2013 IEEE 10th International Conference on e-Business Engineering

# Detection of E-Commerce Systems with Sparse Features and Supervised Classification

Kurt Uwe Stoll and Martin Hepp
E-Business & Web Science Research Group
Universität der Bundeswehr München
Werner-Heisenberg-Weg 39, 85577 Neubiberg, Germany
Email: uwe.stoll@unibw.de, mhepp@computer.org

***Abstract*—Enriching web shop pages with structured data has recently become popular in e-commerce. It is mainly driven by search engines favouring those pages. While structured data in e-commerce is mainly generated automatically by shop extensions, this data covers only a small share of the market, resulting in a major hamper for applications operating on aggregated data. In this context, more than 90% of product detail pages on the web are generated by only 7 e-commerce systems. Meanwhile, little research addresses methods to automatically detect e-commerce systems. Automated detection would allow to design system-specific extractors able to grow the amount of structured data in e-commerce. Therefore, we propose a novel approach to this problem, which filters features generated from HTML tag attributes with an e-commerce specific white list. We evaluate 6 classification algorithms on the problem and discuss computational effort. We can show that this approach is capable of detecting the 6 most important e-commerce systems with a F1-score of 0.9 by analyzing only one HTML page per web shop. We evaluate our findings on an independent dataset and on reference shop sites.**

***Keywords*—*e-commerce systems, supervised machine learning, web page classification***

## I. Introduction

Enriching web shop pages with structured data using RDFa[1], Microdata[2], or Microformats[3] has recently become a popular method in e-commerce. The GoodRelations web ontology allows to express a manifold of e-commerce cases. [1]. Embedding structured data in e-commerce pages has recently matured to a first-choice SEO strategy, as the search engines Google, Bing, Yahoo and Yandex officially support GoodRelations in the schema.org consortium [2].

For shop owners, the use of structured data is mainly driven by the promise of search engines to enhance result pages with price, availability and reviews. There is evidence that those enhanced search results lead to higher click-through rates, which in turn lead to higher conversions [3].

There is a multitude of terms for software that allows merchants to run online shops, like shop software package, shopping cart, or shop system. As the software mostly integrates a multitude of functions like payment or inventory control related to ecommerce, we would like to emplay the term e-commerce system (ECS) cohenrently throughout the paper to refer to this kind of software. Structured data in e-commerce is mainly generated by Web shop extensions. Those extensions allow shop owners to easily integrate structured data. Many shop owners run their sites on standard e-commerce systems like Magento[4], Prestashop[5] or Virtuemart[6]. This allows for the development of shop extensions for structured data generation with high reach [4]. Meanwhile, the overall market coverage of those extensions is relatively low. This is especially a limitation for applications that build on aggregated data. While there are about 20.000 Web shops that generate structured data, this only covers a relatively low share of the Web.

Meanwhile, there is little research exploiting Web information extraction [5] to grow the amount of structured data in e-commerce. In this context, Stoll et. al 2013 [6] showed that only 7 ECS generate more than 90% of the product pages on the Web. Manual inspection of ECS showed distinct patterns in the HTML pages the shops generate. We expect those patterns to be useful for designing ECS-specific extractors.

Therefore, we propose a novel approach to automatically detect ECS. It operates on the ”id” and ”class” attributes of HTML tags. After filtering these features with a white list, only allowing those containing the strings ”product”, ”price” and ”cart”, we apply supervised classification algorithms. With this approach, we can detect 6 ECS with a combined F1-score of 0.9.

The paper is structured as follows: In section two, we discuss related work, in section 3, we describe our approach and provide implementation details. In section 4, we discuss the results of the experiment. In section 5, we evaluate our approach on an independent dataset and additionally on reference web shops. We discuss limitations in section 6. Finally, we summarize our findings in section 7.

## II. Related Work

There are two dimensions of work related to this paper. The problem is situated in web page classification, whereas the methods we use are in the supervised classification sub field of machine learning.

### A. Web Page Classification

*1) Fundamental problems:* According to Qi and Davison 2007 [7], who provide a encompassing survey on the topic, Web page classification aims at assigning predefined category

[1] http://www.w3.org/TR/xhtml-rdfa-primer
[2] http://www.w3.org/TR/microdata/
[3] http://microformats.org/about
[4] http://www.magentocommerce.com/
[5] http://www.prestashop.com/
[6] http://virtuemart.net/

978-0-7695-5111-1/13 $26.00 © 2013 IEEE
DOI 10.1109/ICEBE.2013.30

labels to Web pages. Web page classification can be divided into three subtopics. (1) Subject classification aims at detecting the topic of a Web page, e.g. ”sports”, ”culture”, ”business”. (2) Functional classification tries to identify the role of a Web page, e.g. ”Business home page”, ”Personal blog entry” or ”Web shop product page”. (3) Sentiment classification targets opinions and attitudes conveyed with a page. As our approach does not fit in any of the aforementioned subtopics, we propose to suborder it as (4) generator classification, as ECS are the generators of the Web pages of the shops.
A further dimension that characterizes Web page classification is the number of classes. As we aim to detect 6 different ECS, our research elaborates a multi-class problem. As only one ECS can be assigned to a given page, we handle a single-label classification. As classification, in our case, allows instances to be either in a class or not, we target hard classification. Finally, we operate on a flat classification problem, as the labels show no hierarchical order.
Qi and Davidson regard web directories, search engines, question answering systems and focused crawlers as important applications of Web page classification, and mention Web content filtering, assisted Web browsing and knowledge base construction as less important. Our work can be classified best to knowledge base construction, as our final aim is to generate additional structured data for e-commerce. In structured data research, data expressed according to ontologies is also commonly referred as knowledge base [8]. Finally, it can be argued that the work should rather be ordered into web *site* classification. We think the aforementioned criteria are suited nevertheless, as we execute the classification by operating on a single page, inducing our findings to the whole Web site. In that context, we assume that a Web shop is always generated by a *single* ECS.

*2) Web page classification in e-commerce:* Narrowed down to the e-commerce domain, there are two non-scientific sources that perform generator classification. Since 2011, (1) Robertshaw conducts an analysis of the market share of ECS semi-annually [9]. (2) Builtwith.com generates ECS statistics on a daily basis [10]. As both sell the resulting data, they don't disclose their methods of detection. Stoll et al. 2013 [6] provide a scientific overview about ECS market share on a product page level and their impact on the adoption of structured data on the Web.
Beside the classification of ECS, classification of different page types is a significant problem when trying to extract structured data in e-commerce. [6] identified (1) product pages, (2) category pages and (3) arbitrary pages as fundamental categories in the Web shop domain. To the best of our knowledge, there is no scientific work on automatically labelling shop pages to those categories.

### B. Supervised classification

Supervised machine learning operates on problems where a learning set with given labels is provided. The discrete label problem is called classification, whereas the continuous label problem is called regression [11]. The learning problem in supervised classification is characterized by $i$ feature vectors $\overrightarrow{v}_i$ that are labelled by $j$ classes $c_j$. The goal is to label novel vectors correctly, yielding high precision and recall [12].
Following Kotsiantis, figure 1 shows the general approach to supervised machine learning. (1) The problem is defined. (2)

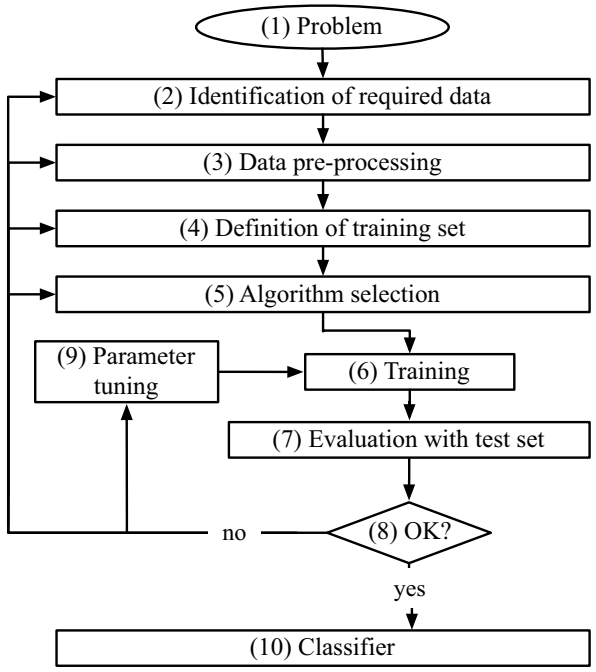



Fig. 1: Supervised Machine learning: General approach (following [11])

Then, the data needed for training the classificator is identified and obtained. (3) Preprocessing is performed, eliminating e.g. erroneous instances. (4) The data is split into learning set and test set to prevent overfitting. Overfitting means to predict the training set very well, but failing on unknown data. (5) A suitable algorithm is selected. (6) The algorithm is trained on the training data. (7) The performance is evaluated on the test set. (8) If the results are satisfactory, the classifier can be (10) implemented into a production system, if not, step 2 to 5 can be reviewed. (9) Additionally, the parameters of the classification algorithm can be tuned.

*1) Vectorization: Tf-idf term weighting:* As string features we exploit can't be used directly in classification algorithms, they have to be transformed into numerical vectors. A common approach to this task is counting the frequency of terms. But against our aim, this would emphasize terms that occur often. Contrary to that, we assume that a high discriminative power emerges from the terms occurring rarely. This problem has been addressed by the term frequency - inverse document frequency (tf-idf) approach by Jones 1972 [13], which we use to transform the string features to vectors. As word count in an instance increases, the tf-idf value increases proportionally, but is corrected by an offset reflecting the occurrence in the aggregated instances.

*2) Classification algorithms:* In the next subsection, we briefly introduce the selected classification algorithms. We also introduce common abbreviations to refer to the algorithms later.

1) **NC** The nearest centroid classifier is derived from nearest neighbour methods. The centroid reflects the

vector average of the class members [12]. Nitin and Bhatia 2010 provide a survey about k-nearest neighbours algorithms [14].

2) **SGD** Stochastic gradient descent improves the model by subsequently analysing instances. It gains superior performance with special optimization methods, that allow early convergence. [15]. Recently, it has drawn more attention by a publication of Zhang 2004, that emphasised its power for large scale learning [16].
3) **SVM** Support-Vector Machines have been introduced by Cortes and Vapnik 2005 [17]. SVMs operate on maximising the margin between a separating high-dimensional hyperplane and the given features.
4) **DTREE** Decision tree learning has originally been introduced by Quinlan 1986 [18]. It aims at inducing the trees by learning from examples. Decision trees are a commonly used method in knowledge discovery and decision support systems, as by asking a set of questions, they provide an straightforward approach to classify a pattern [19].
5) **RF** Random forests have been introduced by Breiman 1999 and belong to ensemble methods [20]. Ensemble methods combine predictions of multiple classifiers to raise performance. Instances are labelled by selecting the mode output of multiple decision trees that operate on a random subset of features. These are selected by choosing the most discriminative thresholds.
6) **XTREE** Extremely randomized trees [21] evolve the approach of random forests. They mainly differ in regard to a randomization of chosen attributes when splitting a tree node.

## III. Methodology, Approach & Implementation

### A. Design rationales

When using machine learning based classification systems, an important task for humans is feature generation. With the initial hypothesis of targeting "class" and "id" values, the curse of dimensionality of machine learning occurred. The curse of dimensionality refers to the problem of needing a very big learning set when a high-dimensional feature space is given, originally introduced into statistics by Hughes 1968 [22]. We first tried to reduce dimensions with manually crafted blacklists to exclude noisy terms, e.g. those that are related to document styling, and automatically filtering terms of very high and very low frequency. While this yielded little significant improvements, it showed that features filtered by a short domain-specific white list to be (a) sufficient to achieve significant classification performance, and (b) to be a good measure to reduce the curse of dimensionality.

### B. Generating datasets and preprocessing

*1) Classificator design dataset:* To generate the learning data, we used the sitemaps [23] downloaded by Stoll et al. 2013 [6]. We selected CS-Cart, Magento, Prestashop, Virtuemart, XT-Commerce and Zencart, as they provided a significant amount of training data, and were the most important ECS in terms of product pages, according to [6]. We provide an overview of the instances per ECS in the learning set in table 1. As a first step, of all available URIs, we eliminated those containing the string "blog", as they would dilute the data, and those containing ".png" / ".jpg", as images are out of the focus.

| System | Instances |
|---|---|
| CS-Cart | 460 |
| Magento | 234 |
| Prestashop | 2205 |
| Virtuemart | 431 |
| XT-Commerce | 684 |
| Zen-Cart | 458 |
| Sum | 4472 |

TABLE I: Learning set instances by ECS

To maximize the entropy of HTML pages, we randomly selected three URIs per site, downloaded them and rinsed the resulting directory excluding files smaller than 2KB, as by manual inspection, those were most often error pages. A learning set of 4472 HTML files remained.

*2) Additional evaluation dataset:* In addition to the usual splitting of the data into learning set and training set, we evaluated our results on entirely different data, which was extracted from GR-Notify [24]. GR-Notify is a Web service that allows Web shop owners to register their sites after they have implemented GoodRelations. The shop extensions for Magento and Virtuemart automatically submit data. Additionally, a front end to manually submit Web shops exists. By default, the generating ECS is included in the submission. Therefore, gr-notify provides a highly accurate external evaluation set for the trained classifier. From the gr-notify data, we could download 3461 HTML files labelled with Magento, Prestashop and Virtuemart. Other systems were not included, as they were not significantly represented in the GR-Notify dataset.

### C. Building a classifier

*1) Feature generation & white listing:* We assume that pages generated by distinct ECS show patterns in the values of HTML tag attributes "class" and "id". For instance, we spotted Magento generating often <H1 id="product-name">, or Prestashop <img class="pdct-img-main">.
Meanwhile, on the given data, this approach generates a feature set with a dimensionality in the lower $10^5$ range. As introduced in section 3.A, this results in limitations to the computability. There exist automated methods to reduce the dimensions in a learning problem, e.g. PCA, Pearson 1901 [25].
However, we chose a domain-specific white list approach to mitigate the problem. We reduced the feature set of attributes to only those which contain, but do not match, the strings "price", "product" and "cart". We generated the white list by starting with a manual collection of sensible terms, iteratively reducing those and reviewing the resulting classification performance. The final three terms showed to be of high discriminatory power. In table 2, we show the $recall_{\text{base}}$ related to the 4472 downloaded files after applying the white list to the feature sets "class", "id" and the combination "class+id", and the number of instances. The main point of not choosing PCA or related methods is occam's razor [26]. We achieve sufficient performance with this compact approach, instead of integrating another large scale method into our model.

*2) Classification algorithms:* We split the data into a 60% learning set and 40% test set. We trained the six classification

| | class | id | class+id |
|---|---|---|---|
| $recall_{\text{base}}$ | 0.849 | 0.811 | 0.875 |
| Instances | 3798 | 3628 | 3915 |

TABLE II: Remaining $recall_{\text{base}}$ after white list filtering

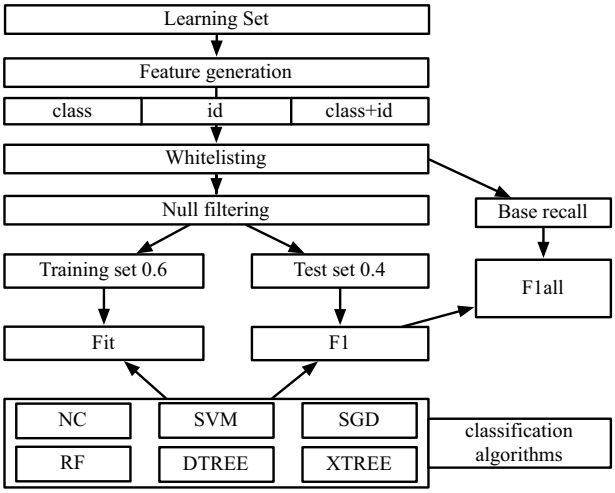



Fig. 2: Overview of experimental design

algorithms (NC, SGD, SVM, DTREE, RF, XTREE) discussed in section 2. Combining them with the three different feature sets resulted in an experiment size of 18 different combinations. It is important to state that for ECS detection, we analysed only *one* HTML page. We compile our experimental design in figure 2.

*3) Performance metric:* To asses the final performance of the feature set / classifier combination, we modified the common F1-score integrating the loss linked with feature generation.

$$F1_{\text{all}} = 2 * \frac{precision * (recall_{\text{base}} * recall_{\text{classifier}})}{precision + (recall_{\text{base}} * recall_{\text{classifier}})}$$

### D. Implementation

The experiment was implemented in the Python[7] programming language. The learning set was generated by a small script that drew sample URIs based on the sitemaps of Stoll et al. 2013 [6] and downloaded those asynchronously with the library grequests[8].
Vectorization, classificator application and evaluation has been realized with the library Scikit Learn 2011 [27]. The features were generated by applying regular expressions to the HTML files yielding lists of all values of the attribute in regard. We then excluded the instances that yielded no features after white list filtering. Before training and testing on the algorithms, to prevent overfitting, each dataset was split by 0.6 / 0.4 into train and test set.

[7]http://www.python.org/
[8]https://github.com/kennethreitz/grequests

| | XTREE | RF | SGD | DTREE | NC | SVM |
|---|---|---|---|---|---|---|
| class+id | 0.902 | 0.897 | 0.893 | 0.886 | 0.81 | 0.634 |
| class | 0.88 | 0.868 | 0.859 | 0.865 | 0.809 | 0.643 |
| id | 0.856 | 0.854 | 0.852 | 0.849 | 0.747 | 0.639 |

TABLE III: $F1_{\text{all}}$-score for 18 feature / algorithm combinations

The results have been visualized with matplotlib [28]. Aside from the learning set generator, which did not benefit from an interactive environment, we employed iPython notebook [29] to prototype in agile manner. Additionally, the pandas library has been used to manipulate matrix data [30].
We finally implemented shopdetector[9], a public Web interface to the classificator. It is built on the Web framework flask[10] and runs on a free-tier AWS EC2[11] instance. It provides 3 interfaces. (1), there is a REST [31] interface, (2) there is a form to enter the URI of the shop and (3) there is a bookmarklet which enable a straightforward usage of the service without leaving the browsing context. Putting the classificator on the Web was a tricky task, as Scikit Learn has a comprehensive list of dependencies that needed to be loaded into the Web server context. As a result, server deployment took much longer than implementing the app locally. After time-consuming trials on Apache HTTP Server[12], we successfully deployed on Nginx[13] [32]. For future work, we plan to automate server setup with the puppet[14] configuration management system. The web service is based on the Class-ID / XTREE classifier. We introduced a probability threshold of at least 0.6 to cut out predictions that were to ambiguous, providing the user of the service with a warning.
To set the later discussed speed results into perspective, a 2012 Mac Mini equipped with a quad-core 2.3 GHz Core I7 CPU, 8 GB of RAM and a SSD hard disk was used, scoring a 32-bit geekbench[15] of 10823.

## IV. Results

### A. Featureset & Algorithm performance

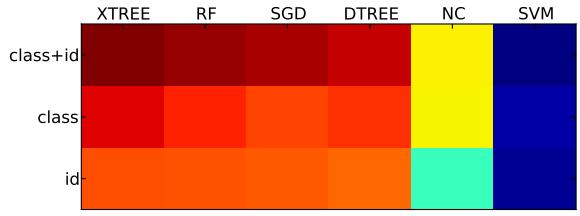
Table 3 shows the $F1_{\text{all}}$-score introduced in section 3.2 for the 18 feature / algorithm combinations. We additionally provide a heat map in figure 3.
We can see that the combination "class+id" performs best. That is mainly due to the highest base recall we showed in table 1, as the top algorithms did not yield significantly different performance. Applied to "class+id", XTREE, RF, SGD and DTREE perform similarly well around 0.9, while XTREE shows the best results. NC shows significantly worse results of 0.81, and SVM performs worst with 0.634. This result is in agreement with the current state of the art in classification research, where ensemble algorithms yield the best results.

### B. Speed

Additionally, we analysed the computational complexity of the feature / algorithm combinations. The results in seconds

[9]http://www.datacommerce.org/shopdetector
[10]http://flask.pocoo.org/
[11]http://aws.amazon.com/de/free/
[12]http://httpd.apache.org/
[13]http://wiki.nginx.org/Main
[14]https://puppetlabs.com/
[15]http://www.primatelabs.com/geekbench/

Fig. 3: Heatmap of $F1_{\text{all}}$-score for 18 feature / algorithm combinations




| | NC | SGD | DTREE | RF | XTREE | SVM |
|---|---|---|---|---|---|---|
| id | 0.119 | 0.131 | 0.596 | 1.461 | 1.893 | 13.093 |
| class | 0.155 | 0.16 | 0.859 | 2.173 | 3.069 | 18.499 |
| class+id | 0.23 | 0.252 | 1.158 | 3.086 | 3.928 | 30.687 |
| mean | 0.168 | 0.181 | 0.871 | 2.24 | 2.963 | 20.76 |

TABLE IV: Time elapsed (s) for 18 feature / algorithm combinations

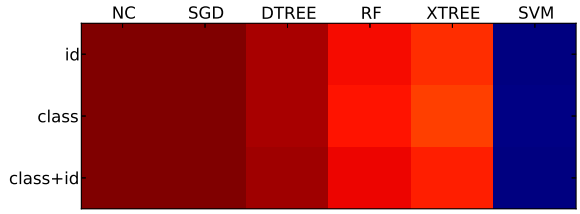



Fig. 4: Heatmap time elapses for 18 feature / algorithm combinations

are shown in table 4. We measured the time for fitting and generating the scores of a combination.
Regarding elapsed time in terms of feature sets, we see that the bigger ones took exponentially longer to compute. Regarding the algorithms, NC and SGD constitute a very fast group with means 0.168s and 0.181s. While DTREE is also relatively fast with 0.871s, RF and XTREE form a second, slower group with 2.24s and 2.963s. SVM is about two magnitudes slower than the fastest group. We additionally provide a heatmap for this analysis in figure 4. To adjust it to the same colors as above, we first normalized the elapsed times and then subtracted from 1. A theoretical discussion of the computional complexity of the underlying algorithms is out of the scope of this paper.

### C. Performance on different clusters

We additionally provide the classification report of the feature / algorithm combination with the highest performance, "class+id" / XTREE based on $recall_{\text{classifier}}$ in table 5. In terms of precision, CS-cart, Prestashop, xt-Commerce and Zen Cart can be detected with results of 1.00/0.99. Magento can be detected significantly worse with 0.86, and Virtuemart worst with 0.77. In recall, CS-Cart and Prestashop again produce a very good score of 0.99, while the other ECS move $\pm$ 0.01 around 0.93. This results in 4 groups of F1-scores: CS-Cart and Prestashop with 1.00 and 0.99, xt-Commerce and Zen Cart

| | precision | $recall_{\text{classifier}}$ | $F1_{\text{classifier}}$ |
|---|---|---|---|
| CS-cart | 1.00 | 0.99 | 1.00 |
| Magento | 0.86 | 0.94 | 0.90 |
| Prestashop | 0.99 | 0.99 | 0.99 |
| virtuemart | 0.77 | 0.92 | 0.84 |
| XT-commerce | 0.99 | 0.92 | 0.96 |
| Zen-Cart | 0.99 | 0.93 | 0.96 |
| avg / total | 0.97 | 0.97 | 0.97 |

TABLE V: Classification report of "class+id" / XTREE classifier on distinct ECS

| | NC | SGD | DTREE | RF | XTREE | SVM |
|---|---|---|---|---|---|---|
| Speed | +++ | ++ | + | o | o | - - - |
| Performance | - | ++ | ++ | ++ | +++ | - - |

TABLE VI: Consolidated review on speed / performance of used algorithms

with 0.96, Magento with 0.90 and Virtuemart with 0.84. An average F1-score of 0.97 could be measured.

### D. Consolidated algorithm review

We conclude the results with a consolidated review of strengths and weaknesses of the different classification algorithms in table 6, regarding performance and speed. NC is very fast, but does not provide competitive results. SGD and DTREE are fast and perform well. RF and XTREE provide a little more performance, but are significantly slower. SVM seems to be a bad choice for our problem.

## V. Evaluation

### A. Evaluation on gr-notify dataset

We evaluated the "class+id" / XTREE classifier discussed in section 4 on the gr-notify [24] dataset, consisting of Magento, Prestashop and Virtuemart instances. In table 7, we provide the base recall after applying the white list. We can see that the results are significantly worse than those in the learning set. This is because we only loaded the root URIs of the given sample, mostly excluding product pages, that the white list was tailored to. A classification report is provided in Table 8. Magento and Prestashop performed good with f1-scores of 0.94 and 0.97, while Virtuemart only yielded 0.79. An average F1-Score of 0.94 could be measured based on $recall_{\text{classifier}}$. The results show that the classifier performs nearly similarly well on a total independent dataset, even when analysing the root URI only. Based on the proposed formula in section 3.3, the classifier yields a $recall_{\text{all}}$-score of 0.73.

### B. Evaluation on targeted ECS reference shops

We provide an evaluation of the classification performance on ECS reference shops the system has been trained for. We

| | class | id | class+id |
|---|---|---|---|
| $recall_{\text{base}}$ | 0.63 | 0.55 | 0.64 |
| Number of instances | 2831 | 2461 | 2864 |

TABLE VII: GR-Notify [24] evaluation: Remaining recall after white list application

| | precision | $recall_{\text{classifier}}$ | $F1_{\text{classifier}}$ |
|---|---|---|---|
| Magento | 0.95 | 0.92 | 0.94 |
| Presta | 0.98 | 0.95 | 0.97 |
| Virtuemart | 0.85 | 0.74 | 0.79 |
| avg / total | 0.96 | 0.93 | 0.94 |

TABLE VIII: GR-Notify [24] evaluation: Classification report of "class+id" / XTREE classifier

discuss sources of reference shops and results in the following section. As described in section 3.D, we classified results that yielded a probability below 0.6 not to belong to the targeted ECS.

1) **CS-cart** Acquiring the URIs of 10 reference CS-Cart stores was possible by checking the portfolios of specialized agencies. All 10 CS-Cart shops have could be classified correctly.
2) **Magento** Magentoshopping.de[16] is a german portal that listed 256 Magento shops as of 04/23/2013. We picked the 10 shops that were listed to be the newest for the evaluation. 8 shops could be detected correctly. One shop had been erroneously submitted to the portal, as manual checking showed it was generated by ubercart[17].
3) **Prestashop** To acquire the Prestashop reference sites we consulted the prestashop.com showcases[18]. Again, we extracted the 10 URIs that were listed most recently and checked them with the classificator API. This resulted in 7 correctly detected URIs and 3 URIs that did not meet the treshold.
4) **Virtuemart** Virtuemart provides a collection of live stores[19]. We picked again the 10 shops listed most recently. Of those, four have been labeled correctly above the treshold. All others were below. We explain the poor performance in this part of the evaluation as a result of the weak technical impression Virtuemart showed, and with the non-curated character of the data source. As it is principally possible to add arbitrary sites, we expect it to be polluted by sites submitted only for SEO benefits.
5) **XT-Commerce** To assess XT-Commerce, we used the URIs referred by the official XT-Commerce website[20]. Of 10 extracted URIs, three could be detected correctly as XT-Commerce. 5 URIs fell below the threshold, the remaining two were classified wrong. We interpret the poor results to be rooted in a severe overfitting in this specific ECS, and by the strong template customization of the showcase XT-Commerce shops.
6) **Zen-Cart** For Zen-Cart we consulted the apparel subcategory of the official showcase site[21]. The results were the worst in this part of the evaluation. One URI could not be resolved at all, and the 9 remaining URIs did not pass the threshold. We think this is related to the very dubious impression most Zen-Cart shops left, as many of those sold imitations of brand products. The poor technical realization might affect the patterns the templates spot. Additionally the low number of learning instances must have resulted in overfitting.

[16]http://www.magentoshopping.de
[17]http://www.ubercart.org/
[18]http://www.prestashop.com/de/showcase
[19]http://virtuemart.net/features/live-stores/16
[20]http://www.xt-commerce.com/
[21]http://www.zen-cart.com/showcase.php?do=showcat&catid=1

| | True pos. | False pos. | False neg. | Errors |
|---|---|---|---|---|
| cs-cart | 1.000 | 0.00 | 0.00 | 0.000 |
| mage | 0.800 | 0.00 | 0.10 | 0.100 |
| presta | 0.700 | 0.00 | 0.30 | 0.000 |
| virtuemart | 0.400 | 0.00 | 0.60 | 0.000 |
| xt-commerce | 0.300 | 0.20 | 0.50 | 0.000 |
| zen-cart | 0.300 | 0.10 | 0.50 | 0.100 |
| mean | 0.583 | 0.05 | 0.35 | 0.017 |

TABLE IX: Evaluation on targeted ECS reference shops - classification results

| | Precision | Recall | F1-score |
|---|---|---|---|
| cs-cart | 1.000 | 1.000 | 1.000 |
| mage | 1.000 | 0.800 | 0.889 |
| presta | 1.000 | 0.700 | 0.824 |
| virtuemart | 1.000 | 0.400 | 0.571 |
| xt-commerce | 0.600 | 0.375 | 0.462 |
| zen-cart | 0.750 | 0.375 | 0.500 |
| mean | 0.892 | 0.608 | 0.708 |

TABLE X: Evaluation on targeted ECS reference shops - precision, recall, F1-score

We see the the mixed result of this part caused by two factors. First, shops that miss technical sophistication generally suffer from wrong classification. At the same time, we expect the labelled URIs of those ECS to be way less accurate, as listings often are crowd-curated. Second, result differences on test set and on this data hint that our approach tends to overfit on specific clusters. We discuss this potential shortcoming in the limitations section 7. We provide an overview of the results of this section in table 9 and 10. Table 9 provides the relative instance frequency in each result group. Table 10 provides the achieved precision, recall and F1-scores. Overall, a precision of 0.892, a recall of 0.608 and F1-score of 0.708 could be measured.

### C. Evaluation on non-targeted ECS reference shops

We additionally conducted a performance analysis for ECS reference sites the classificator was not trained for on the systems 3DCart, Oxid esales, and Volusion. In this experiment, by definition there exist only true negatives, false positives and errors (if the page could not be fetched e.g.). We can't compute precision and recall based on those figures, but the true negative rate. The true negative rate for 3DCart is 0.6, for Oxid esales 0.875, and 1.0 for Volusion. Again, we interpret the result aligned to the low-end impression 3Dcart conveyed on the system's home page and in the associated shops. We compile our results in table 11.

### D. Evaluation on non-shop sites

We additionally evaluated the predictive performance of the system on non-shop sites. As test sample we used 20 randomly selected sites from the Alexa Top 1M URIs list and manually

| | True neg. | False pos. | Errors | True neg. rate |
|---|---|---|---|---|
| 3DCart | 0.600 | 0.400 | 0.000 | 0.600 |
| Oxid esales | 0.700 | 0.100 | 0.200 | 0.875 |
| Volusion | 1.000 | 0.000 | 0.000 | 1.000 |
| mean | 0.767 | 0.167 | 0.067 | 0.825 |

TABLE XI: Evaluation on non-targeted ECS reference shops

filtered shops. We then submitted the URIs to the Class+ID / XTREE classificator. Beside one 404, 19 URIs did not meet the threshold, and were labeled correctly as true negatives. This result confirms our approach also works well for non-shop-sites.

## VI. Limitations & outlook

*1) Limited features:* We did not exploit further features that might raise the performance, for instance HTTP header [33] or tag frequency in the HTML document. In that context, we propose that computing graph properties of the HTML trees might be a promising way to design classificators. We don't think that our focus on the limited features is a problem, as the classificator matches our performance needs for the outlined extractor. From our point of view, it is already a contribution to gain an additional low percentage of market coverage.

*2) Biased learning set:* A severe bias might have been introduced by choosing a learning set that has been labelled automatically. For future work, we aim at collecting a curated dataset of Web shop URIs labeled by ECS. We assume that this would raise the performance of the classificator significantly. A fundamental principle in machine learning is that having more data is of higher importance to the performance that the used algorithm [34]. We think that this is not critical, as we yield sufficient results for the projected use. Additionally, one might argue that a bias is introduced by the learning not evenly distributed across the different ECS, see Table 1. At the same time, taking into account all labelled pages available yielded the best overall classification performance.

*3) Parameter tuning:* Additional classificator performance could have been generated by tuning the parameters of the algorithms. We decided not to perform parameter tuning, as a preliminary tuning of the best classifier yielded only little better F1-scores. These get further diminished by the $recall_{\text{base}}$ and thus didn't seem critical for this prototype. Meanwhile, in a production system, parameter tuning should be performed as soon as the final algorithm is chosen, e.g. based on our heuristic provided in section 4.4. We assume especially that the poor SVM results are due to missing parameter tuning.

*4) White list generation:* It is possible that the heuristically generated white list may not be the optimal one. Formally approaching this problem could be future work. Compact approaches seem to be promising in comparision to full scale methods like PCA.

*5) Detecting ECS by only analysing one HTML page:* The performance of the classificator could be further risen by considering multiple HTML pages per web site. Again, for the projected use case, the yield rate is satisfactory.

*6) Overfitting:* Section 5.B showed that the system tends to overfit on some ECS. We think this problem could be addressed by additional learning data. The procurement of high-quality labelled ECS is a non-trivial task and could serve as future work.

*7) Threshold:* We heuristically set the threshold to invalidate predictions to 0.6. Future work based on the evaluation could provide formal method to set the figure.

## VII. Conclusion

We designed an approach of ECS detection based on supervised classification and a filtered set of HTML attribute values. It is capable of detecting 6 different e-commerce systems by analysing only one random HTML page of a Web shop. Taken into account the loss in recall when not able to generate any features, it shows an F1-score of 0.9. An extensive evaluation fundamentally confirmed the results. We provided an analysis of the speed of the different algorithms, the performance on specific ECS, and a heuristic to choose a classification algorithm for the task at hand. We additionally implemented a Web frontend and API for the public use of the system.

## References


[1] M. Hepp, "GoodRelations: An ontology for describing products and services offers on the web," in *Knowledge Engineering: Practice and Patterns* (A. Gangemi and J. Euzenat, eds.), vol. 5268 of *Lecture Notes in Computer Science*, pp. 329–346, Springer Berlin Heidelberg, 2008.

[2] "Home - schema.org." http://schema.org/. [Online; accessed 01/10/2013].

[3] "GoodRelations Business Value — Strategies for Building Semantic Web Applications." http://notes.3kbo.com/goodrelations-business-value. [Online; accessed 03/20/2013].

[4] "Shop extensions - GoodRelations Wiki." http://wiki.goodrelations-vocabulary.org/Shop_extensions. [Online; accessed 03/20/2013].

[5] C.-H. Chang, M. Kayed, M. R. Girgis, and K. F. Shaalan, "A survey of web information extraction systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 10, pp. 1411–1428, 2006.

[6] U. Stoll, M. Ge, and M. Hepp, "Understanding the impact of e-commerce software on the adoption of structured data on the web," in *BIS* (W. Abramowicz and R. Tolksdorf, eds.), vol. 47 of *Lecture Notes in Business Information Processing*, Springer, 2013.

[7] X. Qi and B. D. Davison, "Web page classification: Features and algorithms," *ACM Comput. Surv.*, vol. 41, no. 2, pp. 1–31, 2009.

[8] M. Hepp, "Ontologies: State of the art, business potential, and grand challenges.," in *Ontology Management* (M. Hepp, P. D. Leenheer, A. de Moor, and Y. Sure, eds.), vol. 7 of *Semantic Web And Beyond Computing for Human Experience*, pp. 3–22, Springer, 2008.

[9] "October 2012 eCommerce Survey — Tom Robertshaw." http://tomrobertshaw.net/2012/11/october-2012-ecommerce-survey/. [Online; accessed 01/10/2013].

[10] "Ecommerce technology web usage statistics." http://trends.builtwith.com/shop. [Online; accessed 03/20/2013].

[11] S. B. Kotsiantis, "Supervised machine learning: A review of classification techniques.," in *Informatica 31:249–268*, 2007.

[12] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[13] K. S. Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of Documentation*, vol. 28, pp. 11–21, 1972.

[14] N. Bhatia and Vandana, "Survey of nearest neighbor techniques," *CoRR*, vol. abs/1007.0085, 2010.

[15] S. Owen, R. Anil, T. Dunning, and E. Friedman, *Mahout in Action*. Manning Publications Co., first ed., 2011.

[16] C. E. Brodley, ed., *Machine Learning, Proceedings of the Twenty-first International Conference (ICML 2004), Banff, Alberta, Canada, July 4-8, 2004*, vol. 69 of *ACM International Conference Proceeding Series*, ACM, 2004.

[17] C. Cortes and V. Vapnik, "Support-vector networks," in *Machine Learning*, pp. 273–297, 1995.

[18] J. R. Quinlan, "Induction of decision trees," *Mach. Learn*, pp. 81–106, 1986.

[19] B. Sidaoui and K. Sadouni, "Efficient binary tree multiclass svm using genetic algorithms for vowels recognition," in *Proceedings of the 10th WSEAS international conference on Computational Intelligence, Man-Machine Systems and Cybernetics, and proceedings of the 10th WSEAS international conference on Information Security and Privacy*, CIMMACS'11/ISP'11, (Stevens Point, Wisconsin, USA), pp. 228–234, World Scientific and Engineering Academy and Society (WSEAS), 2011.

[20] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[21] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine Learning*, vol. 63, no. 1, pp. 3–42, 2006.

[22] G. Hughes, "On the mean accuracy of statistical pattern recognizers," *Information Theory, IEEE Transactions on*, vol. 14, no. 1, pp. 55–63, Jan.

[23] "sitemaps.org - protocol." http://www.sitemaps.org/protocol.html. [Online; accessed 03/20/2013].

[24] "GR-Notify." http://gr-notify.appspot.com. [Online; accessed 03/20/2013].

[25] K. Pearson, "On lines and planes of closest fit to systems of points in space," *Philosophical Magazine*, vol. 2, no. 6, pp. 559–572, 1901.

[26] J. Malpas, *The Stanford Encyclopedia of Philosophy*. The Metaphysics Research Lab, Stanford University, 2012.

[27] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[28] J. D. Hunter, "Matplotlib: A 2D graphics environment," *Computing In Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[29] F. Pérez and B. E. Granger, "IPython: A System for Interactive Scientific Computing," *Computing in Science and Engineering*, vol. 9, no. 3, pp. 21–29, 2007.

[30] W. Mckinney, "pandas: a foundational python library for data analysis and statistics," in *SC '11: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 273–297, ACM, 2011.

[31] R. T. Fielding, *REST: Architectural Styles and the Design of Network-based Software Architectures*. Doctoral dissertation, University of California, Irvine, 2000.

[32] W. Reese, "Nginx: the high-performance web server and reverse proxy," *Linux J.*, vol. 2008, Sept. 2008.

[33] "Hypertext Transfer Protocol – HTTP/1.1." http://www.ietf.org/rfc/rfc2616.txt. [Online; 03/20/2013].

[34] A. Halevy, P. Norvig, and F. Pereira, "The unreasonable effectiveness of data," *Intelligent Systems, IEEE*, vol. 24, no. 2, pp. 8–12, 2009.