Robust Web Data Extraction based on Unsupervised Visual Validation

Benoit Potvin and Roger Villemaire

Université du Québec à Montréal, Department of Computer Science,
Montréal, H3C 3P8, Canada

Abstract. Visual validation is the process of validating sets of extracted entities by means of visual information. The main advantage of visual validation is to make use of visual information for web information extraction without impacting on the robustness of extractors. In this paper, we show that unsupervised visual validation can be used to create robust web data extractors. More precisely, we evaluate the performance of visual validation on a corpus of visually heterogeneous documents. The selected extraction task consists in extracting the price, name, description, and SKU of unspecified products from unseen documents. Our corpus contains 1000 various products from 100 different sources, which we render public. Results also show that visual validation improves web data extraction even when the extractor is trained with visual features.

Keywords: Visual validation · Robustness · Isolation forest · Web information extraction · Classifiers

1 Introduction

Information Extraction (IE) is the task of extracting structured information from unstructured documents that were intended for human usage. IE is hence paramount to leverage the vast amount of information available in digital form. This is particularly true in the context of the *World Wide Web* (WWW), where an ever growing number of documents – mostly web pages – offer a broad spectrum of information whose ubiquitous availability defines the Information Age. Extracting information from web documents, known as *Web Information Extraction* (WIE), is also central to bridging the gap between the vastly unstructured web mostly experienced today, and the objective of a truly *Semantic Web* [2] that would allow much easier knowledge extraction.

The WWW is also nowadays the preferred medium for businesses to advertise services and goods, allowing would-be customers to compare products and prices. Business web pages are therefore designed to appeal to customers, emphasizing for instance, availability of new products or special discounts and pricing. As appealing as the page layout and presentation can be from a marketing point-of-view, this usually makes it even more difficult to devise methods that automatically extract factual information such as the product name, sale price, description, and the Stock Keeping Unit (SKU). Information extraction

methods must hence do their best to leverage the web page's structure, with its elements, their position and the way they are rendered (e.g. fonts and colors), in order to extract meaningful data values.

One can distinguish two broad approaches to web data extraction [7]. One can first leverage the semi-structured nature of HTML documents, for instance the Document Object Model (DOM) tree. Alternatively one can use Machine Learning (ML) methods, and train a classifier to predict whether an element belongs to a class of interest. In the former case, DOM-based (or tree-based) methods offer excellent extraction performance on previously seen templates but cannot extract information from novel documents [21]. In the latter case, ML methods usually require large manually labelled training sets, whose production is time-consuming and error-prone. Furthermore, the selection of discriminant features is a laborious task. In practice, ML-based methods are often website-dependent as generalizing websites absent from the training set usually sharply degrades performance [7,23]. Therefore extraction on novel documents, regardless of the selected approach, is highly challenging.

In this paper, we show that one can create a robust extractor for product information extraction by combining supervised classification with unsupervised methods. We consider the task of extracting a specific set of data types, i.e. the name, sale price, description, and SKU of unspecified products for sale from unseen web pages. In other words, we aim at creating a robust product information extractor that can extract a specific set of information for any product from any web page.

In fact, we introduce a general method to extend the range of supervised classification and create robust extractors. While a classifier's performance will sharply degrade as one move further away from the types of entities present in the training set, we show that Visual Validation, an unsupervised learning method, can restore classification performance. In the context of WIE, this means that one can rely on *flexible* features in order to extract information from as many documents as possible. The opposite scenario would be to use very specific features in order to obtain outstanding results on a limited set of documents or a specific task but extremely poor results on general tasks and large corpora. Visual Validation therefore enables to create extremely robust extractors with minimal effort in features selection and training set annotation, while relying on unsupervised methods to assure generalization to unseen entities. The proposed method uses visual information of web pages without being conducive to layout dependencies, which for instance is not the case when training classifiers with visual features. We will show that Visual Validation improves extraction results even if the original classifier has been trained with visual features.

2 Related Work

Web data extraction relies on information patterns (textual, structural or visual) in order to extract relevant entities [5]. The success of extraction therefore depends on the presence of identified patterns in analyzed documents [17,23]. In the

literature, we often encounter the terms "language-dependent" [22], "template-dependent" [21], or "layout-dependent" [11] to describe methods that respectively use textual, structural or visual patterns.

A major challenge for WIE is to create robust extractors that lead to good performance. Robustness is the degree of insensitivity of an extractor to web page modifications, including changes in the syntax, template, formatting and layout [16]. In other words, it is the ability of an extractor to rely on discriminant patterns that are unlikely to change, whether a document is modified or extraction is performed on novel documents.

When designing an extractor, one will have to find the right balance between the performance and the robustness of the extractor. For instance, one can easily create an extractor that will obtain perfect results on few documents but also extremely poor results on novel documents. We refer to this tradeoff as the performance-robustness tradeoff [17].

Robustness has been extensively studied in the literature with a special focus on DOM-based methods [7]. These methods rely on structural patterns, i.e. regularities across the DOM tree (template-dependent). As a consequence, DOM-based methods lead to excellent performance at the expense of robustness [21]. Several techniques have been suggested in order to automatically adapt the relevant regularities into the new DOM tree [6]. This however only concerns the maintenance of extractors and do not make these methods applicable to unseen documents.

Visual information plays an important role in the definition of web documents. The use of discriminant visual regularities for WIE can improve extraction results [4,14,1,9]. However, these improvements are often obtained at the expense of robustness [19]. In fact, visual regularities are rarely expected to be consistent across all web documents. Therefore, visual information is either exploited in sets of documents that have visual similarities [1], or in specific tasks that rely on an object with similar visual cues across all documents, such as a table [8], a text block [12], etc. In the first case, the extraction process relies on a set of visual regularities and robustness is therefore impacted. In the second case, most visual information is ignored.

Visual Validation, an unsupervised learning method that rely on the visual information of extracted entities, has recently been used to improve information extraction without impacting on the robustness of extractors [19]. The method has only been used on documents with visual similarities. In this paper, we evaluate the performance of Visual Validation on a corpus of heterogeneous documents and show the positive impact of the method for robust extraction. This is, to our knowledge, the first example of a method where visual information is broadly used for robust web information extraction.

3 Background

3.1 Web data extraction as a classification problem

In Supervised ML a dataset of entities labelled by their correct class is used to devise a *classifier* in order to predict the class of previously unseen entities. A web data extraction task can be seen as a classification problem where a classifier is trained to predict the classes of entities that one wants to extract [20]. In this setting, entities can be tokens that are generated from the textual content (common in NLP tasks), DOM nodes or any other relevant entity. The DOM is a W3C recommendation that allows programs and scripts to dynamically access and update the content, structure, and style of documents. Since it is the most common approach to manipulate web documents, we will solely consider entities that are DOM nodes. Training is hence done on a dataset of DOM nodes labeled with their correct class and the objective is to extract relevant nodes that belong to the classes of interest in each document.

XPath is the W3C recommended and preferred tool to address nodes of the DOM and has been largely used in WIE systems and web annotation tools. Each node of a document can be located through its XPath expression and its textual content, style properties, and position can then be retrieved (e.g. with a headless browser). Based on a set of discriminant features, a vector of values describes each node. The classifier will hence map each vector to its corresponding class. The training set is formed of labeled web documents, which is a set of nodes' vectors labeled with the class to which the node belongs.

In order to predict multiple classes one can either use a specific (binary) classifier for each class or a single (multinomial) classifier predicting one of many classes. For instance, in order to predict product name, sale price, description and SKU one can have four classifiers, one for each class. On the other hand, a multinomial classifier takes an entity and returns the class, which would be in this case either "product name", "sale price", "description", "SKU" or "None of these classes".

A web document can contain several thousand DOM nodes. Extraction is done by having the classifier process each node in order to extract those that are relevant. This method is usually used in settings where one intends to extract many nodes. For instance, a content extractor should identify all nodes containing a part of the main content, and the objective is to extract as many nodes of the main content as possible. In our case, we extract exactly four data values, hence four nodes, from each page. We therefore simply rely on the classifier's underlying real-valued output [24] and select the node with the highest output for each of the four classes "product name", "sale price", "description" and "SKU".

3.2 Visual validation

Visual Validation (VV) is the process of validating sets of extracted entities by means of visual information [19]. It aims at identifying false positive entities from those returned by a web data extractor. VV consists in three steps. First,

visual information of extracted entities is obtained. A common approach is to use XPath expressions to locate each entity and obtain its visual characteristics. Secondly, visual information is used to identify visual outliers in the set of extracted entities. Finally, visual outliers are discarded.

VV assumes that visual outliers in a set of extracted entities are false positives. A web data extractor is designed to extract a specific type of information and resulting sets therefore contain similar entities, at the exception of false positives. In visually-rich documents such as web pages and PDFs, it is common that similar entities have visual similarities. Discriminant visual patterns are often used to identify relevant entities and facilitate web data extraction, in the same manner that visual information helps human users to navigate across a web page and understand its content [4,14,7,9]. VV however uses visual information in a specific way. VV does not aim at identifying visual patterns across true positive entities but only assumes that such patterns exist in order to state that visual outliers, i.e. entities that visually differ from the norm, are false positives.

Visual outliers are furthermore point anomalies, i.e. single anomalous instances that differ from the norm. Point anomalies have been extensively studied in the domain of anomaly detection. In order to determine point anomalies, one usually uses unsupervised anomaly detection methods.

Isolation Forest (iForest) [15] is an unsupervised anomaly detection algorithm that makes use of the concept of "isolation", i.e. the process of separating an instance from all other instances. The rationale behind iForest is that anomalies are rare items that are different, and therefore they are more susceptible to isolation than normal instances. iForest consists of two steps. In the first step, random decision trees (called iTrees) are constructed by recursively partitioning sub-samples of the given dataset until instances are isolated or a specific tree height has been reached. In the evaluation step, all instances of the dataset are passed through each iTrees, and anomaly scores are computed based on the average path length. Anomalies are instances with shorter path lengths (as they are easier to isolate), and the average tree height is sufficient for anomaly detection. This enables iForest to use sub-sampling to an extent that is not feasible with other methods. iForest achieves high detection performance with small sub-sampling sizes. iForest can hence process large datasets in linear time with low memory requirements. iForest is therefore particularly well suited for web data extraction tasks.

4 Methodology

Our method goes as follows 1. We first train a classifier to extract product information from unseen web pages. Secondly, processing the whole dataset, node by node, we select for each page the nodes with the highest output for each of the four classes "product name", "sale price", "description" and "SKU". Finally, we use VV to filter out visual outliers from the set of all selected nodes.

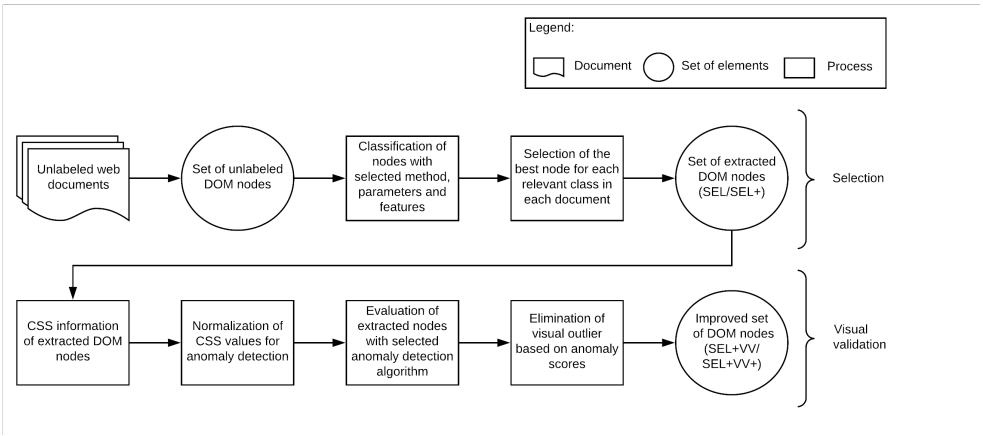

Fig. 1. The successive steps of the proposed method

4.1 Selection

In the second step of our approach, a name, price, description, and SKU are selected for each page. Note that at this step precision, recall, and F_1 score are the same, as precision and recall are equal. This is the case as four nodes are selected for each document (true and false positive), while there is exactly the same number of relevant elements (true positives and false negatives).

4.2 Visual Validation

Note however that some nodes selected in the second step will be removed in the last step, leaving some pages with missing data values. This is consistent with the fact that these missing values have been removed by VV since they are outliers, and we hence expect them to be erroneous.

Although this is not the case in this paper's experiments, data values are often absent of web pages. For example, not all product web pages have a SKU. Contrary to DOM-based methods, a classifier will fill all data slots, which can lead to a considerable amount of false positive entities. VV could help to mitigate this problem by eliminating false positive entities.

Finally, note that VV filters out elements selected in the second step of our method. The number of retrieved true positives can hence only decrease, and recall cannot improve. In our experiments, we will hence investigate the possibility of increasing precision and F_1 score. Since recall cannot improve, increasing F_1 score means that the increase in precision compensated for the decrease in recall.

5 Experimental Setup

We train the classifier on a set of shallow features. Shallow features have previously been used to create domain- and language-independent classifiers [13]. The selected shallow features for this task are: 1) the string length, 2) the number of digits in the string, 3) the number of whitespace characters in the string, 4) the number of line breaks in the string, 5) the number of currency symbols in the string, 6) the number of hash symbols in the string and 7) the XPath expression depth level (i.e. the number of slashes). For simplicity, the same set of features is used for all classes.

5.1 Dataset

To our knowledge, there is no available corpus of web documents where visual information, as it is rendered on the user's screen is available i.e. that includes CSS, JavaScript and other third party files that can change the appearance of a web page. Therefore, we annotated our own corpus that one can access online.[1] The dataset contains 1000 labeled web pages from 100 e-commerce websites. For each website, ten products have been randomly selected. The corpus includes a large selection of products from several countries and in different currencies. Examples of products are: clothes, musical instruments, bicycles, car parts, food, furniture, cosmetics, art pieces, games, tools, sport goods, computers, etc. On each page the same four entities have been identified: the name, price, description and SKU of the main product for sale.

To these 4000 *positive* labeled entities, i.e. 1000 entities for each of the four classes (one per page), we add 4000 *negative* entities corresponding to the "None of these classes" class. As each page contains a unique node corresponding to the four classes of interest, any other node can randomly be chosen for the fifth class "None of these classes". The annotation set hence includes 8000 labeled entities: 4000 positives cases and 4000 negative cases.

Websites were found using Google search engine with keywords such as "on sale", "buy online", etc. All web pages are in English.

5.2 Selection

At the second step, we select the DOM nodes that have the best scores for the classifier, one for each class in each document. When multiple DOM nodes have the same high score, a node is selected randomly. Precision is computed. We will refer to this result as SEL. As explained in Section 4.1, precision, recall, and F_1 score are equal in this case.

[1] Files can be downloaded at this link: https://drive.google.com/drive/folders/1GYU6ZgZOXsNq4–F7o8v3qjr47DcLvK3.

5.3 Visual Validation

We use iForest for VV, developing a single anomaly model for all classes. We evaluated the case where a model is learned for each class, e.g. by evaluating price entities with a model trained only with prices. However, both approaches lead to similar results, and we therefore selected the simpler approach. For VV, we furthermore used the following visual properties: the position, font-color, font-size, and font-weight.

DOM nodes are hence sorted in ascending order according to their computed anomaly scores. For iForest a low score denotes an anomalous instance. Visual outliers are successively deleted and precision, recall, and F_1 score are computed. We will refer to these results as the SEL+VV results.

5.4 Tools and Implementation

Web pages have been annotated with Pundit Annotator [10]. Pundit can be used through a Chrome extension, and annotations are exported in the JSON-LD format. Labeled elements are located through XPointer, and their CSS characteristics and position are extracted with Headless Chrome Node API Puppeteer.

Classifiers' implementations are those from scikit-learn Python library [18], as well as iForest anomaly detection algorithm. In order to obtain values in the range [0,1] the "predict_proba" function from scikit-learn is used. If not indicated, all parameters are those by default.

6 Experimental Evaluation

Using 10-fold cross-validation on our labeled dataset, we readily identified that the most promising classifiers for the selection step are Random Forest (RF) [3] and Multi-Layer Perceptron (MLP) classifiers.[2]

In this section we first compare SEL and SEL+VV results with these two types of classifiers (for selection) and iForest for VV (Figure 2 and Table 1). Finally, we retrain the classifiers using visual features and then compare the same values, which we will identify by SEL+ and SEL+VV+ to distinguish them from the previous results (Figure 3 and Table 2).

6.1 Training Classifiers

In order to get fair results on the classifiers' ability to extract information from unseen documents we divide our dataset in four sets where documents from a same website are always gathered in the same set (and absent from other sets). This way, we make sure that documents from a same website will not be at the same time in the training and test datasets, which could drastically improve

[2] Tested classifiers for this task were: a Gaussian Naive Bayes classifier, a k-nearest neighbor classifier, a multi-class SVM classifier (one-versus-one), and the two selected classifiers.

results. In our case where 1000 documents have been annotated, we obtain four sets of documents, each containing 250 documents from 25 websites. There are four instances of each classifier (MLP and RF), each instance being trained on three distinct sets in order to evaluate it on the fourth one containing only novel documents. Overall results for a classifier are the average of the results obtained with each instance, making sure that the documents in one set are not easier to extract than documents in other sets.

In this setting, we trained seven classifiers, i.e. a random forest (RF) classifier and six multi-layer perceptron (MLP) classifiers with respectively one, three and ten hidden layers (100 neurons) with "tanh" and "ReLU" activation functions (i.e. 1 hidden layer with ReLU, 1 hidden layer with tanh, 3 hidden layers with Relu, 3 hidden layers with tanh, etc.).

Therefore, there are 28 instances of classifiers, i.e. seven classifiers trained in four instances. Each classifier predicts the classes of all DOM nodes contained in its test set.

MLP classifiers had similar performance across all versions, independently of the number of hidden layers, the number of neurons on each layer, or the selected activation function. For simplicity, we only exhibit the results obtained with one hidden layer (100 neurons) MLP using "tanh" activation function.

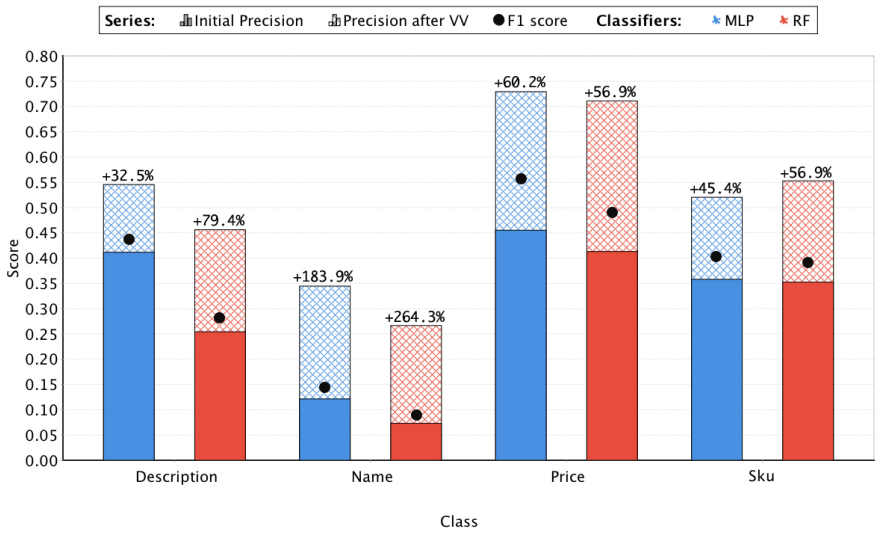

Fig. 2. SEL (solid bars) and SEL+VV (crosshatched bars) results

6.2 Comparing SEL and SEL+VV

Initial results (SEL) for both classifiers (MLP and RF) have been improved with VV (SEL+VV) as shown in Figure 2. Solid bars denote the initial classifiers'

results (SEL) while the crosshatched areas show improvements obtained with VV (SEL+VV). In the case of the SEL results, as explained in Section 4.1, precision, recall, and F_1 score are equal. Black dots denote the F_1 scores for SEL+VV. The number on top of each bar shows the percentage by which the precision is increased (SEL to SEL+VV). F_1 percentage increases are presented in table 1.

Table 1. Comparison of F_1 scores between SEL and SEL+VV

	Description		**Name**		**Price**		**SKU**	
	MLP	RF	MLP	RF	MLP	RF	MLP	RF
SEL	0.4118	0.2543	0.1214	0.0732	0.4554	0.4133	0.3580	0.3526
SEL+VV	0.4357 (+5.81%)	0.2807 (+10.38%)	0.1429 (+17.67%)	0.0873 (+19.23%)	0.5574 (+22.34%)	0.4891 (+18.34%)	0.4017 (+12.19%)	0.3892 (+10.39%)

We note a consistent improvement both in precision and F_1 score from SEL to SEL+VV, attesting to the improvement provided by VV.

6.3 Comparing SEL+ and SEL+VV+ and classification to VV

The results of Section 6.2 were obtained with unsupervised VV on classifiers that have not been trained with visual information. In this section, we train classifiers with a set of features to which the visual features used for VV has been added, and again compare selection alone with selection and VV, i.e., SEL+ and SEL+VV+ results. These results are particularly interesting as they allow us to evaluate whether VV can improve classifiers that have been trained with the same visual information.

Indeed, comparing SEL+ and SEL+VV+ in Figure 3 and Table 2 shows that VV consistently improves results, even if the same set of visual information used for VV is already available in the classifier's training set. In fact, this reflects the fundamental difference between leveraging *similarity* and leveraging *dissimilarity*. The classifier does not use visual information in the same way as VV. Extracting entities based on the fact that similar entities share visual patterns indeed differs from identifying visual outliers based on the assumptions that dissimilar entities have visual dissimilarities.

Table 2. Comparison of F_1 scores between SEL, SEL+ and SEL+VV+

	Description		**Name**		**Price**		**SKU**	
	MLP	RF	MLP	RF	MLP	RF	MLP	RF
SEL	0.4118	0.2543	0.1214	0.0732	0.4554	0.4133	0.3580	0.3526
SEL+	**0.2623 (-36.30%)**	0.3184 (+25.18%)	0.2553 (+110.19%)	0.6306 (+761.57%)	**0.3429 (-24.71%)**	0.7302 (+76.69%)	**0.1596 (-55.42%)**	0.4435 (+25.78%)
SEL+VV+	0.2869 (+9.38%)	0.3364 (+5.66%)	0.2965 (+16.14%)	0.6660 (+5.61%)	0.3747 (+9.27%)	0.7745 (+6.07%)	0.1799 (+12.71%)	0.4661 (+5.10%)

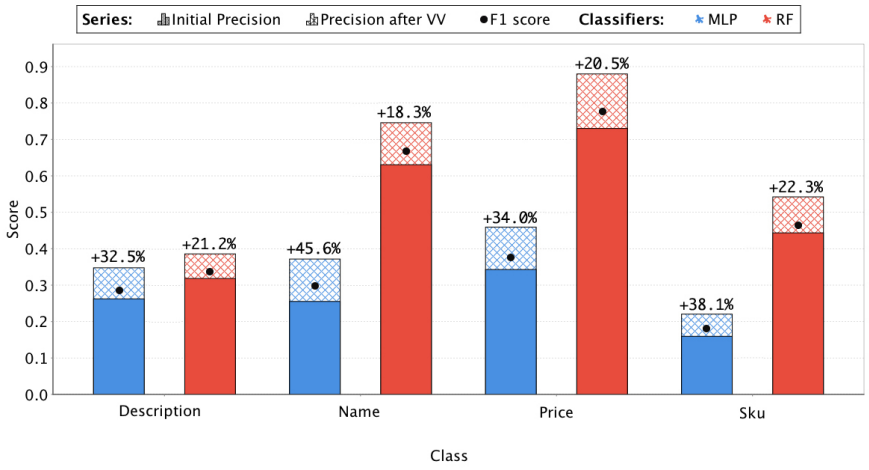

Fig. 3. SEL+ and SEL+VV+ results

Comparing SEL+ to SEL in Table 2, we note a large improvement in F_1 scores for RF and a large degradation for MLP, except in the case of the name class. In fact, for RF, SEL+ always outperform SEL+VV, while for MLP this is the case only for the name class. Therefore, using more features to train the classifier can lead to large performance improvement, but this can sharply depend on the type of classifier.

7 Conclusion

In this paper, we showed that unsupervised visual validation can improve web data extraction from heterogeneous documents. First, unsupervised VV can be combined with supervised classifiers that have been trained with shallow features in order to obtain robust extractors. This strategy allows to reduce the workload that is usually required to create supervised classifiers with acceptable performance. For instance, in our experiments the same features are used for all classes, and textual information is only analyzed quantitatively. Secondly, we compared the results obtained by using the same set of visual information for VV and for training the classifier. We hence determined that VV can still improve extraction results in this setting, which shows that selection, and VV use visual information in a fundamentally different way. Finally, we showed that iForest can effectively be used to perform VV.

Acknowledgments. The authors gratefully acknowledge the financial support of the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

1. Apostolova, E., Pourashraf, P., Sack, J.: Digital leafleting: Extracting structured data from multimedia online flyers. In: Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 283–292 (2015)
2. Berners-Lee, T., Hendler, J., Lassila, O., et al.: The semantic web. Scientific american **284**(5), 28–37 (2001)
3. Breiman, L.: Random forests. Machine Learning **45**(1), 5–32 (2001)
4. Burget, R., Rudolfova, I.: Web page element classification based on visual features. In: Intelligent Information and Database Systems, 2009. ACIIDS 2009. First Asian Conference on. pp. 67–72. IEEE (2009)
5. Chang, C.H., Kayed, M., Girgis, M.R., Shaalan, K.F.: A survey of web information extraction systems. IEEE transactions on knowledge and data engineering **18**(10), 1411–1428 (2006)
6. Ferrara, E., Baumgartner, R.: Automatic wrapper adaptation by tree edit distance matching. In: Combinations of Intelligent Methods and Applications, pp. 41–54. Springer (2011)
7. Ferrara, E., De Meo, P., Fiumara, G., Baumgartner, R.: Web data extraction, applications and techniques: A survey. Knowledge-Based Systems **70**, 301–323 (2014)
8. Gatterbauer, W., Bohunsky, P., Herzog, M., Krüpl, B., Pollak, B.: Towards domain-independent information extraction from web tables. In: Proceedings of the 16th international conference on World Wide Web. pp. 71–80. ACM (2007)
9. Gogar, T., Hubacek, O., Sedivy, J.: Deep neural networks for web page information extraction. In: IFIP International Conference on Artificial Intelligence Applications and Innovations. pp. 154–163. Springer (2016)
10. Grassi, M., Morbidoni, C., Nucci, M., Fonda, S., Ledda, G.: Pundit: Semantically structured annotations for web contents and digital libraries. In: SDA. pp. 49–60 (2012)
11. Han, H., Noro, T., Tokuda, T.: An automatic web news article contents extraction system based on rss feeds. Journal of Web Engineering **8**(3), 268 (2009)
12. Kang, J., Choi, J.: Detecting informative web page blocks for efficient information extraction using visual block segmentation. In: Information Technology Convergence, 2007. ISITC 2007. International Symposium on. pp. 306–310. IEEE (2007)
13. Kohlschütter, C., Fankhauser, P., Nejdl, W.: Boilerplate detection using shallow text features. In: Proceedings of the third ACM international conference on Web search and data mining. pp. 441–450. ACM (2010)
14. 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. In: Proceedings of the 11th ACM symposium on Document engineering. pp. 129–138. ACM (2011)
15. Liu, F.T., Ting, K.M., Zhou, Z.H.: Isolation forest. In: Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on. pp. 413–422. IEEE (2008)
16. Liu, L., Özsu, M.T.: Encyclopedia of database systems, vol. 6. Springer New York, NY, USA: (2009)
17. Parameswaran, A., Dalvi, N., Garcia-Molina, H., Rastogi, R.: Optimal schemes for robust web extraction. Proceedings of the VLDB Conference **4**(11) (September 2011)

18. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al.: Scikit-learn: Machine learning in python. Journal of machine learning research **12**(Oct), 2825–2830 (2011)
19. Potvin, B., Villemaire, R.: When different is wrong: Visual unsupervised validation for web information extraction. In: International Conference on Machine Learning and Data Mining in Pattern Recognition. pp. 132–146. Springer (2018)
20. Tang, J., Hong, M., Zhang, D.L., Li, J.: Information extraction: Methodologies and applications. In: Emerging Technologies of Text Mining: Techniques and Applications, pp. 1–33. IGI Global (2008)
21. 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? In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 1345–1354. ACM (2009)
22. Wang, R.C., Cohen, W.W.: Language-independent set expansion of named entities using the web. In: icdm. pp. 342–350. IEEE (2007)
23. Weninger, T., Palacios, R., Crescenzi, V., Gottron, T., Merialdo, P.: Web content extraction: a metaanalysis of its past and thoughts on its future. ACM SIGKDD Explorations Newsletter **17**(2), 17–23 (2016)
24. Witten, I.H., Frank, E., Hall, M.A., Pal, C.J.: Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann (2016)