifying a sub-tree in the input DOM) and produce (positive or negative) labels.

Constraints are built from predicates, which represent that a given node $n \in T_P$ is classified with a certain classification label $L \in L^{+}_{BT}$. They may also use other basic predicates which characterise DOM tree and CSS model properties, e.g. font size or CSS box width, but neither relations nor features (see Definition 2).

Example 3 An example of a constraint is that a header and a footer cannot be contained within each other. This constraint translates to the following BER$_y$L fragment (here and in the following we use $\Longrightarrow$ as syntactic sugar):

```
constraint::containment(Id,N,N′,true) ⇐
  param::instantiation_id(Id),
  relation::contained_in(RId,_,_),
  (cls::classification(N,header,header),
   cls::classification(N′,footer,footer)) ⟹
   ¬relation::contained_in(RId,N,N′),
   ¬relation::contained_in(RId,N′,N)).
```

This maps all pairs of nodes for which the above constraint holds to `true`. We call these pairs *witness pairs* of the constraint. An analogous rule maps all *violator pairs* for which the constraint is not satisfied to `false`. If there is at least one violator pair for a given constraint ξ , there is a conflict for this constraint on the given set of classifications, and BER$_y$L drops one of the violating classifications, and then checks whether ξ is now satisfiable and that there are no violator pairs produced by it.

We verify the constraints over the set of initial classifications in the `cls` namespace produced by all constituent classifiers of the BER$_y$L system. As mentioned above, if a constraint ξ between classification types BT_1 and BT_2 is not satisfied on some pair (N, N'), a heuristic must be applied to remove one or more of the violating classifications on the involved nodes. In the current version of BER$_y$L, we employ a heuristic that first maximises the number of newly satisfied constraints. If there is a tie, it breaks that tie by removing a violating classification on a node with smaller relative size. Size is measured as **(a)** the area of the CSS box of the node or **(b)** the total number of DOM nodes in sub-tree(s) rooted at this node.

4.4 Heuristic constraint resolution for BER$_y$L

We now describe the algorithm used to determine the subset of the original set of classifications that approximates our optimisation criteria with regard to the given system of constraints.

The input is the system of constraints that we check for being satisfied and the set of classifications produced for all block types present in this system of constraints.

Algorithm 1 The naive constraints satisfaction algorithm $SATISFYING-CLASSIFICATIONS$

Data: DOM tree T_p with classifications, a system of constraints $\Xi(T_P, \emptyset) = \{\xi_i | i \leq |\Xi(T_P, \emptyset)|\}$, and a classification $\mathscr{C}_{in}$ over a set of given block types $\mathbb{BT}$

Result: A classification $\mathscr{C}_{out}$ over a set of given block types $\mathbb{BT}$ on the nodes of the DOM tree T_p

```
for (i ← 0; i ≤ |Ξ(T_P, ∅)|; i++) do
    L_i ← GET-LABELS(ξ_i);
    C_i ← cls :: classification(_, BT, Label) ∧ Label ∈ L_BT ⊂ L_i;
    C_conflict ← EVALUATE-CONSTRAINT(ξ_i, C_i, T_P);
    if C_conflict ≠ ∅ then
        l ← REMOVE-RANDOM-CLASSIFICATION(C_conflict);
        C_trimmed ← C_in \ l;
        SATISFYING-CLASSIFICATIONS(Ξ(T_P, ∅), C_trimmed));
C_out ← C_in;
return C_out;
```

In the naive constraints resolution algorithm, we traverse the set of constraints, for each constraint ξ_i checking which classification labels are present in ξ_i (by calling function $GET\text{-}LABELS(\xi_i)$) and evaluating the constraint on the subset $\mathscr{C}_i \in \mathscr{C}_{in}$ of input classifications (acquired through $\mathscr{C}_i \leftarrow cls :: classification(X, BT, Label) \wedge Label \in \mathscr{L}_{BT} \subset \mathscr{L}_i$). If there is a set of classifications for which the given constraint is not satisfied (denoted as $\mathscr{C}_{conflict}$), we randomly remove one of these classifications from the conflicting set of classifications (by calling function $REMOVE\text{-}RANDOM\text{-}CLASSIFICATION(\mathscr{C}_{conflict})$) and re-run the algorithm on the reduced set of classifications. We repeat this process of trimming the set of classifications until all constraints get classified on the trimmed set. The running time complexity of this algorithm is $O(Eval(\xi, \mathscr{C}_{in}) \times |\Xi(T_P, \emptyset)| \times |\mathscr{C}_{in}|)$, where $Eval(\xi, \mathscr{C}_{in})$ is the worst-case data complexity of evaluating a constraint $\xi \in \Xi(T_P, \emptyset)$ on classification $\mathscr{C}_{in}$ (acquired through function $EVALUATE\text{-}CONSTRAINT(\xi_i, \mathscr{C}_{in}, T_P)$, which identifies all conflicts if there are any given a classification $\mathscr{C}_{in}$ over a DOM tree T_P).

Note that deleting one classification and making the conflicting constraint satisfiable can turn another constraint that had previously been satisfiable into unsatisfiable.

It is obvious that deleting classifications at random in the case of a conflict can eliminate a lot of True Positive classifications. We therefore need to define a more refined strategy for conflict resolution. We now give our greedy algorithm for constraints satisfaction based on the two heuristics introduced in Definition 4 of Section 3.

In the greedy algorithm, in case of a conflict within a constraint we apply two optimisation heuristics for deleting a conflicting classification: **(1)** we first check for classifications $\mathscr{C}_{optimal}$ the removal of which will make the maximal number of constraints that were not satisfied with it present in the classification set into valid ones (by calling function $GET\text{-}OPTIMAL\text{-}SUBSET(\mathscr{C}_{conflict}, constraints\text{-}resolved)$), and **(2)** in case there are multiple classifications the removal of which achieves the maximal number of constraints that get satisfied, we remove the one with the lowest coverage ratio as per Definition 4 in Section 3 (by calling function $MINIMAL\text{-}COVERAGE\text{-}RATIO(\mathscr{C}_{optimal})$). The running time of the greedy algorithm is $O(Eval^2(\xi, \mathscr{C}_{in}) \times |\Xi(T_P, \emptyset)|^2 \times |\mathscr{C}_{in}|^2)$. The

semantics of both the naive and greedy algorithms can be implemented in the form of a declarative program in the BER$_y$L language, with constraint resolution rules based on the sets of violator pairs produced by individual constraints.

Algorithm 2 The greedy constraints satisfaction algorithm $SATISFYING-CLASSIFICATIONS-GREEDY$

Data: DOM tree T_p , a system of constraints $\Xi(T_P, \emptyset) = \{\xi_i | i \leq |\Xi(T_P, \emptyset)|\}$ and a classification $\mathscr{C}_{in}$ over a set of given block types $\mathbb{BT}$

Result: A classification $\mathscr{C}_{out}$ over a set of given block types $\mathbb{BT}$ on the nodes of the DOM tree T_p

for $(i \leftarrow 0;\ i \leq |\Xi(T_P, \emptyset)|;\ i{+}{+})$ **do**
  $\mathscr{L}_i \leftarrow GET\text{-}LABELS(\xi_i)$;
  $\mathscr{C}_i \leftarrow cls :: classification(\_, BT, Label) \wedge Label \in \mathscr{L}_{BT} \subset \mathscr{L}_i$;
  $\mathscr{C}_{conflict} \leftarrow EVALUATE\text{-}CONSTRAINT(\xi_i, \mathscr{C}_i, T_P)$;
  if $\mathscr{C}_{conflict} \neq \emptyset$ **then**
    for $(j \leftarrow 0;\ j \leq |\mathscr{C}_{conflict}|;\ j{+}{+})$ **do**
      $\mathscr{C}_{deleted} = \{\mathscr{C}_{conflict}\}_j$;
      $constraints\text{-}resolved_j \leftarrow 0$;
      for $(k \leftarrow 0;\ k \leq |\Xi(T_P, \emptyset)|;\ k{+}{+})$ **do**
        $\mathscr{L}_k \leftarrow GET\text{-}LABELS(\xi_k)$;
        $\mathscr{C}_k \leftarrow cls :: classification(\_, BT, Label) \wedge Label \in \mathscr{L}_{BT} \subset \mathscr{L}_k$;
        $\mathscr{C}'_{conflict} \leftarrow EVALUATE\text{-}CONSTRAINT(\xi_k, \mathscr{C}_k, T_P)$;
        $\mathscr{C}''_{conflict} \leftarrow EVALUATE\text{-}CONSTRAINT(\xi_k, \mathscr{C}_k \setminus \mathscr{C}_{deleted}, T_P)$;
        if $(\mathscr{C}'_{conflict} \neq \emptyset \wedge \mathscr{C}''_{conflict} = \emptyset)$ **then**
          $constraints\text{-}resolved_j \leftarrow constraints\text{-}resolved_j + 1$;
        else if $(\mathscr{C}'_{conflict} = \emptyset \wedge \mathscr{C}''_{conflict} \neq \emptyset)$ **then**
          $constraints\text{-}resolved_j \leftarrow constraints\text{-}resolved_j - 1$;
    $\mathscr{C}_{optimal} \leftarrow GET\text{-}OPTIMAL\text{-}SUBSET(\mathscr{C}_{conflict}, constraints\text{-}resolved)$;
    if $|\mathscr{C}_{optimal}| = 1$ **then**
      $\mathscr{C}_{trimmed} \leftarrow \mathscr{C}_{in} \setminus \{\mathscr{C}_{optimal}\}$;
    else
      $l \leftarrow MINIMAL\text{-}COVERAGE\text{-}RATIO(\mathscr{C}_{optimal})$;
      $\mathscr{C}_{trimmed} \leftarrow \mathscr{C}_{in} \setminus l$;
    $SATISFYING\text{-}CLASSIFICATIONS\text{-}GREEDY(\Xi(T_P, \emptyset), \mathscr{C}_{trimmed})$;
$\mathscr{C}_{out} \leftarrow \mathscr{C}_{in}$;
return $\mathscr{C}_{out}$;

4.5 Case Study: BER$_y$L's System of Constraints

In the current version of BER$_y$L there are six constraints: **(1)** the visual containment constraint; **(2)** the width constraint; **(3)** the vertical distance constraint (the pagination must be

visually contained in either a header or a sidebar, the footer must have the same width as the header, the width of the sidebar can be at most half the width of the header, and the vertical distance between a header and a footer must be at least the height of a screen if the overall height of a page is bigger than one screen, and half the screen's height if not); **(4)** there is no visual and structural overlap between the CSS boxes and DOM sub-trees corresponding to the two blocks in question; **(5)** the contents of the structural elements of two blocks are disjointed (e.g. there cannot be a link leading to the same page that is present in both blocks); and **(6)** same or similar number of elements sharing the same characteristic property (footers, headers, and sidebars can have no visual or structural overlap, navigation menus and pagination bars can have no visual or structural overlap and all their links must be disjoint, and the number of pagination links in two pagination bars cannot differ by more than one in cases where their width is the same or almost the same).

We now present a sample of the system of constraints used for providing a holistic view of the page in the current version of BER$_y$L. For brevity purposes we limit the constraints to width and containment constraints. Note that we use the template approach described in [20] to define constraints in a generic way that is agnostic to concrete classifiers and classification types. We can then instantiate those constraints with concrete classifiers and classification types (such as `pagination_link` and `non_numeric_next_link`)

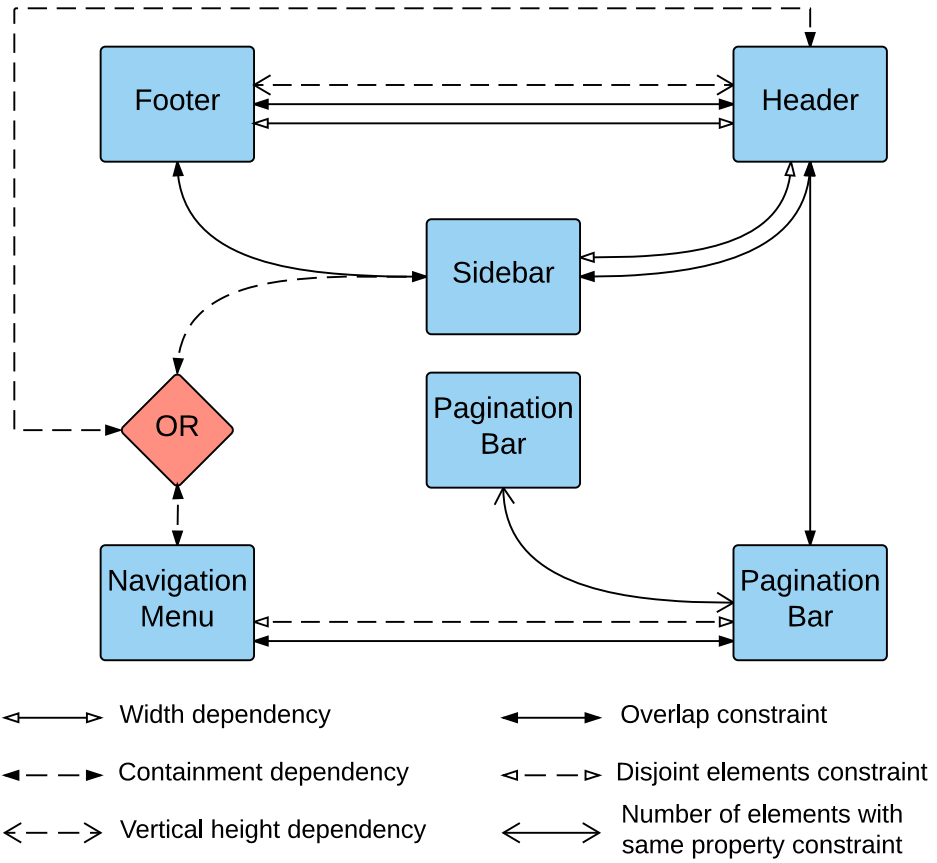

Figure 3 Holistic approach to Web block classification in BER$_y$L

specific to the current implementation of the BER$_y$L system. We also use the `constraint` namespace along with unique predicate names to define constraints. The visual representation of these constraints is given in Figure 3, and the BER$_y$L language rules are given below (for the purpose of conciseness we omit the exact definitions of relations used by constraints and introduce division into the BER$_y$L language, although the actual division is done in the arithmetic procedural component introduced in [20], and the same applies to real non-integer values, as well as $\Longrightarrow$ implication and $|Value|$ modulus operators):

```
constraint::width(Id,N1,N2,true) ⇐
  param::instantiation_id(Id),
  (cls::classification(N1,BT1,Label1),
   cls::classification(N2,BT2,Label2),
   param::firstBlockType(PId1,BT1),
   param::firstLabel(PId2,Label1),
   param::secondBlockType(PId3,BT2),
   param::secondLabel(PId4,Label2)  ⟹
   param::box_width(PId5,N1,Width1),
   param::box_width(PId6,N2,Width2),
   Width1 ≤ Width2/WidthFactorMin,
   Width2 ≤  Width1*WidthFactorMax,
   param::width_factor_min(PId7,WidthFactorMin),
   param::width_factor_max(PId8,WidthFactorMax)).

constraint::containment(Id,N1,N2,true) ⇐
  param::instantiation_id(Id), (cls::classification(N1,BT1,Label1),
  cls::classification(N2,BT2,Label2),
  param::firstBlockType(PId1,BT1),
  param::firstLabel(PId2,Label1),
  param::secondBlockType(PId3,BT2),
  param::secondLabel(PId4,Label2))  ⟹
  relation::visually_contained_in(Id1,N1,N2),
  relation::structurally_contained_in(Id1,N1,N2)).
```

We can then instantiate these generic constraints to fit with the requirements of the BER$_y$L system:

```
constraint::width[firstBlockType↦header, firstLabel↦header,
 secondBlockType↦footer, secondLabel↦footer,
 width_factor_min↦1, width_factor_max↦1]

constraint::width[firstBlockType↦header, firstLabel↦header,
 secondBlockType↦sidebar, secondLabel↦sidebar,
 width_factor_min↦2, width_factor_max↦#maxint]

constraint::containment[firstBlockType↦nav_menu,
  firstLabel↦nav_menu, secondBlockType↦[header,sidebar],
  secondLabel↦[header,sidebar]]

constraint::width[firstBlockType↦header, firstLabel↦header,
 secondBlockType↦footer, secondLabel↦footer,
 screen_height_adjustment↦0.8]
```

5 Evaluation of BER$_y$L's holistic approach

We first present the evaluation results for individual block classifiers without holistic approach being applied in Figure 4. We have evaluated precision, recall, and F_1 values for five classifiers (headers, footers, sidebars, navigation menus, and next links). This evaluation has been performed on 500 pages from four different domains (Real Estate, Used Cars, Online Retail, Blogs and Forums) randomly selected from a listings page (such as yell.com) or from a Google search. Google search tends to give preference to popular websites, but that should not affect the results presented here.

Rendering, which is an expensive operation, is needed for feature extraction on the DOM page, therefore it is rather a property of the system that the training and evaluation corpora are relatively small. We build the training and evaluation corpora in such a way that they are very diverse in the structural and visual representation of the Web block, e.g. horizontal and vertical navigation menus. As another example, consider the number of pages used for next link classification [12]. Although the evaluation corpus contained only 100 pages, it is the diversity of the training and evaluation corpora that produced an accurate and robust classifier capable of accurately identifying "next" links on any page, confirmed later in the large-scale evaluation of the DIADEM system, which included tens of thousands of pages. The BER$_y$L system was a part of this evaluation, and BER$_y$L's classifier was able to identify "next" links correctly in the vast majority of cases. Also, all pages typically follow a limited set of design patterns, so we do not need to evaluate our classifiers on larger sets of pages. In essence, the method is fully extendable to any set of pages despite the size of the evaluation corpus of 500 pages. It is also worth mentioning that each page contains up to 5,000 DOM elements, which are to be processed and analysed by BER$_y$L, and these elements are used in the training and classification phases.

All of the individual classifiers achieve acceptable precision, recall, and F_1 scores, with each classifier having precision and recall scores above 80% apart from the sidebar classifier, which has a precision score of just below 80%. This can be explained by the fact that a

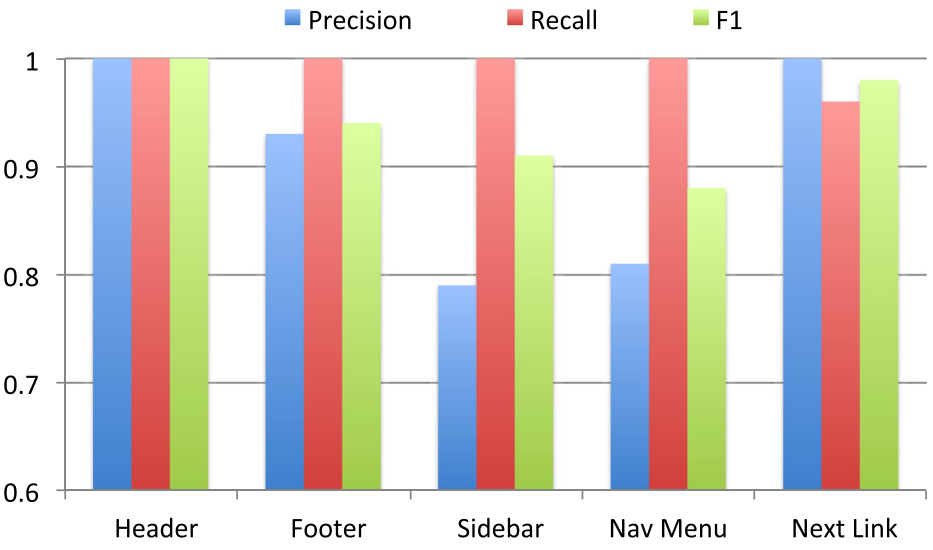

Figure 4 BER$_y$L's accuracy results on the five classifiers

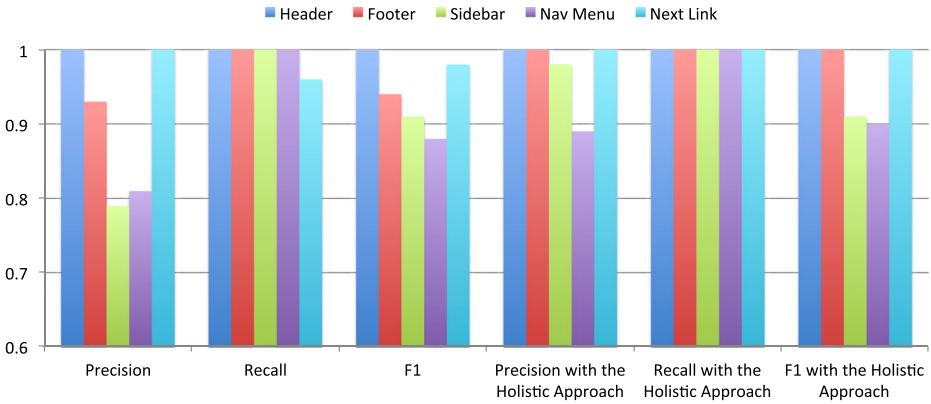

Figure 5 Standard per-block classification vs BER$_y$L's holistic approach

sidebar usually does not have obvious clues, such as the `next` annotation for non-numeric next links, and therefore it is harder to distinguish True Positives from False Positives and False Negatives. Note that the next links and header classifiers achieve a perfect precision of 100%. This can be explained by the fact that we use a highly tailored feature set that includes features specific to the next link block, and that headers are very distinct from other blocks, and our fairly simple set of six features used for the classification of this block is sufficient to achieve a perfect separation between header and non-header DOM nodes.

We have then performed the evaluation of the holistic approach we take in the BER$_y$L system, which is one of our key contributions. The system of constraints that BER$_y$L utilises is described in Section 4. As can be seen from Figure 5, we get significant improvements in both precision and recall for all classifiers, but an especially strong gain in precision for sidebars, navigation menus, and footers and a strong gain in recall for next links. This proves that our hypothesis is valid and even with a limited set of inter-classifier constraints (both domain-independent or the ones that involve domain knowledge) can significantly improve the accuracy of classification.

6 Conclusion

In this manuscript we present a novel approach to holistic classification of Web blocks where individual blocks are considered within the context of a Web page powered by the knowledge representation rules specific to that domain. We have evaluated our approach on a diverse set of pages from different domains and validated that it significantly improves precision and recall for the task of Web block classification.

There are multiple paths for future research that we would like to pursue, in particular: **(1)** evaluation of our holistic approach to Web block classification on more block types and domains, **(2)** integration of classification of domain-specific block types into the BER$_y$L system and the holistic approach to block classification, and **(3)** reformulation of our holistic approach to Web block classification in game-theoretical terms with payoffs being precision and recall values for individual constituent classifiers of the system, as well as the overall performance of the system (with the highest possible speed being preferred at no loss in precision and recall), and computation of the Nash equilibria for these multiple competing classifiers.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley Longman Publishing Co. Inc. (1995)
2. Ashok, V., Puzis, Y., Borodin, Y., Ramakrishnan, I.V.: Web screen reading automation assistance using semantic abstraction IUI'17 (2017)
3. Burget, R., Rudolfova, I.: Web page element classification based on visual features. In: 2009 First Asia conference on intelligent information and database systems (2009)
4. Cai, D., He, X., Wen, J., Ma, W.: Block-level link analysis. SIGIR'04 (2004)
5. Cao, Y., Niu, Z., Dai, L., Zhao, Y.: Extraction of informative blocks from Web pages ALPIT'08 (2008)
6. Chen, J., Zhou, B., Shi, J., Zhang, H., Fengwu, Q.: Function-Based Object Model Towards Website Adaptation. WWW'10 (2010)
7. Cook, S.A.: The complexity of theorem-proving procedures STOC'71 (1971)
8. de Moor, O., Gottlob, G., Furche, T., Sellers, A. (eds.): Datalog Reloaded, Revised Selected Papers. LNCS (2011)
9. Fayzrakhmanov, R.R.: Web Accessibility for the blind through visual representation analysis. PhD Thesis (2013)
10. Ferrara, E., De Meo, P., Fiumara, G., Baumgartner, R.: Web data extraction, applications, and techniques: a survey. Knowl.-Based Syst., vol. 70 (2014)
11. Furche, T., Gottlob, G., Grasso, G., Gunes, O., Guo, X., Kravchenko, A., Orsi, G., Schallhart, C., Sellers, A.J., Wang, C.: DIADEM: Domain-centric, intelligent, automated data extraction methodology. WWW'12 (2012)
12. Furche, T., Grasso, G., Kravchenko, A., Schallhart, C.: Turn the page automated traversal of paginated websites. ICWE'12 (2012)
13. Goel, A., Michelson, M., Knoblock, C.A.: Harvesting maps on the Web. Int. J. Doc. Anal. Recognit. **14**(4), 349–372 (2011)
14. Gupta, S., Kaiser, G., Neistadt, D., Grimm, P.: DOM-based content extraction of html documents. WWW 2003 (2003)
15. Kang, J., Choi, J.: Block classification of a Web page by using a combination of multiple classifiers. In: Fourth International Conference on Networked Computing and Advanced Information Management (2008)
16. Kang, J., Choi, J.: Recognising informative Web page blocks using visual segmentation for efficient information extraction. J. Univ. Comput. Sci. **14**(11), 1893–1910 (2008)
17. Karp, R.M.: Reducibility among combinatorial problems. Complexity of Computer Computations (1972)
18. Keller, M., Hartenstein, H.: GRABEX: A graph-based method for Web site block classification and its application on mining breadcrumb trails. In: 2013 IEEE/WIC/ACM International Conferences on Web Intelligence (WI) and Intelligent Agent Technology (IAT) (2013)
19. Kordomatis, I., Herzog, C., Fayzrakhmanov, R.R., Krüpl-Sypien, B., Holzinger, W., Baumgartner, R.: Web Object Identification for Web Automation and Meta-search. WIMS'13 (2013)
20. Kravchenko, A.: BER$_{\mathrm{y}}$L: A system for Web block classification. Transactions on Computational Science (2018)
21. Krüpl-Sypien, B., Fayzrakhmanov, R.R., Holzinger, W., Panzenböck, M., Baumgartner, R.: A versatile model for Web page representation, information extraction and content re-packaging. DocEng'11 (2011)
22. Lee, C.H., Kan, M., Lai, S.: Stylistic and Lexical Co-Training for Web Block Classification. WIDM'04 (2004)
23. Li, C., Dong, J., Chen, J.: Extraction of informative blocks from Web pages based on VIPS. J. Comput. Inf. Syst. **6**(1), 271–277 (2010)

24. Liu, W., Meng, X.: VIDE: A Vision-Based approach for deep Web data extraction. IEEE Trans. Knowl. Data Eng. **22**(3), 447–460 (2010)
25. Luo, P., Lin, F., Xiong, Y., Zhao, Y., Shi, Z.: Towards combining Web classification and Web information extraction: a case study. KDD'09 (2009)
26. Maekawa, T., Hara, T., Nishio, S.: Image classification for mobile Web browsing. WWW'06 (2006)
27. Ross Quinlan, J.: C4.5: Programs for machine learning. Mach. Learn. **16**(3), 235–240 (1993)
28. Vadrevu, S., Velipasaoglu, E.: Identifying primary content from Web page and its application to Web search ranking. WWW'11 (2011)
29. Wang, J., Chen, C., Wang, C., Pei, J., Bu, J., Guan, Z., Zhang, W.V.: Can we learn a Template-Independent wrapper for news article extraction from a single training site? KDD'09 (2009)
30. Wu, C., Zeng, G., Xu, G.: A Web page segmentation algorithm for extracting product information. In: Proceedings of the 2006 IEEE International Conference on Information Acquisition (2006)
31. Xiang, P., Yang, X., Shi, Y.: Web page segmentation based on gestalt theory 2007. In: IEEE International Conference on Multimedia and Expo (2007)
32. Zheng, S., Song, R., Wen, J., Giles, C.L.: Efficient record-level wrapper induction. CIKM'09 (2009)