Chapter 8

SEMI-SUPERVISED PRODUCT SPECIFICATIONS EXTRACTION FROM THE WEB

George Krys[1], Ebrahim Bagheri[2]
[1]*School of Computing and Information Systems, Athabasca University*
[2]*George Vari Engineering and Computing Centre, Ryerson University*

Abstract: This paper introduces an approach that utilizes the inductive semi-supervised learning strategy – self-training model for extracting property-value pairs from a collection of Web pages. The proposed work employs a novel concept of short properties and values for learning high confidence property-value pair seeds. The seeds are then used to discover repetitive HTML formatting patterns and consequently, using these patterns as the wrappers to extract the rest of the property-value pairs that the Web page contains. The experimentations on the collection of Web pages, drawn from over one hundred diverse, real life electronic goods retailer Web sites, show promising results.

Key words: Learning — Knowledge Acquisition, Semi-Supervised Machine Learning, Web Information Extraction, Product Property Name and Value Pair

1. INTRODUCTION

With billions of dollars already spent by digital shoppers each year and favorable forecasts of capturing even larger share of sales in the near future, online retail is one of the fastest growing industries. The success of e-commerce in turn has given rise to online aggregators, search engines, comparison shopping sites, shopping portals, deal and auction sites that help users sift through the retail debris. However, each online aggregator faces the

extremely challenging problem of building a cohesive product catalog, which can be leveraged for smarter search and better customer experience.

The information about products is very volatile, i.e., they are frequently updated and typically generated dynamically on the fly when a user wants to see the product specifications. The specification Web pages vary in HTML formatting and are often only optimized for human browsing rather than machine processing. There are several irregularities in the presentation of product property-value pairs (PVPs) on numerous retailers' Web sites, such as layout formats that range from regular table, list to unstructured free text, usage of different product properties or different names of the same properties or values (i.e., synonymous words or phrases having the same or similar meaning, abbreviations), different ordering of properties from site to site as well as omission of some PVPs or values even on the same site. Moreover, the product detail pages, besides product specifications contain other blocks of information such as toolbars and navigation bars that contain links for browsing the Web site inventory, additional information about the product, customer reviews, among others. Although these supplementary informational blocks can enhance consumer appeal, usability, and visual attractiveness, it brings great challenge for locating the PVPs on the page in order to automatically extract them.

In light of these challenges, the key objective of our work is to develop an efficient algorithm capable of tackling the presentation irregularities and specification block identification obstacles of product specification pages for extracting PVPs. We employ a combination of semi-supervised machine learning and pattern mining techniques for discovering and extracting property-value pairs from product specification Web pages. In our self-training machine learning model, we introduce the novel concept of short properties and values for discovering high confidence PVP seeds. The seeds are then used to discover repetitive HTML formatting patterns found surrounding the PVPs. These discovered patterns serve as the wrappers to extract the rest of the property-value pairs that the Web page contains.

2. RELATED WORK

The procedure for extracting content from a Web page based on the knowledge of its format is often referred to as 'wrappers' and the process of information extraction (IE) is known as Wrapper Induction (WI). Formally, a wrapper is a function for extracting the related content from a Web page while discarding the irrelevant text (Kushmerick, 2000). The main wrappers' assumption is that one can find repetitive patterns in one or more Web pages that conform to a common template. The goal of WI is to automatically

generate a wrapper that is used to extract the targets from an information resource.

In a supervised wrapper induction (Kushmerick, 1997; Wang & Hu, 2002), a set of extraction rules are learnt from a set of manually labeled pages that are used to extract data items from similar pages. Because manually labeling pages is labor intensive, much effort has been devoted to automating the wrapper generation process by employing semi-supervised or unsupervised machine learning (ML) techniques (Chang et al., 2006).

IEPAD (Chang & Lui, 2001) is one of the first IE systems that semi-automatically generalizes extraction patterns from unlabeled Web pages. The system identifies record boundaries by repeated pattern mining and multiple sequence alignment utilizing a PATRICIA trie data structure (Gusfield, 1997; Morrison, 1968). However, the method described in (Chang & Lui, 2001) produces a large number of irrelevant patterns, and in several cases fails to find the right one. It mainly occurs because the domain-specific information is not utilized in this approach. To deal with this issue, IPEAD has an interface for users called pattern viewer through which they can choose a proper record extraction pattern. In contrast, our method integrates domain-specific information into the learning mechanism, which aids in the discovery of the relevant patterns and hence requires much less direction from the user.

Some of the most cited unsupervised systems include RoadRunner (Crescenzi, Mecca, & Merialdo, 2001) and ExAlg (Arasu & Garcia-Molina, 2003). These systems compare HTML pages and generate a wrapper based on their similarities and differences. The process, starts by taking one input page as an initial version of the wrapper, then it is matched against the test page and iteratively refined solving mismatches by generalizing the wrapper. More recently, comparable unsupervised adaptive template-based method was proposed in (Tang et al., 2012). The method includes constructing the attribute word bag using Web titles from a single domain. The bag of words is then used to learn high-quality page templates by selecting most frequent patterns across multiple pages. However, all these methods require sufficient number of equivalent pages to discover their common templates. Our algorithm, in comparison, is template-independent; that is, given a small dictionary of short properties and values, the product's PVPs can be extracted even from a single document.

Similar template-independent but unsupervised approach is proposed in (Walter, 2012). The approach combines several different features from a document's tokens, its tree representation, and visual information for clustering Web page elements. The element lists are then purged to drop insignificant clusters and create extraction candidates. Finally, based on the best-rated candidate a wrapper consisting of a set of XPath queries is created and executed to extract actual product specifications. However, the main

drawback of this approach is the quite high costs in terms of time. Even though during the algorithm evaluation the extraction routines that exceeded 100 seconds were terminated, an average runtime to process a Web page was 13 seconds, which makes the method infeasible in practice.

The pivotal point of our research was the algorithm for constructing document-specific character-level wrappers automatically proposed in (Wang & Cohen, 2009). The algorithm finds maximally long contextual strings that bracket at least one seed instance of every seed using PATRICIA tries. Although the main focus in (Wang & Cohen, 2009) is on the expansion of a partial set of "seed" objects into a more complete set by extracting named entities with wrappers, we adapted this method for binary relations (i.e., property and value pairs) extraction and received very positive results.

Lastly, our work bears a resemblance to the system proposed in (Wu et al., 2009). The paper describes a semi-supervised, template-independent method, which uses a few manually labeled pages for a product domain. The labeled pages are combined with unlabeled ones to boost the learning of candidate attributes using a co-training algorithm with Naïve Bayes classifier. The candidates are used to identify product specification blocks on the page and to extract data of interest. However, unlike our approach, the newly identified name-value pairs (NVPs) are only temporarily added to the training set. They are removed from the training set when the classifier moves to the next page. It generally means that for each page the learning starts from scratch (i.e., original training set) and the process is not enhanced with new knowledge after it progresses to the subsequent pages. As stated in (Wu et al., 2009), the reason for this is that the misclassified text nodes cause semantic drift and lead to many invalid extractions. Our approach solves the semantic drift problem by utilizing only the high confidence short property and values. It also progressively increases the number of learned PVPs with each processed page, which, in turn, leads to higher precision and recall.

3. PROPOSED TECHNIQUE

In order to discover patterns and extract PVPs from product specification Web pages, we propose a semi-supervised, template-independent method. As illustrated in Fig. 1, the method comprises four main tasks: preprocessing, seed learning, pattern discovery, and pattern-based extraction.

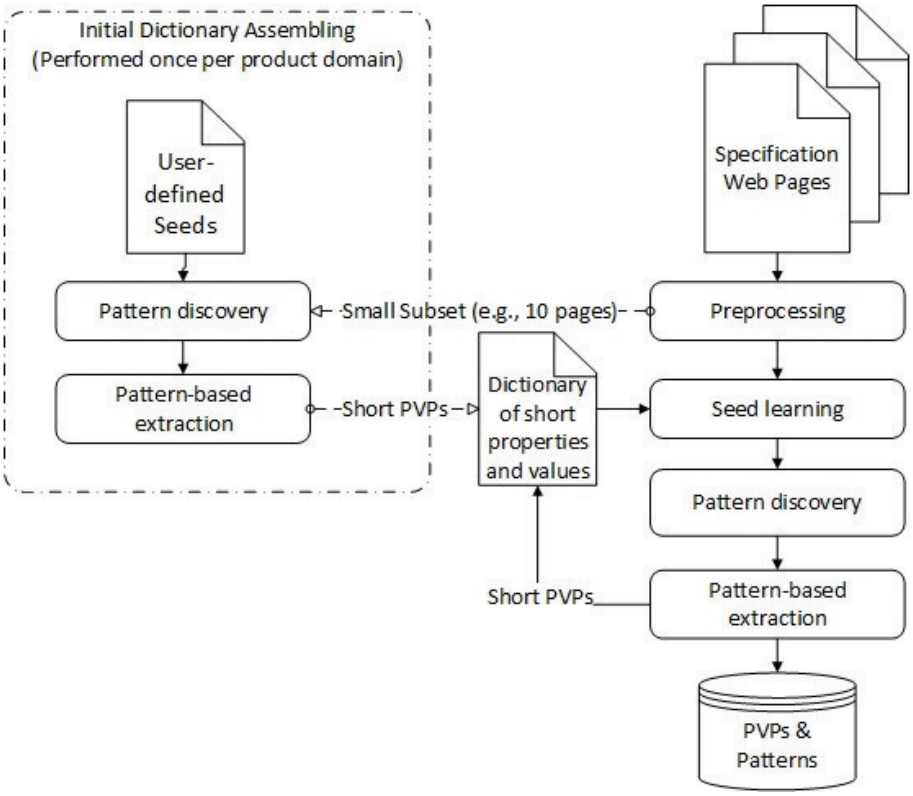

Fig. 1. The proposed semi-supervised property-value pair extraction algorithm.

The following subsections present the details about the four steps of the proposed technique.

3.1 Preprocessing

To reduce the amount of characters that need to be processed by the character-level pattern discovery algorithm, the input HTML document is parsed chunk-by-chunk and converted to a clean source document. The chunks are the open tags, closed tags, and the text between the tags, which are referred to as the text chunks in this paper. A text chunk may contain irrelevant text (i.e., neither property nor value, nor PVP) or either property or value, or PVP. According to the relative position of a property and value, we assume that a Web page may meet two conditions: i) the product property name and value, which constitute a PVP, organized in different but adjacent text chunks (e.g., <tr><td>Camera Resolution</td><td>3.5 megapixels</td></tr>); ii) the product property name and value belong to the same text chunk and separated by a symbol such as the colon symbol ":" (e.g., Camera Resolution: 3.5 megapixels).

The preprocessing task includes removing new line symbols, tabs, and extra white spaces as well as tags that certainly do not surround PVPs (e.g., header, script, image, etc.) and contents between these opening and closing tags. To ensure presence of repetitive patterns in HTML formatting, we also remove unique attribute values (i.e., id and name) and strip any other attributes and their values of all the symbols, white spaces, and numbers (e.g., <th scopecolclassspecvaluecol>). The text chunks are checked for a separator (e.g. ':') and only the first separator in the chunk is retained if there is more than one separator in the text chunk. Additionally, we automatically close <tb>, <th>, and <li> tags if they do not have corresponding closing tags.

3.2 Seed Learning

One of the main steps of our technique is to identify a few (at least two) PVPs that can be used to learn the wrapper for the rest of the PVPs. We refer to these base PVPs as seeds. To automatically learn the seeds and discover patterns, we use the following two basic assumptions: (1) a property always comes before a value, and (2) the Web page uses some repetitive formatting for presenting PVPs. If there is a repetitive pattern, then only a small number of PVPs are required for its discovery.

Our approach to seed learning employs the inductive semi-supervised machine learning strategy – self-training model and is based on the observation that there is a limited number of common short property-value pairs in any product domain. (e.g., Weight: 1 kg, Display Resolution: 1024 × 768). Because of the limited number of common short PVPs and very little variety in their unique spelling, no typical natural language processing (NLP) tasks were required (i.e., POS tagging, stemming, etc.).

3.2.1 Dictionary of Short Properties and Values

The automated seed learning process is preceded by the initial semi-automated assembly of the domain dictionary of short properties and values. The semi-automated process is shown in the dashed box in Fig. 1. The process uses a small subset of pre-processed product specification pages from a specific domain (e.g., 10 pages) and user-provided seeds to automatically discover patterns and extract PVPs. The extracted short PVPs during this stage are saved to the dictionary and serve as the initial feature set for the semi-supervised machine learning algorithm of the main process.

We tested two types of domain dictionaries (aka feature types): i) Bag-of-Words (BOW), and ii) n-grams (NGRAM). The BOW dictionary contains words found in short properties and values disregarding the word order. The NGRAM dictionary (i.e., bigram, trigram, …, n-gram where n is the number

of tokens) contains entries of a property or value text string without the white spaces between the words (e.g., CAMERARESOLUTION); thus preserving the word order. Both of the dictionaries maintain the number of each unique word or n-gram occurrence in the training set.

3.2.2 Classifiers

To automatically discover the seeds in the page, we experimented with two advanced classifiers, namely Bayes Point Machine (BPM) and Support Vector Machine (SVM) as well as with a simple probabilistic Naïve Bayes classifier. The initial domain dictionary of short properties and values entries (tokens) serve as the set of features and the documents that the dictionary is built on – as the training data for the classifiers. The classifiers are to predict P (Property) or V (Value) labels for each text chunk of the tested Web page whose number of words do not excessed the token limit. The token limit is the maximum number of tokens (i.e., number of words) for property and value pairs to be accepted as a short PVP.

For BPM and SVM classifiers, a vector whose dimensions are equal to the number of features (i.e., number of property and value entries in the dictionary) represents the test data. The BPM classifier in our experiments uses the Bernoulli model of features (i.e., binary occurrence information) ignoring the number of occurrences. The train\test text string is represented as one row of comma separated binary values where the value 1 marks the presence of a word (i.e., feature) in the train\test text string. The position within the string indicates an ordinal position of the entry (i.e., feature) in the dictionary. The SVM classifier, on the other hand, takes into consideration the number of occurrences of a feature in a training set. The tested text string for SVM is represented similarly to the following string of values: 1:3,32:1,56:45, where the first number is the ordinal value of the feature and second is the number of occurrences of the word in the training set. Since both, BPM and SVM assign one of the labels to all tested text chunks, we experimentally determined the classifiers' cut off scores to separate the high and low confidence predictions.

The Naïve Bayes (NB) classifier matches individual tokens of the provided text chunks (i.e., test data) with the dictionary of short properties and values. It calculates the probability of the text chunk being a property or value, then performs the sorting operation on the probabilities. The first element in the sorted list of probabilities (i.e., highest probability) reveals the label to assign. However, if the probabilities of the text chuck belonging to one of two categories are the same, the classifier assigns U (Undetermined) label; in other words, the classifier cannot decide which category it belongs to because of a

lack of information. This "naivety" allowed accepting any text chunk labeled as P or V as the high confidence predictions of the NB classifier.

Lastly, after the tested text chunks were labeled by any of the above classifiers, a text chunk classified as property was selected as a part of a seed only if it is directly followed by the text chunk classified as value.

3.3 Pattern Discovery

A pattern (aka wrapper) in our proposed approach is a subclass of regular expressions over an alphabet of tokens. An example of a pattern is shown below:

<tr><td>(.*?)</td><td>(.*?)</td></tr>

where <tr><td>, </td><td>, and </td></tr> are tokens that match groups of HMTL tags and the (.*?) is a wildcard that matches any string of characters (i.e., including HTML tags). The following string matches the above pattern:

<tr><td>Camera Resolution</td><td>3.5 megapixels</td></tr>

To discover patterns, we first find all instances of the identified seeds (from Section 3.2) within the document of interest. The instances are grouped by the context that separates properties and values. The middle context as well as three-character long left and three-character long right context is maintained with each instance. Table 1 shows an example of seed instances found for one of the corpus documents.

Table 1. Example of seed instances.

Left	Property	Mid	Value	Right
ol>	Sensor Type	</td><td classcol>	CMOS	</t
ol>	Focal Length	</td><td classcol>	8.50 mm	</t
ol>	Color	</td><td classcol>	Black	</t
li>	Sensor Type	:	CMOS	</l

In the next step, we filter out the seed instances that do not have at least one counterpart with the same left, middle, and right context (i.e., Sensor Type: CMOS in our example), then join each PVP instance together using the middle context (e.g., Col-or</td><td classcol>Black). Next, for each seed instance, the system extracts a left character string, which starts from the first character of the document and ends before the first character of the joint PVP and a right character string, which starts from the first character after the joint PVP and ends at the last character of the document. The left character strings is inserted into the left-context PATRICIA trie. The right character string is inserted in reverse character order into the right-context PATRICIA trie. Every node in both tries maintains a list of IDs for keeping track of the seed instances that follow the string associated with that node.

As a final step, the tries are used to find maximally long contextual strings that bracket at least one seed instance of every seed. Similar to (Wang & Cohen, 2009), we first find all the longest possible strings from one trie, then for every such string, find the longest possible string from another trie such that both of them bracket at least one occurrence of every given seed in the document. Patterns are assembled for PVP extractions using the found left and right contextual strings and the middle context of the group of seeds

3.4 Pattern-based Extraction

It should be noted that the longest character-level prefixes and suffixes found in the method proposed in Section 3.3 are not always "HTML-compliant" and some-times produce patterns similar to the following example:

/td></tr><tr><td width>(.*?)</td><td>(.*?)</td></tr><tr><td width>

However, one of the major assumptions of our approach is that there has to be at least one HTML pattern (i.e., HTML-compliant pattern) in order to extract PVPs. Therefore, our algorithm attempts to normalize patterns by removing all the incomplete tags (i.e., /td>), tags that have no matching opening or closing counterparts, and tags that surround other tags, such as table and table row tags, which surround table cell tags (i.e., < table>, <thead>, <tbody>, <tr>), and list tags which surround list items tags (i.e., <ul>, <ol>, <dl>). The invalid HTML pattern example above would be normalized to the following pattern: <td width>(.*?)</td><td>(.*?)</td>

The normalized (if required) pattern is used for PVP extraction. The regular expression is applied to the document of interest in both directions: left-to-right and right-to-left, then the two groups of regular expression matches are used to assemble property-value pairs. The reason for two-directional application is that in some cases, specifically with patterns based on , <p>, or HTML blocks, the first property or the last value of the extracted PVPs may include extra markup and text. When left-to-right and right-to-left extracted PVPs do not match, our system validates the inner HTML of non-matching items and selects shorter strings that do not contain any markup or longer strings only if they have valid HTML.

After all PVPs are extracted, the relative short PVPs (e.g., three-word property and three-word value) are added to the dictionary and their counts of occurrences are increased. These PVPs are used in the feature set for the subsequent classification. Finally, the discovered patterns and extracted PVPs are saved to the database for product specification cataloging.

4. EVALUATION

The evaluation of the proposed approach was performed using the standard IE performance measures of precision, recall, and F_1 score. The corpus used in the experiments was collected from over one hundred distinct electronic goods retailers' Web sites from two domains – Digital Cameras and Smartphones. In our experiments, we focused on the ability of the algorithm to discover broad variety of patterns and therefore selected specification Web pages that are uniquely formatted.

The entire corpus was semi-automatically processed; and then manually examined, edited and together with missing PVPs combined into *page item set* – a set of actual PVPs that the document contains. The page items of each document were saved to the database and served as the ground truth test data set in order to benchmark our proposed algorithm. Ten documents from each product domain and their page items also served as the seeds for performing the semi-supervised learning.

The experiments were performed for two types of PVP extractions: (1) utilizing known seeds (i.e., user provided seeds), in order to assess effectiveness of the pattern discovery functionality, and (2) utilizing learned seeds, in order to identify best suitable ML classifier and corpus processing strategy.

4.1 Pattern Discovery Evaluation

To evaluate the effectiveness of the developed PVPs pattern discovery technique, the accuracy of the technique was measured on the gold standard corpus. To infer the unbiased performance of the algorithm, its precision, recall, and F_1 scores were calculated for exact matches (i.e., actual PVP matched expected PVP word-for-word) and for partial matches (i.e., actual PVP was close enough to the expected PVP but contained some additional, irrelevant words or had some words missing).

The results of our tests showed that the character-level approach for constructing document-specific wrappers works very well for discovering patterns and extracting property-value pairs from Web pages. Its application allows discovering patterns not only for properties and values structured in simple tables and lists but also uncommon, unique PVP formatting (i.e., patterns based on or HTML blocks) similar to the ones shown in Table 2.

Table 2**.** Examples of discovered patterns

URL	http://estore.canon.ca/webapp/wcs/stores/servlet/prouct_12152_10102_10355_24
Pattern	(.*?) (.*?)

URL	http://www.naaptol.com/digital-cameras/olympus-point-shoot-vg-150/p/3683646.html
Pattern	<div classtechdsclayout><div classclm> (.*?)<span classhelpicon><div idclasshelpdatastatehiddenstyledisplaynonepositionabsolute> <div classclm> (.*?)
URL	http://store.sony.com/xperia-z1-zid27-xz1c6902//cat-27-catid-all-unlocked-phones?vva_colorcode=000000&_t=pfm%3dcategory
Pattern	<h5 classwsproducttitlefndatadynamicblockiddatadynamicblockname>(.*?)</h5><div classwsgroupsnyproductrating><div classwsgroupcontentssnyproductratingcontentsstylewidthpx><div\ classwsgroupsnyratingstars><div classwsgroupcontentssnyratingstarscontents> <div classsnyavgrating>(.*?)

Table 3 presents the final results of the experiments. In the first row of the table the partial match extractions were counted as both – False Positive and False Negative extractions; in the second, they were counted as True Positive extractions. As the test results indicate, the implemented algorithm is very effective in discovering patterns for PVP extraction.

Table 3. Evaluation results of the pattern discovery algorithm performance utilizing known seeds.

	Expected	**Actual**	**TP**	**FP**	**FN**	***Precision***	***Recall***	F_1
Exact	4848	4841	4790	51	60	99.34%	98.90%	99.11%
Partial	4848	4841	4827	15	22	99.77%	99.37%	99.55%
TP – true positive, FP – false positive, FN – false negative.								

4.2 Seed Extraction Evaluation

The objectives of the second set of experiments was to find the most suitable ML classifier for identifying PVPs and determine the corpus processing strategy, which will deliver the highest precision, recall, and F_1 score. To meet the former objective, we integrated and extensively tested the following classifiers: i) Microsoft Infer.NET Bayes Point Machine [1], ii) .NET implementation of LIBSVM[2], and iii) a custom implementation of a simple Naïve Bayes classifier.

To find the best-suited classifier for the task, we compared the performance of BOW, NGRAM and BOTH feature types for each classifier. The BOTH type works in the following fashion: first the system attempts to find seeds using the NGRAM approach, then, if it does not succeed (i.e.,

[1] http://research.microsoft.com/en-us/um/cambridge/projects/infernet/
[2] http://www.matthewajohnson.org/software/svm.html

number of found seeds is less than 2 and only then, it uses BOW approach. The chart in Fig. 2, shows that the BOW results in higher recall and NGRAM in higher precession as well as that all classifiers deliver identical results with BOTH feature types setting. The BOTH approach smoothens the difference between BOW and NGRAM precision and recall and delivers highest F_1 score. Therefore, since all classifiers with BOTH settings deliver identical precision and recall, the only performance indicator that discriminates them is the task execution time: around 8 min for BPM, 7 min for SVM, and 2 min for NB on commodity hardware with the in-house system optimized for research (i.e., logging and output to the screen of each processing step during the test) rather than efficient performance. Thus, NB classifier was selected as the best suited classifier for the task because of the shortest processing time.

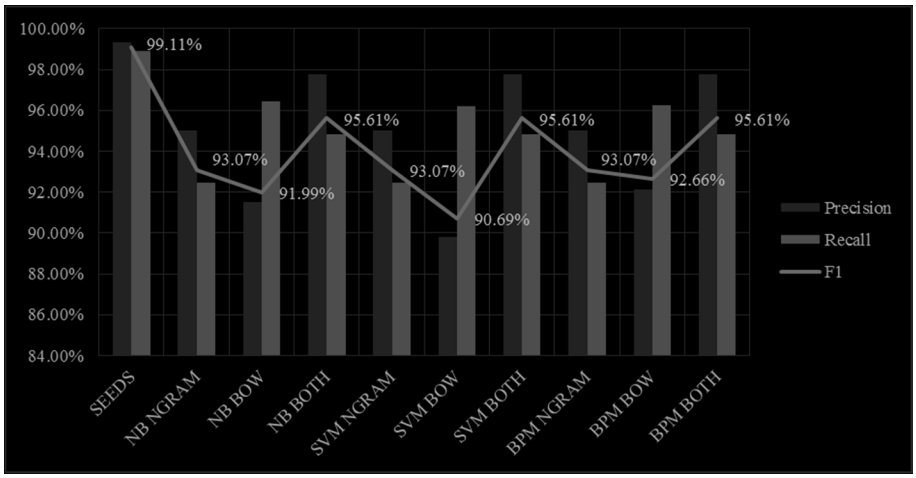

Fig. 2. Feature type evaluation results.

After the classifier was selected, we conducted an experiment to determine the corpus processing strategy. In this experiment, four different strategies were compared with each other as well as with the corpus processing results utilizing known seeds. The four strategies as follows:

1. *One iteration NB BOTH plus an extra iteration with BOW features and increased token limit.* The NB BOTH processing was enhanced with an extra iteration in which the failed documents (i.e., documents that did not produce any extractions) were processed with BOW features and the token limit was increased by two. The purpose of increasing the token limit is to successfully process the Web documents that might not have enough 3 token PVPs (default) for discovering patterns (i.e. less than two short PVPs per page).
2. *One iteration NB BOTH with 20 training set pages plus an extra iteration with BOW features and increased token limit.* The same arrangement as in

the previous strategy but the training set was composed of 20 corpus documents that contained the maximum number of PVPs. The aim was to test if changing the training data will have effect on the processing results.

3. *Two iteration NB BOTH plus an extra iteration with BOW features and increased token limit.* The first iteration performed with the default settings. In the second iteration, all documents are processed using features collected during the first iteration and only features that do not already exist in the dictionary of short properties and values were added to the list.
4. *Three NB NGRAM iterations plus an extra iteration with BOW features and increased token limit.* The aim of this strategy was to test if using primarily NGRAM features will reduce the number of erroneous extractions. The second and extra iterations were similar to the ones used in the previous strategies. In the third iteration only the failed pages were processed using BOW features and no features were added in this iteration to the dictionary of short properties and values.

The chart in Fig. 3 illustrates that the strategy number three (two iterations) delivers the best results. We believe that the proposed technique can easily be extended to include other product domains. However, greater variety of domains needs to be used in order to get more conclusive results.

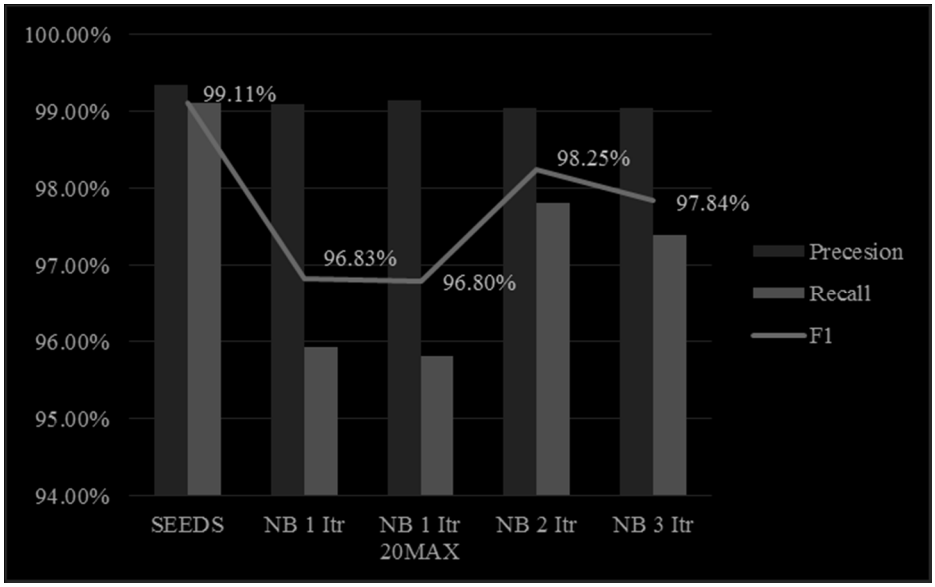

Fig. 3. Corpus processing strategies evaluation.

5. CONCLUSIONS

In this paper, we have proposed an algorithm for extracting property-value pairs from a collection of Web pages. The algorithm employs inductive semi-supervised learning strategy – self-training model and a novel concept of short properties and values for learning high confidence property-value pair seeds. The seeds were then used to discover repetitive HTML formatting patterns and consequently, using these patterns as the wrappers, extract the rest of the property-value pairs that the Web page contains. The key feature of this approach is the focus on semi-supervised learning of a limited number of short property-value pairs per product domain, which normally do not vary in spelling. It greatly reduces the annotation effort and amount of data (i.e., Web page content) that needs to be processed, streamlines the ML task because no typical NLP preprocessing is required (i.e., POS tagging, stemming, etc.), and allows usage of simplistic classifiers such as Naïve Bayes, all of which in turn makes the entire process very efficient. The empirical testing on the collection of Web pages, drawn from over one hundred diverse, real-life electronic goods retailers' Web sites indicate that the algorithm performs well.

REFERENCES

Arasu, A., & Garcia-Molina, H. (2003, June). Extracting structured data from Web pages. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of data* (pp. 337-348). New York, NY: Association for Computing Machinery.

Chang, C.-H., Kayed, M., Girgis, M., & Shaalan, K. (2006). A Survey of Web Information Extraction Systems. *IEEE Transactions on Knowledge and Data Engineering, 18*(10), 1411–1428. doi:10.1109/TKDE.2006.152

Chang, C.-H., & Lui, S.-C. (2001, April). IEPAD: information extraction based on pattern discovery. In V. Y. Shen (Ed.), In *Proceedings of the 10th International Conference on World Wide Web* (pp. 681–688). New York, NY: Association for Computing Machinery.

Crescenzi, V., Mecca, G., & Merialdo, P. (2001, September). RoadRunner: Towards Automatic Data Extraction from Large Web Sites. In *Proceedings of the Twenty-seventh International Conference on Very Large Data Bases* (pp. 109–118). San Francisco, CA: Morgan Kaufman Publishers Inc.

Gusfield, D. (1997). *Algorithms on strings, trees, and sequences: computer science and computational biology*. New York, NY: Cambridge University Press.

Kushmerick, N. (1997). *Wrapper induction for information extraction* (Doctoral dissertation). University of Washington

Morrison, D. R. (1968). PATRICIA - Practical Algorithm To Retrieve Information Coded in Alphanumeric. *Journal of the ACM*, *15*(4), 514–534. doi:10.1145/321479.321481

Tang, W., Hong, Y., Feng, Y.-H., Yao, J.-M., & Zhu, Q.-M. (2012). Simultaneous Product Attribute Name and Value Extraction with Adaptively Learnt Templates. In *Proceedings of 2012 International Conference on Computer Science and Service System (CSSS)*, (pp. 2021–2025). Washington, DC: IEEE.

Wang, R. C., & Cohen, W. W. (2009, August). Character-level analysis of semi-structured documents for set expansion. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*: *Volume 3* (pp. 1503-1512). Morristown, N.J: Association for Computational Linguistics.

Wang, Y., & Hu, J. (2002, May). A machine learning based approach for table detection on the Web. In *Proceedings of the 11th international conference on World Wide Web* (pp. 242–250). New York, NY: Association for Computing Machinery.

Walter, M. (2012, November). Unsupervised Extraction of Product Information from Semi-structured Sources. In *Proceedings of the 13th International Symposium on Computational Intelligence and Informatics* (pp. 257–262). Washington, DC: IEEE Computer Society.

Wu, B., Cheng, X., Wang, Y., Guo, Y., & Song, L. (2009, September). Simultaneous Product Attribute Name and Value Extraction from Web Pages. In *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology* (pp. 295–298). Washington, DC: IEEE Computer Society