Semi-Supervised Product Specification Extraction from Web Pages

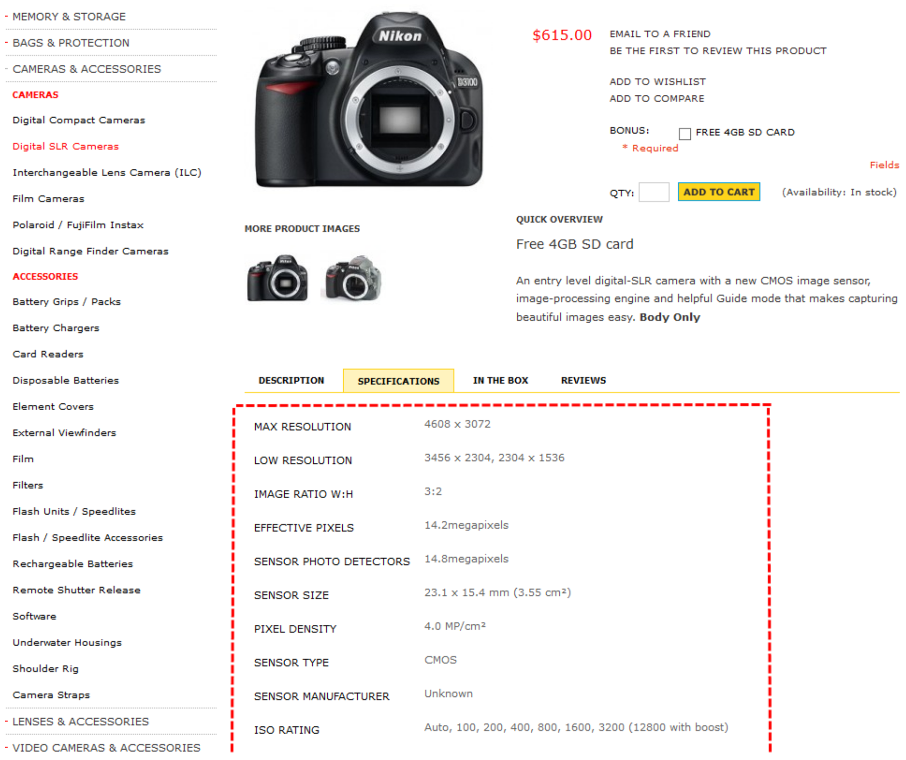
Abstract. This paper introduces an algorithm that utilizes the inductive semi-supervised learning strategy – self-training model for extracting property-value pairs from a collection of Web pages. The algorithm employs a novel concept of short proper-ties and values for learning high confidence property-value 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 ninety diverse, real life electronic goods retailer Web sites shows promising results. The corpus of over one hundred Web documents can be processed on a commodity hardware within a few minutes and deliver as much as 99% precision and 97% recall.

Keywords: Web Information Extraction, Product Name and Value Extraction

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 boom in e-commerce in turn gave rise to online aggregators – search engines, comparison shopping sites, shopping portals, deal and auction sites that help users to sift through the retail debris. However, each online aggregator initially faces extremely challenging problem of building a cohesive product catalog, which can be leveraged for smarter search and better customer experience.

The information about the product that aggregators are interested in is very volatile. It is frequently updated and typically generated dynamically on the fly when a user wants to see the product specifications. The specification Web pages, similar to one shown in Fig. 1, are crawled from the diverse e-commerce sites which vary in HTML formatting and 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. Additionally, the product detail pages, besides product specifications contain several other blocks of information. For example, as shown in Fig. 1, the leftmost block is a navigation bar that contains links for browsing the Web site inventory. There are also other tabs alongside the "Specifications" (highlighted with the red dashed box) that provide additional information about the product as well as customer reviews. Although these additional 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.



- MEMORY & STORAGE
- BAGS & PROTECTION
- CAMERAS & ACCESSORIES

CAMERAS
Digital Compact Cameras
Digital SLR Cameras
Interchangeable Lens Camera (ILC)
Film Cameras
Polaroid / FujiFilm Instax
Digital Range Finder Cameras

ACCESSORIES
Battery Grips / Packs
Battery Chargers
Card Readers
Disposable Batteries
Element Covers
External Viewfinders
Film
Filters
Flash Units / Speedlites
Flash / Speedlite Accessories
Rechargeable Batteries
Remote Shutter Release
Software
Underwater Housings
Shoulder Rig
Camera Straps

- LENSES & ACCESSORIES
- VIDEO CAMERAS & ACCESSORIES

$615.00 EMAIL TO A FRIEND
BE THE FIRST TO REVIEW THIS PRODUCT

ADD TO WISHLIST
ADD TO COMPARE

BONUS: ☐ FREE 4GB SD CARD
* Required Fields

QTY: ADD TO CART (Availability: In stock)

MORE PRODUCT IMAGES

QUICK OVERVIEW
Free 4GB SD card

An entry level digital-SLR camera with a new CMOS image sensor, image-processing engine and helpful Guide mode that makes capturing beautiful images easy. **Body Only**

DESCRIPTION | SPECIFICATIONS | IN THE BOX | REVIEWS

MAX RESOLUTION	4608 x 3072
LOW RESOLUTION	3456 x 2304, 2304 x 1536
IMAGE RATIO W:H	3:2
EFFECTIVE PIXELS	14.2megapixels
SENSOR PHOTO DETECTORS	14.8megapixels
SENSOR SIZE	23.1 x 15.4 mm (3.55 cm²)
PIXEL DENSITY	4.0 MP/cm²
SENSOR TYPE	CMOS
SENSOR MANUFACTURER	Unknown
ISO RATING	Auto, 100, 200, 400, 800, 1600, 3200 (12800 with boost)

Fig. 1. A fragment of a product specification page.

The key objective of our work was to develop an efficient algorithm capable of tackling the presentation irregularities and specification block identification challenges and extracting the data of interest across different sites. We employ 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 learning high confidence PVP seeds. The seeds then used to discover the repetitive patterns of HTML formatting then using these patterns 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 relational content from a Web page while discarding the irrelevant text [7]. The main wrappers' assumption is that repetitive patterns that occur in a Web page or multiple pages conform to a common template. The goal of WI is to automatically generate a wrapper that is used to extract the targets for an information resource.

In a supervised wrapper induction [6, 11], a set of extraction rules are learnt from a set of manually labeled pages then these rules are used to extract data items from similar pages. Because manually labelling pages is labor intensive, much effort has been devoted to automating the wrapper generation process [2]. Beginning with IEPAD [3], RoadRunner [4], and ExAlg [1], the tendency has been to build systems that employ semi-supervised or unsupervised ML techniques for either learning or identifying repetitive patterns in a Web page or across multiple pages. The works by Wu et al. [12], Zheng et al. [13], Tang et al. [9], and Wang & Cohen [10] conform to this trend.

Tang et al. [9] proposed an unsupervised adaptive template based method to simultaneously extract name-value pairs (NVPs) from e-commerce sites. Tang et al. first construct the attribute word bag using Web titles from a single domain. The selection of candidate name-value pairs is based on their organization in DOM tree: adjacent text nodes or the same text node but separated in some way (e.g. ":”). The bag of words is then used to learn high-quality page templates by selecting most frequent patterns across multiple pages. In the next step, attribute-value pairs are extracted from the page by matching each of these templates. Many ideas from Tang et al. research, such as the domain-specific word bag, page preprocessing, noise filtering are adopted in our work.

Zheng et al. [13] introduced an analogous system but utilizing a template-independent method. The system uses short listing of products to find the link to the offer then retrieves and extracts data from “data-intensive page" – a page that contains detailed product information. It also relies on a set of manually collected attribute name synonyms for a given domain and relies on the assumption that attributes occupy a contiguous region in a “data-intensive” page. Thus, the main goal of their approach has been to identify the "data-rich region" boundaries of a specific product. These regions are identified by calculating and comparing structural-semantic entropy of the nodes in a DOM tree. For the experiments reported in [13] authors used an open source HTML parser to clean up the bad HTML tags and automatically close elements with optional end tags. Similar functionality to "fix up" many common mistakes in HTML documents has been implemented in our work.

Wu et al. [12] also proposed template-independent method, which uses few manually labeled pages for a domain. The labeled pages are combined with unlabeled ones to boost the learning of candidate attributes using a co-training algorithm with Naïve Bayesian classifier. The candidates are used to identify product specification block on the page. Wu et al. [12] based their method on the hypothesis that the attribute name and value presented together (i.e., they are contiguous and aligned horizontally or vertically) on the "product details" page. Our algorithm bears some similarity to this work. We also used analogous hypothesis about property and value alignment on the page and semi-supervised ML approach for learning the PVP seeds.

The Set Expander for Any Language (SEAL) system developed by Wang & Cohen [10] expands a partial set of "seed" objects into a more complete set by constructing page-specific extraction rules (character-level wrappers) that are independent of the human language and markup language of the Web pages. The system can expand sets of category instances as well as binary relation instances. SEAL extracts named entities with wrappers, each of which is composed of a pair of character-level prefix and suffix. The algorithm finds maximally long contextual strings that bracket at least one seed instance of every seed using PATRICIA tries (Practical Algorithm to Retrieve Information Coded in Alphanumeric [5, 8]). Wang & Cohen described in detail an algorithm for constructing document-specific wrappers automatically and experimentally illustrated that character-based wrappers are better suited than HTML-based wrappers for the task of set expansion. Although the main focus of Wang & Cohen was on developing algorithm for named entity extraction, they also presented a method, which utilizes an additional middle context, for constructing wrappers for binary relation instances (i.e., "Ford"\"USA", "Nissan"\"Japan"). The method is language-independent and can be applied to property-value extraction from semi-structured documents. Our work is inspired by the SEAL experimental results. The algorithm for constructing wrappers described in [10] is a pivotal point of our research.

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. 2, the method contains four main tasks: preprocessing, seed learning, pattern discovery, and pattern-based extraction.

The automated discovery of patterns and extraction of the property-value pairs for each product domain is preceded by the initial semi-automated assembling of domain dictionary (aka training set, aka classifier features set) of short properties and values. The semi-automated process is shown in the dashed box in Fig. 2 on the left of the main process. The process uses a small subset of preprocessed product specification pages from a specific domain (e.g., 10 pages) and user-provided seeds to automatically discover patterns and extract property-value pairs. The extracted short property-value pairs during this stage serve as the initial training set for the semi-supervised machine learning algorithm of the main process. The following subsections present the details of the four steps of the proposed technique.

3.1 Preprocessing

To reduce the amount of characters that need to be processed by 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.

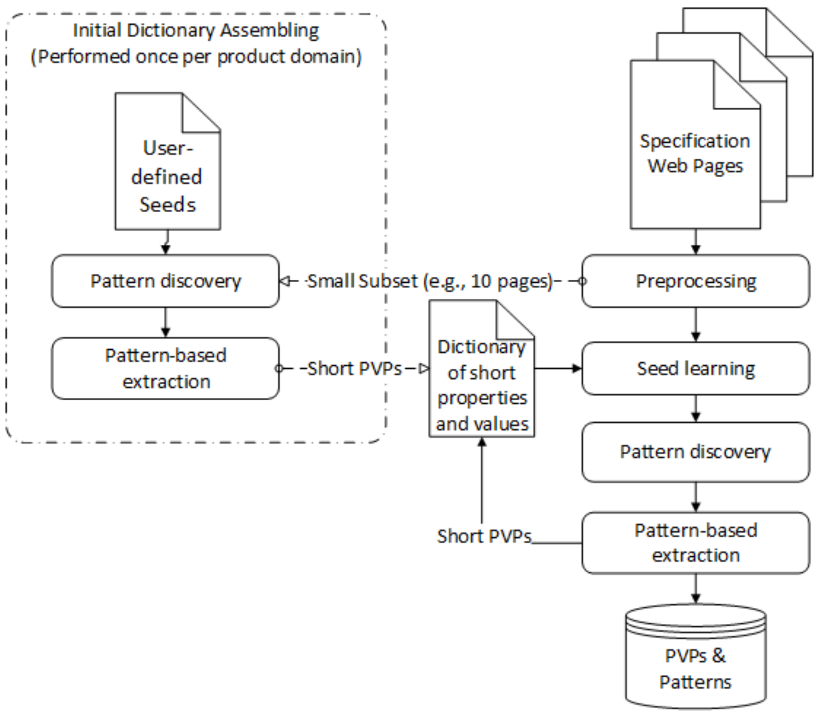


Fig. 2. Semi-supervised property-value pairs extraction algorithm

According to the relative position of a property and value, the existing method assumes that a Web page may meet two conditions:

– 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>);
– the product property name and value belong to the same text chunk and separated by the colon symbol ":" (e.g., <li>Camera resolution: 3.5 megapixels</li>)[1].

During the conversion process, any tags that are on the tag exclusion list (i.e., the tags that certainly do not surround PVPs such as, header, script, etc.) and text chunks between these opening and closing tags are skipped and those that are not, saved to a new document. Before saving the tags and the text chunk to a new document, first, the new line symbols, tabs, and extra white spaces are removed. Next, for the tags with attributes, the unique (i.e., id and name) attribute values are removed and the rest of

[1] A detailed analysis of a vast number of the products' specification Web pages revealed that the colon symbol ":" is the most common property and value separator.

the attributes are stripped of all the symbols, white spaces, and numbers (e.g., <th scopecolclassspecvaluecol>). The text chunks are checked for ":" separator and only the first ":" in the chunk is retained if there are 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

Our ML approach to seed learning employs an inductive semi-supervised learning strategy – self-training model and is based on the observation that there is a limited number of common to any product domain as well as specific to each product domain short property-value pairs (e.g., Weight: 1 kg, Display Resolution: 1024 × 768). These short PVPs are extracted from the training set of the Web pages and used for building classifier's feature set. Because of the limited number of common short PVPs and very little variety of their unique spelling, no typical NLP preprocessing required (i.e., POS tags, stemming, etc.); the classifier does not need a large training set, and the number of features does not grow endlessly during semi-supervised learning.

Another key parts of this approach are the two following assumptions: (1) a property always comes before a value, and (2) there must be a pattern(s) in PVPs formatting. If there is a pattern(s) then only a small number of PVPs are required to discover it (i.e., as few as 2). Therefore, only those learned PVPs that the classifier is the most certain of are selected (i.e., learned PVPs with the highest scores). After classification is done, the text chunk classified as property is selected as a part of PVP seed only if it is directly followed by the text chunk classified as value. The learned PVPs are then used as the seeds for pattern discovery. Using the discovered pattern, new PVPs are extracted, added to the training set and then used in subsequent classification task.

During the seed-learning task, the system uses the dictionary of short property and values which are semi-automatically assembled once for each product domain and used as the training set for classifier in order to learn new short properties and values on the test page. The dictionaries are the property and value Bag-of-Words (BOW) and n-grams (NGRAM) of properties and values. The BOW dictionary contains words found in short properties and values disregarding 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 with removed white spaces between the words (e.g., CAMERARESOLUTION); thus preserving the word order. As the evaluation of our algorithm indicate, the BOW results in higher recall and the NGRAM in higher precision. Therefore, both of them are used in the following fashion: first we attempt to find seeds using NGRAM approach, then, if we do not succeed (i.e., number of found seeds is less than two) and only then, we use the BOW approach.

3.3 Pattern Discovery

To discover patterns, we first find all instances of the provided seed 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.

Next, we then filter out the seed instances that do not have at least one counter-part with the same left, mid, and right context. After filtering the instances, the PVPs are joint together using mid context (e.g., Color</td><td>Black) then for each joint string we extract and insert into two PATRICIA tries left character string (starting from the first character of the document) and right character string (ending at the last character of the document) which is inserted in reversed character order. The tries maintain a list of ids for every node in order to keep track of the seed instances that follow the string associated with that node.

Finally, the algorithm proposed in [10] for constructing unary wrappers is exploited to find maximally long contextual strings that bracket at least one seed instance of every seed. The 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

First, the patterns assembled using the longest character-level prefixes and suffixes found in Step 3 are *normalized* 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, table row tags which surround table cell tags (i.e., < table>, <thead>, <tbody>, <tr>), and list tags (i.e., <ul>, <ol>, <dl>).

Then, the normalized patterns are considered as regular expressions (RegEx) (e.g., <tr><td>(.*?)</td><td>(.*?)</td></tr>) and used for extraction of PVPs. The RegEx 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 <span>, <p>, or <div> HTML blocks, the first property or the last value of extracted PVPs may include extra markup and text. When left-to-right and right-to-left extracted PVPs do not match exactly, the system validates inner HTML of non-matching items and selects the shorter strings that do not contain any markup or longer strings only if they have valid HTML.

When all the PVPs are extracted, the short PVPs (i.e., three-token property and three-token value) are added to the dictionaries and their counts of occurrences are increased. These PVPs are used in the training set for the subsequent classification. Finally, the discovered patterns and extracted PVPs are saved to the database.

4 Evaluation

4.1 Document Corpus and Experimental Design

The evaluation of the proposed approach was performed using a corpus of 108 product specification Web documents (similar to one shown in Fig. 1), and the standard IE performance measures of precision, recall, and F_1 score were measured. The corpus used in the experiments contained documents from two electronic goods' domains – Digital Cameras (56 documents) and Smartphones (52 documents). In order to have unbiased evaluation results, the corpus was collected from 92 distinct electronic goods retailers' Web sites.

The entire corpus was semi-automatically processed, that is PVPs were extracted from each document using few (i.e., two to five) seeds identified on the page and the implemented pattern discovery algorithm, then manually examined, edited and together with missed PVPs (i.e., not extracted by algorithm for some reasons) 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 semi-supervised learning.

The algorithm evaluation 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.

4.2 Pattern Discovery Evaluation

To evaluate the effectiveness of the developed PVPs pattern discovery, the extraction accuracy using known seeds were compared with the manually annotated, expected extractions of the corpus (i.e., page items). To infer the unbiased performance of the algorithm, the precision, recall, and F1 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 expected PVP but contained some additional, irrelevant words or had some words missing). The partial matches occurred in the documents that had very uncommon, unique PVP formatting (e.g., <p><b>(.*?)<br></b>(.*?)</p>).

Table 1. 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.

The first two columns in the Table 1 display total number of all corpus PVPs that were expected and actually extracted respectively. 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 pattern discovery algorithm is a great tool for learning formatting patterns in HTML documents, which delivers impressive performance with precision and recall being around 99%.

4.3 Machine Learning Approach Evaluation

The main goal of ML experiments was to achieve similar recall, precision, and F1 score to the ones attained in the pattern discovery evaluation – close to 99%. To find the best suited classifier for the task, three different text classifiers were integrated into the in-house application and extensively tested.

Classifiers

Microsoft Infer.NET sparse Bayes Point Machine (BPM). For BPM classifier, the initial domain dictionary of short properties and values words (tokens) serves as the set of features and the documents that the dictionary is built on – as the training da-ta. The test data is represented by a vector whose dimensions are equal to the number of features. The BPM uses Bernoulli model of features (i.e., binary occurrence infor-mation) ignoring the number of occurrences. The train\test text string is represented as one raw string similar to the following comma separated values: 1,0,0,0,1,0,0,0,0,0,0,0,0,…, where the numbers mark the presence of a word (i.e., feature) in the train\test text string. The position within the string indicates ordinal position of the feature in the dictionary. This dense data is made sparse by ignoring 0s and is fed to the sparse BPM.

Support Vector Machine (SVM) – the .NET conversion of LIBSVM from Matthew Johnson. The SVM classifier uses the same set of features as BPM. The SVM takes into consideration the number of occurrence of a feature in the training set. The tested text string is represented similar to the following csv: 1:3,32:1,56:45, where the first number is the ordinal of the feature and second is the number of occurrence of the word in the training set.

Naïve Bayes (NB) – a custom implementation of a simple Naïve Bayes multiclass classifier is used. The classifier matches provided test set with available training set. It calculates the score (i.e., prior probability) of every individual token then performs the sorting operation on the probabilities. If the probability of multiple items is the same, this implies that it is an undetermined category; in other words, the classifier cannot decide which category it belongs to because of a lack of information.

Results

In order to determine the best classifier for the task, we compared the performance of BOW and NGRAM feature type for each classifier. The BOTH type works in the following fashion: first the system attempts to find seeds using NGRAM approach, then, if it does not succeed (i.e., number of found seeds is less than 2) and only then, it uses BOW approach. The chart in Fig. 3 shows that the BOW results in higher recall and NGRAM in higher precession, and all classifiers deliver identical results with BOTH feature type setting. The BOTH approach smoothens the difference between BOW and NGRAM precision and recall and delivers highest F1 score – 95.61%. Therefore, since all classifiers with BOTH settings deliver identical precision and re-call, 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.

Thus, NB classifier was selected as the best suited classifier for the task because it takes the least amount of time to process a corpus of over one hundred documents.

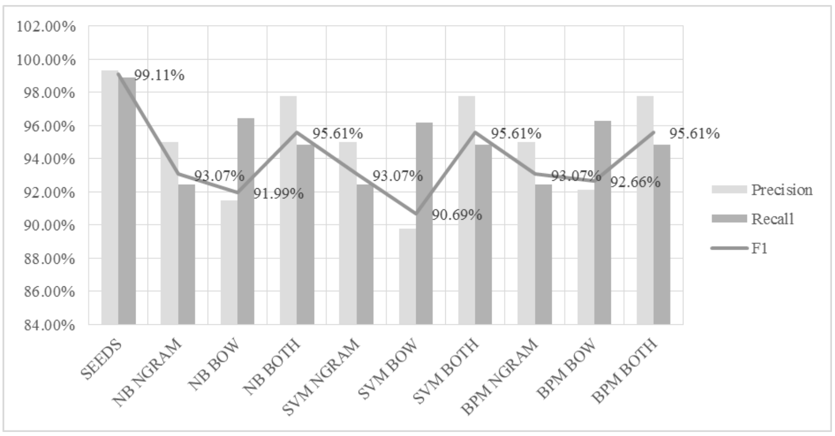


Fig. 3. Feature type evaluation results.

After the classifier was selected we conducted an experiment to determine the corpus processing strategy that will deliver highest precision, recall, and F1. We compared four different strategies with each other as well as with 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 increased by two token limit. The token limit is the maximum number of words of property and value in short PVP that the system uses as the training data. 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 arrangements as in the previous strategy was used but the training set consisted of 20 corpus documents that contained the maximum number of PVPs. The aim was to test if doubling the training data will have positive effect on the processing results.
3. *Two iteration NB BOTH plus an extra iteration with BOW features and increased token limit.* The first iteration was performed with the default settings. In the second iteration, all documents were processed using features collected during the first iteration and only the 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 would 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 though, 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.

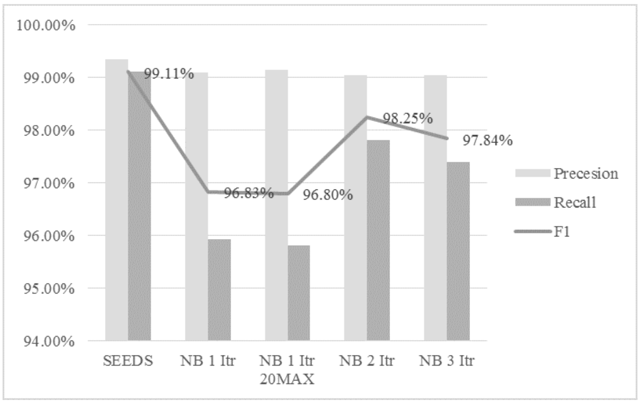


Fig. 4. Corpus processing strategies evaluation.

The chart in Fig. 5 illustrates that the two iterations with BOTH features and an extra iteration with BOW features and increased token limit to process documents that might not have enough 3-token properties and values delivers the best results. The impressive 98.25% F_1 score is very close to the one achieved after processing the corpus utilizing known seeds. Although these results are generalizable to the extent that the two specific domain areas that we have worked with are generalizable, 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 result.

5 Conclusions

In this paper, we 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 are used to discover repetitive HTML formatting patterns and consequently, using these patterns as the wrappers, we 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 ninety diverse, real life electronic goods retailers' Web sites indicate that the algorithm performs extremely well. The corpus of over one hundred documents can be processed on a commodity hardware within a few minutes and delivers as much as 99% precision and 97% recall.

There are several interesting directions for future work. The first direction is to develop a technique for filtering discovered patterns. The technique has to be capable of identifying in some way and discarding patterns that lead to erroneous extractions. The second is to implement some sort of validation of the extracted PVPs and mapping them to the common set of the product properties in order to create cohesive collection of products per each domain. Finally, a property-value extraction system can be assembled based on the prototype system developed during our research.

References

1. Arasu, A., & Garcia-Molina, H. (2003, June). Extracting structured data from Web pages*, The 2003 ACM SIGMOD International Conference on Management of data* (pp. 337-348).
2. 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
3. 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). ACM.
4. 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). Morgan Kaufmann.
5. Gusfield, D. (1997). *Algorithms on strings, trees, and sequences: computer science and computational biology*. Cambridge University Press.
6. Kushmerick, N. (1997). *Wrapper induction for information extraction* (Doctoral dissertation). University of Washington.
7. Kushmerick, N. (2000). Wrapper induction: Efficiency and expressiveness. *Artificial Intelligence*, 118(1-2), 15–68. doi:10.1016/S0004-3702(99)00100-9
8. 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
9. 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. *2012 International Conference on Computer Science and Service System*, (pp. 2021–2025). IEEE.
10. 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). ACM
11. Wang, Y., & Hu, J. (2002, May). A machine learning based approach for table detection on the Web. *11th international conference on World Wide Web* (pp. 242–250). ACM.
12. Wu, B., Cheng, X., Wang, Y., Guo, Y., & Song, L. (2009, September). Simultaneous Product Attribute Name and Value Extraction from Web Pages. *IEEE/WIC/ACM Joint Conference on Web Intelligence and Intelligent Agent Technology* (pp. 295–298), 2009.
13. Zheng, X., Gu, Y., & Li, Y. (2012, April). Data extraction from Web pages based on structural-semantic entropy. In *Proceedings of the 21st international conference companion on World Wide Web* (pp. 93–102). ACM.