U.P.B. Sci. Bull., Series C, Vol. 79, Iss. 1, 2017 ISSN 2286-3540

EXTRACTION OF ATTRIBUTES AND VALUES FROM ONLINE TEXTS

Alexandra GHECENCO[1], Traian REBEDEA[2], Costin CHIRU[3]

Web documents contain vast amounts of information that can be extracted and processed to enhance the understanding of online data. Often, the structure of the document can be exploited in order to identify useful information within it. Pairs of attributes and their corresponding values are one such example of information frequently found in many online retail websites. These concentrated bits of information are often enclosed in specific tags of the web document, or highlighted with certain markers which can be automatically discovered and identified. This way, different methods can be employed to extract new pairs from other, more or less similar, documents. The method presented in this paper relies on the DOM (Document Object Model) structure and the text within web pages in order to extract patterns consisting of tags and pieces of text and then to classify them. Several classifiers have been compared and the best results have been obtained with a C4.5 decision tree classifier.

Keywords: information retrieval; attribute-value pairs; information extraction; text mining; classification

1. Introduction

This paper presents a system for extracting tokens that represent attributes and their corresponding values from web pages. As part of the combined effort to build a semantic web, this would improve the automatic understanding of web pages by computers, giving additional meaning to texts. Furthermore, this would significantly improve the efficiency and intuitiveness of web searches, by correlating words and phrases to the role humans attribute to them in natural conversations.

The web contains heterogeneous texts, most of which are structured in HTML (or web) pages. Given their inherent tag (or DOM – Document Object Model) hierarchy and division into specific components, web pages can be easily split into entities that form a tree and that have attributes of their own. The basic idea is that, based on the correlations found between these attributes and the role

[1] Engineer, Department of Computer Science, University POLITEHNICA of Bucharest, Romania, e-mail: alexandra.ghecenco@gmail.com

[2] Lecturer, Department of Computer Science, University POLITEHNICA of Bucharest, Romania, e-mail: traian.rebedea@cs.pub.ro

[3] Lecturer, Department of Computer Science, University POLITEHNICA of Bucharest, Romania, e-mail: costin.chiru@cs.pub.ro

each analyzed entity plays in a semantic view, a classifier can be trained on a manually annotated corpus with attribute-value pairs so that any web page may be later processed and useful information derived from it by identifying the attribute-value pairs with the trained classifier.

The rest of the paper is structured as follows. In the next section, we present the most important previous approaches to the problem of attribute and value extraction. Section 3 is dedicated to an overview of the proposed system, while each component is described in detail in section 4. Then, section 5 describes the experiments performed to validate our approach, together with the results obtained by the system. Finally, the last section highlights the main contributions of this paper and proposes several future improvements.

2. Related Work

Exploiting the Document Object Model (DOM) tree structure is a popular idea in online information extraction, and the more similar a web page will be to the ones in the training corpus, the better the accuracy – as pointed out by Ravi and Pasca [1]. They developed a system that analyzes the location of the relevant nodes (i.e. attributes and values) within the DOM tree, keeping track of it using pattern vectors. The training phase consists in building these pattern vectors, while the classification phase applies them to other pages, extracting the attributes and values, together with a certain correlated probability. This approach is very intuitive to humans and similar to the way we interpret web pages too – note how on websites such as Wikipedia or retailer sites, punctual information, in the form of attribute-value pairs, is usually in the same place relatively to the layout. In computer science terms, this translates into pattern vectors, probabilities, parameterization and machine learning techniques. The method presented in this paper also takes into account the placement of important nodes within a webpage, using their XPath and child / sibling information as training attributes in the learning process. Moreover, a heuristic described by [1] is also of critical importance in our proposed method: the discrimination between attribute nodes and irrelevant nodes by the corresponding tag. Ravi and Pasca [1] suggest that nodes representing an attribute are frequently enclosed in HTML tags that emphasize the text within, such as <b>, <i>, <u>, etc. This helps building a set of tags that are relevant to the learning algorithm and another set of tags that denotes nodes that can be completely discarded – for instance, <img>, <script>, <meta>, etc.

Ravi and Pasca [1] present a comparative analysis of the results obtained with three variants of the system. The first represents the baseline system with structured text, ranking the attributes by frequency. The second one (S_{hier}) uses hierarchical patterns to rank the attributes instead, while the third one (D_{patt}) uses a previous, classic approach based on patterns such as X-of-Y to extract attributes from unstructured text. Pruning by POS tags returned by WordNet [2] enhances each of these subsystems, trimming out irrelevant parts of speech. The data source consisted in web search results of querying class instances, filtering HTML documents out of the first N results of a search.

While the system presented in this paper also relies on the DOM structure, it does not use pattern vectors, but instead it builds training instances that are classified by several algorithms. The best score for precision obtained by [1] was 0.87 with the S_{hier} implementation at rank 5.

Moreover, as far as structuring information goes, tables are the most effective way of synthesizing attribute-value relations, as observed by Wang et al. [3]. They built a search engine that processes tables on the web and builds the result set as a table as well, given a training set of tabular data for learning the semantic connections between attributes (found in the header) and values (found in the body of the table). Therefore, the previously mentioned set of relevant nodes can be extended to contain tags such as <th> and <tr>. Their search engine processed two corpora of tables – the ones in Wikipedia, counting up to 0.65 million, and the rest of the World Wide Web, around 0.3 billion pages – that unified the retrieved information into a table-formatted answer.

In order to connect the data in a table with their meaning, the authors used Probase [4], a probabilistic taxonomy of facts significantly larger than WordNet (WordNet – 25,229 concepts, Probase – 2,653,872 concepts) containing sets of ranked entities and instances for each concept. The first step in the algorithm they designed was to find tables relevant to the query. Then, each table is scanned for the header in order to assign a label/attribute to the data, and then one is generated using Probase if the header cannot be found. The column containing entities is identified, such that the remaining columns contain attributes and associated values. If the header or entity column cannot be derived from the table, it is discarded; otherwise, the information it contains is structured into relationships and added to Probase for further use.

The data corpus used by this framework consists in approximately 65.5 million raw HTML tables extracted using a simple rule-based system that filters out irrelevant tables (tiny, very large, calendars etc.) from webpages, paying special attention to Wikipedia due to its high degree of relevance. Although not all extracted tables are relational, Probase processing eliminates them at a later point. Random sampling has led to a percentage of 79.5% relational tables. Interrogating the search engine with queries containing concepts, entities, attributes and keywords leads to the discovery of new attributes and values for them.

Wang et al. [3] obtained results of over 0.80 for precision@1. These numbers surpass the scores achieved by the system presented in this paper. It is noteworthy, however, that their framework is a search engine specialized for tables and cannot deal with standard HTML pages.

Another concept that has previously attracted the attention of researchers is the fact that product description pages' structure data in a way that makes it easy to distinguish between attributes, values, and other text. Probst et al. [5] have developed a semi-supervised system for the extraction of attribute-value pairs from product descriptions. They used textual descriptions taken from retailer sites using simple web crawlers. Their system is composed of specialized modules for data collection, automatic seed generation, entity extraction using 2 algorithms (Naïve Bayes and co-EM), relationship extraction using MiniPar [6] and user aided error correction.

The data collection and preprocessing are done on unlabeled training data. Only relevant phrases are extracted from the text, eliminating irrelevant parts of speech. This is done using a POS tagger. The remaining words are stemmed, and numbers and measures are replaced with special tokens. After these steps, the method proceeds to seed generation, providing the input data for the classifiers. Initially, all seeds are labeled either as attributes or as values, and the remaining words as neither. After the learning algorithm of choice is done processing, each word will have a label with an associated probability of being an attribute or a value. After the labeling, the relationships between attributes and values remain to be established. This is done using heuristics. First, the correlation values are computed for the phrases that are to be analyzed and the phrases with high scores are merged. Then, known seed pairs are identified and linked together. Linking is also done on phrases that have a high syntactic correlation score and that have relevant co-location information.

The best precision obtained by Probst et al. [5] was 0.96, considering attributes and values that were not necessarily related. As the system proposed in this paper does not discover attribute-value relationships either, it can be said that the precision is similar, with a slightly higher score of 0.97. The methods, however, differ. The ones presented in this paper rely on supervised search instead of semi-supervised and do not compute correlations between attributes and values, but extract them separately.

Bellare et al. [7] employ different learning algorithms in their attribute/value learning system. One of them is DL-coTrain, an adaptation of the classic co-training algorithm [8] to learn decision lists given examples from only one class. A decision list is a function that maps a feature and a label to a confidence value. The algorithm induces decision lists from multiple classes starting from seeds provided by the user. The authors also propose an alternative algorithm, MaxEnt, based on maximum-entropy self-training. This classifier works on a dataset divided in positive and negative instances (initially, the seeds being only the positive examples).

The system works in two steps. The first step is one of preprocessing. After annotating the text with a POS tagger, pairs in the form of (noun, noun) are extracted from sentences. These represent entity-attribute pairs that are used as seeds for the training of the classifiers. After these operations, one of the learning algorithms is run, resulting in a larger number of attribute-value pairs. These results are re-ranked using a function of the confidence value and co-training scores specific for each attribute and value. The final results are the top ranking ones.

The data used in the experiments consisted in a newswire corpus of 122 million tokens with articles collected from Wall Street Journal, AFP and Xinhua News. The attributes and corresponding values extracted belonged to the Country and Company classes.

Several flavors of the method were tested and compared, starting with a baseline system, a reconstruction of Espresso [9], then enhancing with CoTraining or MaxEnt extraction and additional context and features. The best performing algorithm, SE+R, consisted of a MaxEnt model with additional context and lexical features and with re-ranking both on the extracted data and on the initial seeds. It had the best results both for attribute and for tuple extraction, with an overall average precision of 0.80. While this is below the scores achieved by the system proposed in this paper, the method developed by Bellare et al. [7] differs from it by working on unstructured text.

Another framework for attribute and value extraction took into account both the page-independent content information and the page-dependent layout. Wong et al. [10] designed a probabilistic graphical model [11] for the relationship between layout and content which extracts and normalizes product attributes from web pages in an unsupervised manner. Their approach can be seen as a combination between the solutions previously discussed. They have developed a complex mathematical formulation describing their model, but the core idea behind it is as follows: the observable information regarding a text fragment depends both on the content, C, and the layout, L. The unobservable information is whether the text fragment is an attribute, which can be modeled as a binary variable called target (T), and its attribute information (value) A. The problem of extracting attributes is therefore identical to finding for each text fragment $T = t^*$ such that $t^* = \text{argmax}\{P(T = t \mid C, L)\}$, while attribute normalization is identical to finding for each text fragment $A = a^*$ such that $a^* = \text{argmax}\{P(A = a \mid C, L)\}$. These two probabilities are dependent, therefore attribute extraction and normalization must be done simultaneously in order to ensure consistency of the variables. The probabilities can be merged into $P(T, A \mid C, L)$, thus reducing the whole problem to one of maximizing the value of this expression.

The system was tested on 85 pages from 41 sites containing information about digital cameras, 96 pages from 62 sites regarding MP3 players, 111 pages from 61 sites about camcorders and 29 pages extracted from a restaurant guide. Each page contained one product with a variable number of attributes. Text fragments were extracted considering certain HTML tags as delimiters, recording the HTML layout of the page during the process. 10 runs of experiments were conducted for each product class. The results obtained by the framework were quite satisfactory, with an average precision of 0.76, recall of 0.79 and F1 measure of 0.78. The main difference between method presented by Wong et al. [10] and the one described in this paper is the mathematical aspect of it. Instead of a learning algorithm, as is used in the system described in the following section, Wong et al. [10] view attribute and value extraction as a function.

More recently, the system developed by Han et al. [12] proposes the use of spatial relations based on the rendering of the web page, similarly to its display in a browser, in order to extract various types of tuples from web pages. Their results are encouraging, but they have not tested their work on pairs of attributes and values.

3. Overview of the System

In order to extract attributes and values, two subsystems work together: the former is responsible with training a classifier on a manually annotated attribute-value pairs extracted from a set of web pages, while the latter performs the actual extraction of new attributes and values during the classification phase.

3.1. Training Phase

The training phase takes place in two steps. Firstly, training data is generated manually by labeling web pages returned by a Google search. The search query should be either a class name (such as ‘digital camera’ or ‘laptop’) or an instance name (‘Nikon Coolpix L280’ or ‘Toshiba Satellite L350’). For better results, the search engine can be customized to prioritize certain sites in the result set; as mentioned in the introduction, retailer sites are among the best structured. The labeling requires the web page to be downloaded locally. For the purpose of this project, the labeling system has been designed to work as simple as possible and in any web browser. The user highlights HTML elements considered to be attributes and values, with different colors. This color code is saved in the HTML body as an attribute of each element. This way, the color code of each element will determine its class – attribute, value or none.

In the second phase, the elements of the DOM tree are used as seeds for the learning algorithm. Since many elements of the web page are not visible to the end user and, therefore, completely irrelevant, they are filtered out. This is the case of <meta>, <head>, <script> and other similar tags. Image and media elements are also eliminated from the training set because they do not contain any text. On this training set, a learning algorithm is then run to obtain a classifier: C4.5 for a decision tree [13], Naïve Bayes for a simple frequentionist approach [14], K* for a cluster set analysis [15], Logistic regression for a probabilistic model [16], and JRip for a rule set algorithm [17].

3.2. Classification Phase

The classifier obtained in the training phase is then used to classify each element in the set of pages that need to be processed. According to the decision rules of each classifier, these elements (once again, counting out the irrelevant ones) are sorted into one of the 3 categories (attribute, value, or none). In the end, the results will be a set of attribute nodes and a set of value nodes. These are highlighted using the same color code as in the labeling phase and can be viewed as part of the web page after saving and deploying it on a web server.

4. System Components

4.1. Platform and Technologies

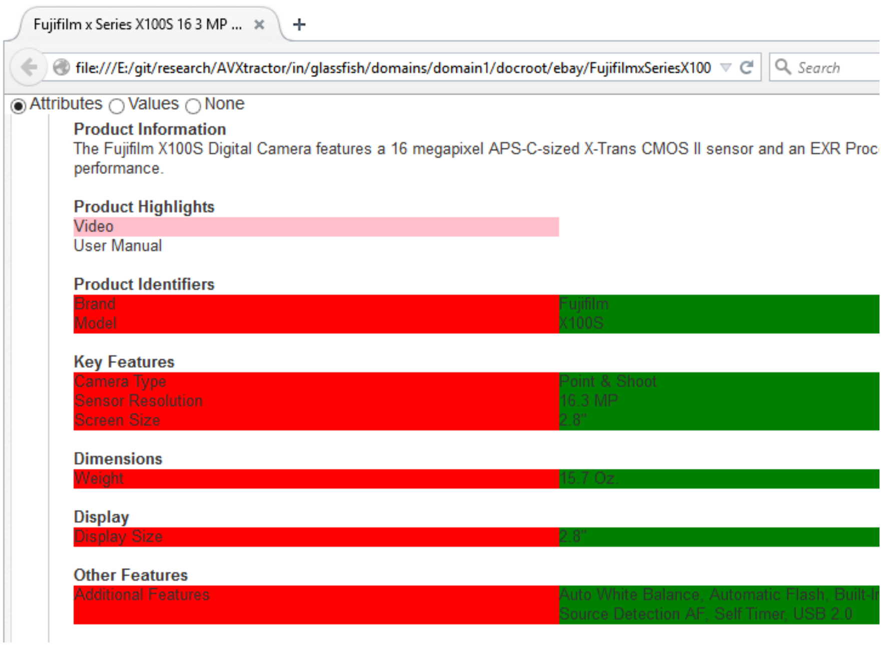
The system core was implemented in Java, while the labeling component was developed in JavaScript. The web search module uses Google Custom Search API [18]. The search engine generating the requests can be customized to focus the search on certain websites. This way, more relevant results can be retrieved by setting the preference for retail sites (Amazon, Ebay, Adorama, etc.).

The HTML parsing is performed using the jsoup library [19]. It creates a tree-structured object corresponding to the DOM tree of the document, keeping all the attributes of the HTML elements. Its traversal is a key step for building the values of several training attributes, such as the node’s XPath, number of child nodes, and number of siblings.

Another training attribute used in the learning process is the part of speech of the text enclosed in the node, or the predominant part of speech in case of multiple words. This information is extracted using the Apache OpenNLP framework [20].

4.2. Labeling System

Hand-picked web pages and search results as well can be used to generate input data for the training phase. This is known as the labeling phase, in which the user selects elements from the HTML tree and marks them as attributes or values. To accomplish this, a JavaScript program must be inserted in the web page. For this purpose, the web page should be downloaded locally, edited and then viewed in a web browser. This script adds an action to mouse gestures, as well as a selector for determining whether the user is currently marking attributes or values. Clicking an element will change the background color to red in case of attribute selection, or green in case of values as shown in Fig. 1. The updated HTML page is saved and deployed on a web server.


Fujifilm x Series X100S 16 3 MP ...

file:///E:/git/research/AVXtractor/in/glassfish/domains/domain1/docroot/ebay/FujifilmxSeriesX100

Search

[✓ Attributes] Values None

Product Information
The Fujifilm X100S Digital Camera features a 16 megapixel APS-C-sized X-Trans CMOS II sensor and an EXR Proc
performance.

Product Highlights
Video
User Manual

Product Identifiers

Brand	Fujifilm
Model	X100S

Key Features

Camera Type	Point & Shoot
Sensor Resolution	16.3 MP
Screen Size	2.8"

Dimensions

Weight	15.7 Oz.

Display

Display Size	2.8"

Other Features

Additional Features	Auto White Balance, Automatic Flash, Built-Ir Source Detection AF, Self Timer, USB 2.0

Fig. 1. Sample training page in the labeling phase DOM Parser

4.3. DOM Parser

Using the jsoup library, web pages are parsed into a tree with the same structure as the document's object model (DOM). Jsoup builds a Document object, having an Element object as root of the tree and a hierarchy of Element objects as children. The structure of the Element class allows access to the HTML properties of the elements, along with CSS style attributes and dynamic properties enclosed in scripts. The tree can be easily traversed, calling user defined functions at each visited node. The used tree traversal is depth-first, which allows easy tracking of the structural information per node. Each element object found within the tree traversal was enriched with several properties: the enclosed text, the XPath expression, HTML entropy for the element itself and for all its child nodes. These will be used in the training phase as training attributes as will be described in the next subsection.

4.4. Training the Classifiers

Weka [21] is an open source collection of algorithms for data mining and machine learning, including various clustering and classification methods, and helper tools for pre-processing data, feature selection and visualization.

For the purpose of this project, several of Weka's classifiers have been deployed to solve the attribute-value extraction problem on the gathered dataset. Both training and classification have been run with each of five chosen classifiers described next, each classifier representing a different paradigm for supervised learning. The training and test datasets have been the same for all the classifiers that were investigated.

Since training records correspond to elements in the DOM tree - but not a one-to-one correspondence because not all elements are relevant, as discussed in the previous section, the possible classes are attribute, value or none. Next, we shall present the features that are used for each element to distinguish between the classes and the classifiers that have been taken into consideration for evaluation.

4.4.1. Attributes

Text Entropy

Text entropy is a continuous attribute with real values which depends on the text contained in the element and represents the quantity of information enclosed in that specific text. Text entropy is defined as follows:

$$H(S) = -\sum p_i \log_2 p_i \tag{1}$$

where p_i is the frequency of the ith text token.

HTML Entropy

The HTML entropy is similar to text entropy, with the difference that instead of analyzing text tokens in a text snippet, in this case the HTML tags in the XPath expression of the current element are considered. Its formula is similar to the one above, but the frequency is computed for each element in the analyzed XPath expression rather than for each text token. This attribute represents the quantity of information that can be extracted from the XPath.

XPath Length

The length of the XPath selector is an attribute with integer values that underlines the relevance of how "deep" in the DOM tree the element under scrutiny is positioned.

HTML Tag

The HTML tag is a discrete attribute. Only a subset of all the existing HTML tags is considered relevant for the studied problem. These tags are: a, b, i, u, li, td, th, strong and span.

Number of Child Nodes

The number of child nodes is another attribute with integer values. Its relevance can be seen on web pages where it is clear that value-type nodes are often leaves in the DOM tree, such as cell nodes in a table.

Number of Sibling Nodes

Similar to the previous attribute, this one can also be empirically observed in web pages with product descriptions, for instance, where a HTML list of value-type nodes contains a large number of siblings with the <li> tag.

Part of Speech

The part of speech is a discrete attribute with 8 possible values considered for the purpose of this project: noun, number, pronoun, adjective, adverb, verb, preposition/conjunction and other. In case of text snippets with more than one word, the most frequent part of speech in the snippet is selected to represent the whole text. Depending on the domain to which the search results pertain, several relations between the part of speech and the node type are easily observed – for instance, pages describing digital cameras contain many numeric value nodes (e.g. focal length, size, weight, price), while pages belonging to furniture shops contain value nodes that have more adjectives or nouns.

4.4.2. Classifiers

C4.5 [13] is one of the most popular algorithms used for statistical classification. It builds a decision tree in the training phase given a set of labeled data, which can be later used as a classifier on sets of unlabeled data that have the same attributes..

Naïve Bayes classifiers [14] are probably among the most simplistic ones. While this works well with HTML tags and parts of speech, which have a finite set of values each, the other attributes taken into account in this project are all continuous. One way of overcoming this is to estimate a distribution of the observed continuous values, typically a Gaussian. This enables the estimation of an unknown feature value aided by the equation corresponding to the parameterized distribution.

Contrary to the Bayes approach, which builds a generative model, the logistic regression classifier is based on a discriminative model [16], which, instead of computing p(y|x), finds a direct way of mapping the probability, given the inputs and the class labels. Weka implements this classifier as a multinomial logistic regression model with a ridge estimator. Moreover, regularization can improve the performance of a logit model when dealing with a dataset that has a larger number of predictors compared to the size of the training set [21].

K* [15] is an instance-based learning algorithm which classifies instances by comparing them with the ones from the training set, instead of formulating relations from which to find out which class the instance falls into. This type of learning is also called lazy learning, as it avoids generalization and formality. K* uses entropy as the distance function between data points.

The final approach attempted in classifying attributes and values is by learning propositional rules from the training set and applying them to the test set. JRip [17] is a rule based classifier implemented in Weka based on the RIPPER algorithm which in turn is based on the IREP [22] algorithm, bringing several enhancements and additional heuristics to improve performance.

5. Experimental Results

5.1. Training Set

A training set of manually labeled pages has been created and used with all the classifiers in all experiments detailed in this section. The URLs of the pages were obtained using Google search results from 4 retail websites: Ebay (www.ebay.com), Adorama (www.adorama.com), Cameta (www.cameta.com), and Ritz Camera (www.ritzcamera.com). The search query was "*digital slr camera specifications*". The top 50 web pages for this query were downloaded and manually tagged. These pages were then parsed into a total of 34,339 instances (HTML elements). Out of these, 3,846 were manually labeled: 1,752 as attributes and 2,094 as values.

5.2. Evaluation Set

For each classifier, two runs/experiments were made with the same training set, but with two different test sets. Both test sets consisted of web pages obtained via Google search, using the same query ("*digital slr camera specifications*"). The first dataset was assembled from the same 4 websites used for extracting the training set, while the second one was built by extending the search to the whole web. Google search was used each time (there was no caching of the results). In the following sections, the classifications will be called "full search" for the case of results of searching on the entire web and "restricted search" for the case of searches run only on the four websites used for building the training set.

Both test sets contained about 50 web pages for each search query. The number is not exact because occasionally, HTTP errors occurred when attempting to download and parse some pages.

The full search test set contains about 40,000 instances, while the restricted search set consists of approximately 36,500 instances. Of these, in the full searches, an average of 1,810 were labeled as attributes or values, while in the restricted searches there were nearly twice as many (around 3,300). Table 1 contains the results obtained by the classifiers, both for full search (the rows containing "Full" in the first column) and for restricted search (the rows containing "Rest." in the column). The first and second best results for the restricted search scenario are highlighted using bold and underscore respectively.

Table 1

Comparison of the results obtained for attribute and value identification by various classifiers

	Labeled	Correct	Labeled attributes	Correct attributes	Labeled values	Correct values
Full C45	1527	414	667	239	860	175
Full JRip	782	106	552	16	230	90
Full K*	3361	441	1888	134	1473	307
Full LR	618	399	184	55	434	344
Full NB	2765	414	2325	239	440	175
Rest. C45	3053	<u>2700</u>	1285	<u>1087</u>	1768	**1613**
Rest. JRip	2541	1820	997	684	1544	1136
Rest. K*	2653	1853	1365	698	1288	<u>1155</u>
Rest. LR	3727	1353	907	360	2820	993
Rest. NB	4363	**2799**	3872	**2347**	491	452

Several performance indicators were computed after the resulted classified web pages were manually analyzed and corrected: precision, recall, specificity, negative predictive value, fallout, false discovery rate, false negative rate, accuracy, and F1 measure. Out of all these indicators, the most representative measures for assessing the performance for this type of problem are precision, recall, accuracy, and F1 measure. Fig. 2 contains a comparison of the results obtained by the classifiers for each of these performance indicators in the case of the restricted search evaluation.

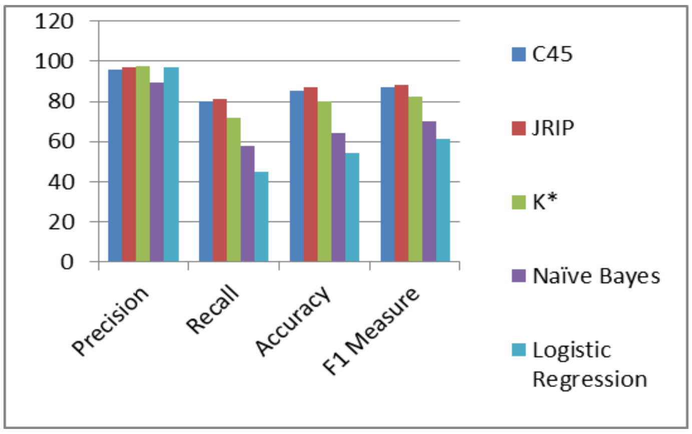


Fig. 2. Aggregate scores for detection of attributes and values in restricted search

In a similar way, Fig. 3 contains the same indicators for the full search experiment.

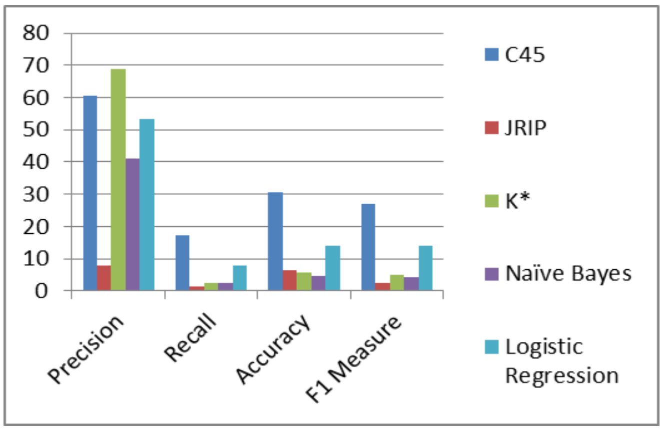


Fig. 3. Aggregate scores for detection of attributes and values in full search

5.3. Discussion

For the restriction search test set, the highest scores for recall (R=0.81), accuracy (0.84), and F1 measure (F1=0.87) were achieved by JRip rule-based classifier. Its precision was P=0.96, also a very large score, but precision is not very important in this case as all classifiers obtained a precision lager than 0.90. As the restricted search dataset contains web pages collected from the same web sites as the training set, naturally their structure will match closer to the ones that were discovered in the training set. The best scoring algorithm in terms of precision was the K* classifier with P=0.97. The rationale is the same: in a feature space dominated by HTML tags, the depth in the DOM tree and the part of speech of the text in the DOM node, the attribute and the value nodes from the classification set will cluster closely together with their counterparts from the training set and the decision tree classifier will discover these rules with a high precision.

However, the full search dataset contains web pages from sources that were not available in the training set, some of which had attributes and values expressed in free text (e.g. reviews, news articles, etc.) and not in a tabular format delimited using HTML tags. This is why the performance is well below the restricted dataset for all classifiers. The C4.5 classifier achieved the highest recall (R=0.17), accuracy (0.31) and F1 score (F1=0.26), also being the second best with regard to precision (P=0.61). A surprise of the full search was the K* classifier which obtained the best precision, P=0.68, outperforming C4.5 in this sense, but with much worse results on all other indicators.

With precision as the measure of choice reported in most of the related works, the proposed system achieves better results when considering restricted search. However, in the case of full searches, the system performed worse than Ravi and Pasca's S_{hier} approach [1] (our precision was P=0.68, compared with P=0.74) and Wang and Zhu's table search engine [3] (which had P@1=0.80, but on a slightly different and simpler task). This could be explained by the fact that content based methods, as discussed in section 2, are better at generalization information from various sources. As most of the features used by our system are not able to catch semantic information, being mostly layout-based, none of the classifiers had impressive results when dealing with unstructured text, such as reviews or user comments. However, they were highly efficient when coming across pages containing information structured in a familiar way using tabular data and other HTML separators, similar to the training set.

6. Conclusions and Future Work

A system aimed at extracting attributes and values from various types of web pages has been developed in an attempt to extract richer information from the web. These pages are obtained by searching for specific products using a search engine. A subset of the retrieved results have been manually analyzed and labeled, while the rest have been used for testing the accuracy of the system. To this purpose, five different classifiers have been compared: Naïve Bayes, C4.5, Logistic Regression, K*, and JRip. To detect attributes and values in web pages, the following features have been employed aiming at understanding the diversity of the text in each HTML element and also in the XPath selector within the DOM structure: text entropy, HTML entropy, XPath length, HTML tag, number of child nodes, number of sibling nodes, and part of speech.

Our results show that the C4.5 decision tree classifier achieves very good results both for the restricted search and for the full search test datasets. Moreover, the restricted search experiment shows that the results obtained for new documents from the same data sources as the one in the training set are much better than when using the same classifier with documents from different sources, including text comments, news, and blog items.

There are two major additions that can be brought to the system. The first one is to link the attributes and values discovered by the system in order to extract attribute-value pairs. This can be done using certain heuristics – for instance, if a tagged attribute and a tagged value are close to each other in the DOM tree and the attribute is before the value, then it is likely that the value belongs to that attribute. Natural Language Processing (NLP) techniques can also be used, linking numbers and adjectives to nouns and analyzing the semantic sense of the lexical construction the two tokens form together.

The second improvement is to extend the system to work with unstructured text as well. Currently, the HTML structure is exploited and most of the information is extracted from it. Precious information is often found in bulks of text, such as reviews, forum posts, etc. However, for this type of pages more complex NLP processing methods are required, such as syntactic or dependency parsing or mining for patterns of words.

To further improve the performance of the system, the greatest benefits can be drawn from enlarging the training set which now consists of data extracted from only four different websites, which mainly use a tabular view to display attributes and values. Adding more pages with heterogeneous structure, including reviews and comments, would further increase the efficiency of the classifiers.

Acknowledgments

The work has been partly funded by the Sectoral Operational Programme Human Resources Development 2007-2013 of the Ministry of European Funds through the Financial Agreements POSDRU/159/1.5/S/132397 and POSDRU/159/1.5/S/ 132395.

R E F E R E N C E S

[1] *S. Ravi and M. Paşca*, "Using structured text for large-scale attribute extraction", CIKM '08 Proceedings of the 17th ACM conference on Information and knowledge management, pp. 1183-1192, 2008

[2] *G.A. Miller*, "WordNet, A lexical Database for English", Communications of the ACM, **vol. 38**, no. 11, pp. 39-41, 1995

[3] *J. Wang, H. Wang, Z. Wang and K. Q. Zhu*, "Understanding tables on the web", ER'12 Proceedings of the 31st international conference on Conceptual Modeling, pp. 141-155, 2012

[4] *W. Wu, H. Li, H. Wang and K. Zhu*, "Probase: a probabilistic taxonomy for text understanding", SIGMOD '12 Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data, pp. 481-492, 2012

[5] *K. Probst, R. Ghani, M. Krema, A. Fano and Y. Liu*, "Semi-Supervised Learning of Attribute-Value Pairs from Product Descriptions", IJCAI'07 Proceedings of the 20th international joint conference on Artifical intelligence, pp. 2838-2843, 2007

[6] *D. Lin*, "Dependency-Based Evaluation of Minipar," in Treebanks: Building and Using Parsed Corpora, Springer Netherlands, pp. 317-329, 2003

[7] *K. Bellare, P. P. Talukdar, G. Kumaran, O. Pereira, M. Liberman, A. Mccallum and M. Dredze*, "LightlySupervised Attribute Extraction for Web Search", Proceedings of Machine Learning for Web Search Workshop, NIPS 2007 Conference, 2007

[8] *A. Blum and T. Mitchell*, "Combining Labeled and Unlabeled Data with Co-Training", COLT: Proceedings of the Workshop of Computational Learning Theory, Morgan Kaufmann, pp. 92-100, 1998

[9] *P. Pantel and M. Pennacchiotti*, "Espresso: Leveraging generic patterns for automatically harvesting semantic relations", ACL-44 Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics, pp. 113-120, 2006

[10] *T. L. Wong, W. Lam and T. S. Wong*, “An Unsupervised Framework for Extracting and Normalizing Product Attributes from Multiple Web Sites”, SIGIR '08 Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval, pp. 35-42, 2008

[11] *D. Koller and N. Friedman*, “Probabilistic graphical models: principles and techniques”, MIT Press, 2009

[12] *W.-S. Han, W. Kwak, H. Yu, J.-H. Lee and M.-S. Kim*, "Leveraging spatial join for robust tuple extraction from web pages," Information Sciences, **vol. 261**, pp. 132-148, 2014

[13] *R. Quinlan*, “C4.5: Programs for Machine Learning”, Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, 1993

[14] *D. D. Lewis*, “Naïve (Bayes) at forty: The Independence Assumption in Information Retrieval”, ECML '98 Proceedings of the 10th European Conference on Machine Learning, pp. 4-15, 1998

[15] *J. G. Cleary and L. E. Trigg*, “K*: An Instance-Based Learner Using an Entropic Distance Measure”, Proceedings of the 12th International Conference on Machine Learning, pp. 108-114, 1995

[16] *A. Y. Ng and M. I. Jordan*, “On Discriminative vs. Generative Classifiers: A Comparison of Logistic Regression and Naïve Bayes”, Proceedings of NIPS 2002 Conference, 2002

[17] *M. Sasaki and K. Kita*, “Rule-Based Text Categorization Using. Hierarchical Categories”, 1998 IEEE International Conference on Systems, Man, and Cybernetics, p. 2827-2830, vol. 3, 1998

[18] ***, Google Custom Search APIS, https://developers.google.com/custom-search/json-api/v1/overview, accessed at 15 November 2015

[19] *J. Hedley*, “jsoup: Java HTML Parser”, http://jsoup.org, accessed at 15 November 2015

[20] *The Apache Software Foundation*, “Apache OpenNLP”, https://opennlp.apache.org/, accessed at 15 November 2015

[21] *M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann and I. H. Witten*, “The WEKA Data Mining Software: An Update” ACM SIGKDD Explorations Newsletter, **vol. 11**, Issue 1, pp 10-18, 2009

[22] *W. W. Cohen*, “Learning to Classify English Text with ILP Methods”, Advances in ILP, ed. L. de Readt, IOS Press, 1995