# Using Ontologies for Extracting Product Features from Web Pages*

Wolfgang Holzinger, Bernhard Krüpl, and Marcus Herzog

Database and Artificial Intelligence Group, Vienna University of Technology,
Favoritenstraße 9-11, A-1040 Wien, Austria
{holzing, kruepl, herzog}@dbai.tuwien.ac.at

**Abstract.** In this paper, we show how to use ontologies to bootstrap a knowledge acquisition process that extracts product information from tabular data on Web pages. Furthermore, we use logical rules to reason about product specific properties and to derive higher-order knowledge about product features. We will also explain the knowledge acquisition process, covering both ontological and procedural aspects. Finally, we will give an qualitative and quantitative evaluation of our results.

## 1 Introduction

The World Wide Web is an excellent source for product information. Product descriptions are posted on numerous Web sites, be it manufacturer Web sites, review portals, or online shops. However, product presentations on the Web are primarily designed for a human audience. Product features are not encoded in a way that they can be automatically processed by machines. In this paper, we investigate the task of extracting product features, i.e., attribute name-value pairs, from Web pages. The extraction process is assumed to work fully autonomous, given some seed knowledge about a product domain of interest. We will use the digital camera domain to illustrate our approach.

Due to the very nature of the World Wide Web, information about the same product is often spread over a large number of Web sites and is presented in quite different formats. However, technical product information tends to be presented in a more structured way, usually in some form of list or table structure. Still, the presentation variety of this semi-structured information is enormous. In the AllRight project, we strive for distilling knowledge about products and their features from the product descriptions found on large numbers of Web sites. This project is also part of a larger research initiative that deals with various aspects of data extraction from Web pages [2].

We assume that the product descriptions are posted on "regular" Web pages that are not semantically annotated in any way. It is therefore part of our task to annotate to these Web data as much relevant semantics as possible. Semantics

* This research is supported in part by the Austrian Federal Ministry for Transport, Innovation and Technology under the FIT-IT contract FFG 809261 and by the REWERSE Network of Excellence.

I. Cruz et al. (Eds.): ISWC 2006, LNCS 4273, pp. 286–299, 2006.

is always a matter of perspective: when examined from different points of view, completely different properties of a subject matter may become important. For most technical products, though, a common understanding about the relevance of features exists. It is exactly this feature set that manufacturers, dealers, and reviewers list when they post product descriptions on the Web. We exploit this common feature set when we retrieve information about products. For each product, we build a feature space and populate it with instance data extracted from Web pages.

**Related work.** Table extraction from Web documents has been addressed in a number of publications [4,6] and has been used as a basis for the more general notion of knowledge extraction [13,1] from the Web by the way of constructing ontologies from tables [10,11]. In most of these approaches however ontologies are only used for storing the knowledge gathered from the extraction process performed by conventional procedural algorithms.

In [3], the authors describe the use of a domain ontology to create a wrapper for data extraction from Web tables. They use data integration techniques to match the extracted data to the domain description. In the same line, our approach tries to integrate table extraction with table interpretation [6], but uses the classification capabilities of the OWL reasoner Pellet [8] to resolve the gap between tabular presentation and the domain model's semantics, without resorting to external matching algorithms.

The OntoGenie system [9] populates a given ontology with instances using the linguistic WordNet ontology as an interpretative bridge between the unstructured data from the Web and the target ontology. Our system has the same goal of instantiating a domain ontology, but relies on structural information about the data in the form of tabular arrangement and a small domain specific vocabulary.

We believe that migrating parts of the logic to OWL reasoning helps building a modular system with easily exchangeable logical table and domain models. The logic needed to build higher level semantic concepts can be formulated in a natural declarative manner, which helps development and elaboration of these concepts. The procedural components in the process are decoupled from each other, with each component having a clear purpose and responsibility. Usage of OWL as a glue representation between them helps keeping the whole system transparent and eases debugging and evaluation.

**Architecture.** The overall knowledge acquisition process is outlined in Figure 1. In a first step, the table extraction algorithm analyzes an input HTML page for tabular structures, given a specific table ontology $T$. The output of this step is an instantiated table ontology $I(T)$ resembling the information found in a tabular structure on the HTML page. Note that the tabular structure on the HTML page is not necessarily encoded in HTML `table` elements, but only needs to look like a table. This will be explained in more detail later.

In the next step, a content spotter algorithm analyzes the instantiated table ontology $I(T)$ for occurrences of specific domain dependent concepts. The content spotter algorithm utilizes keywords and expressions defined in the domain

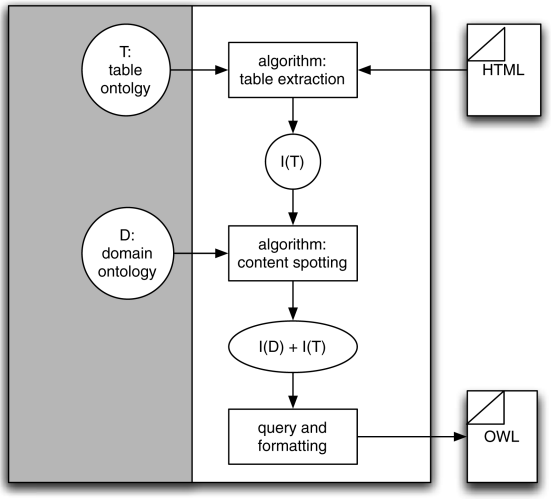



**Fig. 1.** The knowledge acquisition process, covering both ontological and procedural aspects

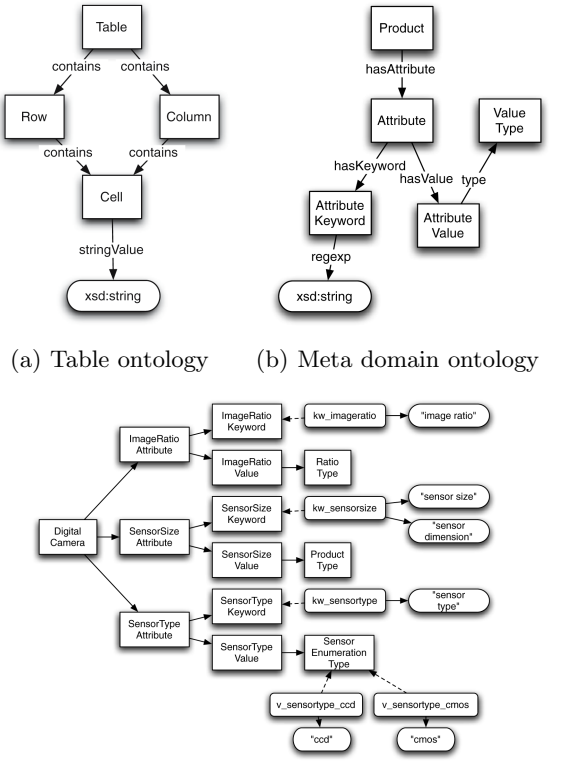
ontology $D$ and annotates the content found in the table structure with the respective concepts. The output of this step is the combination of both the instantiated table ontology $I(T)$ and the instantiated domain ontology $I(D)$, containing enough basic facts to allow for derivation of higher level concepts in $D$.

The final step interprets the instantiated ontologies using a standard OWL reasoner which classifies the instances present in terms of the higher level concepts of the domain ontology. The relevant product information is extracted and stored externally for further processing.

In the following section we will describe how we use OWL ontologies to formalize knowledge about both table structures and product features, as well as how we represent intermediate and final results of the knowledge acquisition process. Section 3 describes in detail the process depicted in Figure 1, covering the table extraction algorithm, the content spotting algorithm, and the derivation of higher-level domain concepts from pure facts extracted from the Web pages. Section 4 presents a quantitative and qualitative evaluation of our process. Finally, in Section 5 we discuss our main findings and discuss further ideas and potential improvements.

## 2 Ontologies

We use two separate ontologies to represent different aspects in our problem domain: knowledge about table structures, i.e., rows and columns, and knowledge about product features, i.e., product attribute keyword–value pairs. We use OWL to represent the ontologies. Moreover, we use the Pellet [8] OWL reasoner to reason about concepts in our ontologies.




(a) Table ontology (b) Meta domain ontology

(c) Digicam ontology

**Fig. 2.** The two base ontologies and a sample domain ontology

## 2.1 Modeling

Figure 2a shows our ontological modeling of tables. The most concise concept is the table cell, represented as concept `Cell`, which contains a textual unformatted string value modeled as an OWL datatype property. Cells are grouped into rows and columns, which is reflected by the `contains`–relation and the concepts `Row` and `Column`. This table model can represent the most basic table type, the rectangular grid table [6] and is adequate in modelling the tables found in the digital camera domain.

The meta domain ontology (Figure 2b) is our basic schema for the description of products. This ontology represents a product as a flat list of attributes, where each attribute is associated with a set of keywords and a typed attribute value. For the description of specific products, we subclass the concepts in the meta domain ontology 2b and fill it with domain specific information, i.e., domain specific attributes. Figure 2c shows an extension of the base ontology for digital cameras (only 3 attributes of the 23 modelled are shown.) For each feature of the camera, we provide an appropriately named attribute. Each attribute is associated

with a matching keyword and value concept. Keyword concepts are basically singleton concepts with only one instance representing the keyword. However, as seen on the instance `kw_sensorsize` in Figure 2c, this single instance can have multiple independent string representations, allowing for various syntactic variants of the keyword. In this way, the domain dictionary is integrated in the domain ontology. Attribute values contain a reference to a type concept which denotes the allowed attribute types. Besides simple numerical types like `RatioType` and `ProductType` that have no further elaborated description in the domain ontology, enumeration types are described in the ontology in a way similar to keywords. For instance, the `SensorTypeValue` of a digital camera can be either `v_sensortype_ccd` or `v_sensortype_cmos`, which are tagged with the respective strings `"ccd"` and `"cmos"`.

### 2.2 Reasoning: The "Containment Forms Context" Assumption

Up to this point, we have two distinct ontologies that are not related to each other. To make a successful interpretation of the content of a table with the semantics defined by the domain model, we have to provide a way to integrate those two ontologies.

Our method is based on the observation that individuals that belong to a certain concept in one ontology, e.g. being a `Cell` in the table ontology, can *at the same time* belong to another unrelated concept in a different ontology, i.e., being the value of a specific attribute in the product ontology.

The crucial point is what we call the "containment forms context" assumption. We use the hierarchical containment relation between texts and cells and rows and columns that is present in the table model to decide on the context that a cell is in. The fact that a cell $c$ belongs to a row $r$ establishes a common context on all the members $c_i$ of $r$.

Consider the attribute value `SensorSizeValue`. We want to classify a cell as a valid sensor size value, iff its text contains both a numerical value and a length unit (like "in" or "cm"). Any cell containing both text fragments binds this two fragments into a common context that we call `SensorSizeValue`:

$$\texttt{SensorSizeValue} == \exists.\texttt{contains}\ \texttt{NumericalType} \cap \exists.\texttt{contains}\ \texttt{LengthUnit}$$

In the same way, we recognize an individual to be an sensor dimension attribute, iff it contains both a sensor dimension attribute keyword and a sensor dimension attribute value. Any individual that `contains` both a `SensorSizeValue` and `SensorSizeValue` should become a `SensorSizeAttribute` also:

$$\begin{aligned}\texttt{SensorSizeAttribute} == {} & \exists.\texttt{contains}\ \texttt{SensorSizeValue} \\ \cap\ & \exists.\texttt{contains}\ \texttt{SensorSizeValue}\end{aligned}$$

At present all definitions of values and attributes follow this simple schema. However, we perceive it to be one of the strong points of our approach that those simple definitions can easily be replaced by more intricate ones if the need arises,

without having to modify any other part of the system, because the handling of these rules is encapsulated in the ontology reasoner.

## 3 The Knowledge Acquisition Process

In the following we will give a detailed description of the knowledge acquisition process as introduced in Section 1. Once started, this process works autonomously until a specified number of product descriptions are harvested from the Web.

### 3.1 Table Extraction

Typical tasks that cannot be handled efficiently by ontological means only are the location and recognition [6] of tabular data regions on Web pages. While table location aims at finding tables in a document, the task of table recognition is to identify the spatial properties of a table. For these tasks we rely on an algorithmic approach that is described in this section. Our approach to table extraction [7] is quite different from previously described ones: we do not operate on the DOM tree or any other incarnation of the HTML source code, but rely on the visual rendition of the Web page. (See [5] for a different variant where they also use positional information of non-text nodes.)

Figure 3 through Figure 5 visualize the process of table extraction, starting from the input HTML page and ending at the output table structure that contains the unlabeled product features. The extraction algorithm detects a product feature table on a Web page, and extracts the spatial features of the table structure along with its content. The result is an explicit representation of the table structure derived from the interpretation of the spatial table features. Note that we do not rely on the structural properties of the HTML source code, e.g. `<table>` elements, to interpret the table structure, but instead utilize visual features that are also accessible to a typical human reader, i.e., word positions and styles. No matter how a table was realized in the HTML source code, whatever looks like a table, i.e., follows certain alignment conditions, will be interpreted as a table.

The table extraction algorithm first groups adjacent words into larger cells (①), thus working in a bottom-up manner starting from the bounding boxes of individual words in a table. Next, the algorithm tries to identify possible table columns (②). We consider a possible table column a set of vertically neighboring cells that are aligned either on the left-hand side, right-hand side, or at the middle. If any cells are found that interrupt the sequence of directly neighboring cells within an identified column candidate, we check if these cells could be intermediate table headings by also testing against an alignment hypothesis (③). Such headings can be important for the further processing because they can give an important context for the cells below the headings.

Once all possible columns were found, the table extraction algorithm tries to identify the column combinations that actually form tables. The strategy looks for adjacent columns that share a common row structure (④): All gaps between

**Fig. 3.** Sample HTML page

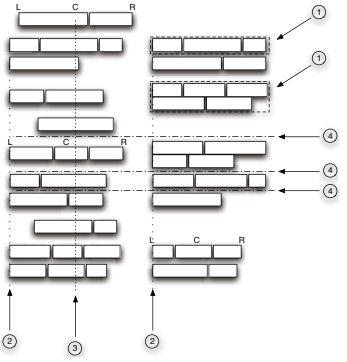



**Fig. 4.** Schema of table extraction algorithm

| | |
|---|---|
| ⋮ | ⋮ |
| Image Ratio | 3:2 |
| Sensor size | 8.4 x 5.2 mm |
| Sensor type | CMOS |
| ISO rating | 100,200,400 |
| ⋮ | ⋮ |

**Fig. 5.** Tabular data structure for product features

the column rows must also be found in the adjacent column. By loosening this requirement to hold only in one direction, we can also allow for rows where a cell in one column corresponds to many cells in another column. We call that procedure *comb alignment* of columns. The strategy also allows for the identification of top, centre, and bottom aligned cells within any table row. Tables that fulfill the comb alignment criterion for columns are returned for further processing.

### 3.2 Applying the Table Ontology

In the next step, we express the structural relationships of the identified tables by means of our table ontology. To this means we translate the spatial properties from the bounding box model into a qualitative model. Consider Figure 5, showing four rows from a typical table describing the features of some digital camera. Applying the table ontology will derive the shown facts about the table, where r1 refers to the first row, c1 to the left-hand side cell in the first row, and c2 to the right-hand cell in the first row. Furthermore, `Row` and `Cell` are concepts of the table ontology, and `contains` and `stringValue` are relations defined in the table ontology. The result is an instance of the table ontology expressing facts about the structural properties of tables produced by analyzing the Web page by means of the table extraction algorithm.

```
r1 a Row.                          c1 a Cell.                          c2 a Cell.
r1 contains c1.                    r1 contains c2.
c1 stringValue "Image Ratio". c2 stringValue "3:2".
```

### 3.3 Content Spotting

Once the structure of a table is represented in the table domain ontology, we turn to the content within the table cells to derive the meaning of the table, i.e., to interpret the table structure in terms of product features represented as attribute

name-value pairs. We utilize *content spotters* for this task. Content spotters are small programs with the purpose to recognize certain semantic concepts in texts. A content spotter is equipped with the necessary knowledge to detect an instance of the concept it represents and, more importantly, to name it and to state the fact in an OWL statement.

**Table 1.** Type spotters detect values with distinctive formatting

| | |
|---|---|
| NumberType | 2,453 |
| ProductType | 8.4 x 5.2 mm |
| TripleProductType | 3.9 x 8.4 x 5.2 in |
| FractionType | 1/400 |
| RatioType | 1:2.8 |

Presently we employ two types of content spotters: *keyword spotters* and *type spotters*. Keyword spotters detect the presence of a particular word or phrase in a number of alternative syntactic representations. Keyword spotters utilize the domain ontology by accessing the regular expressions associated with instances of the various keywords concepts. The keyword spotter remembers the most specific concept for each keyword and will use this concept when it detects the regular expression in a text.

*Type spotters* contain more intrinsic knowledge than keyword spotters. While keyword spotters are only able to detect a limited number of alternative expressions, type spotters are able to detect a whole class of expressions that follow a common schema. Table 1 shows a number of type spotters and the kind of values they typically detect.

Both kinds of content spotters operate by matching a text to a regular expression. Content spotters fetch their regular expressions from the domain ontology. If a substring of the text matches the regular expression of a content spotter, that substring is extracted and annotated with the reference to the annotating spotter.

### 3.4 Applying the Domain Ontology

Once the content spotters have annotated the content within the table cells, the domain ontology can be employed to derive additional facts. Given the table ontology instance as described in Subsection 3.2 and the annotated content, the application of the product ontology can derive the following facts:

```
c1 contains kw_imageratio.    c2 contains v1.    v1 a TypeRatio.
c3 contains kw_sensorsize.    c4 contains v2.    v2 a DoubleProduct.
c4 contains u_mm.
```

Since cell $c2$ contains an individual that is of type `RatioType`, the definition given for the concept `ImageRatioValue` is triggered: cell $c2$ is classified accordingly as an attribute value. Moreover, row $r1$ contains $c1$, which in turn contains

the keyword `kw_imageratio`. Row $r1$ also contains $c2$, which in turn contains a value of the matching type `TypeRatio`. Therefore it is concluded that $r1$ is an `ImageRatioAttribute` according to the product ontology. The following facts are added to the domain knowledge:

```
c2 a ImageRatioValue.
r1 a ImageRatioAttribute.
```

In this way, table rows are successively identified as instances of product attributes.

To conclude, we started from an HTML page, identified the tabular structure containing text fragments, annotated the text fragments with simple semantic concepts according to the domain ontology, and finally derived from those basic building blocks high–level product attributes. The following section gives an evaluation of the quality of both the intermediate and the final derived concepts.

## 4 Evaluation

The automatic, unsupervised identification and extraction of product attributes from Web pages is our ultimate goal. We perform the evaluation of our approach in two steps:

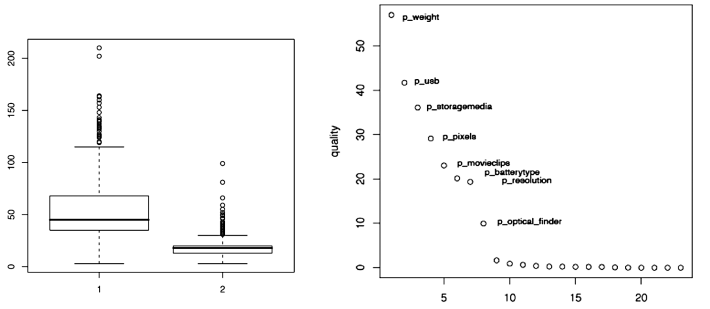
- Firstly, we provide an analysis of the performance of the content spotters that we described in Section 3.3.
- Secondly, we analyze the performance of the whole system by comparing the automatically generated results with manually generated ground truth.

### 4.1 Content Spotter Evaluation

We used the AllRight crawler [2] to automatically locate and retrieve about 6400 Web pages. The crawler searches for pages that, with a high likelihood, contain tables representing the technical specification of digital cameras. The pages originated from manufacturer, dealer and review sites. Due to space constraints, we will not present the AllRight crawler in detail here. It is worth mentioning, though, that the crawling process runs completely unsupervised. The table extraction algorithm we described in Section 3.1 was used to extract 1955 product specification tables from the crawled pages. These 1955 Web pages were then used as candidate pages for the content spotting process.

Figure 6a shows the distribution of the number of rows in the candidate pages along with the distribution of recognized keywords in those tables. The distributions are of similar form, with about 35% of the rows showing an attribute match. The mean value of the identified table rows is 50.4, whereas the mean value for the number of identified attributes is 17.8

Figure 7 shows an overview of how many times each of the keywords matched within the textual content of a table, measuring the ambiguity $a$ of the keyword. There is a clear distinction between keywords appearing with relative high frequency, which also have a tendency to generate outliers — matching extremely




(a) Distributions of table size (1) and matched attributes (2)

(b) Keyword qualities

**Fig. 6.** Attribute distribution and quality

often in a table —, and low frequency keywords that seldom match and never produce excessive multiple matches.

Closely related to Figure 7 is Table 2, showing the number of candidate tables in which each keyword matched at least once, measuring the coverage $c$ of a keyword. Again, the distinction between frequent and infrequent matchers is clearly visible: most keywords either match on more than 80% of the tables, or they match in less than 5%. The keyword `p_weight` matched in 94% of all tables, and produced on average 1.5 matches per table. It is our prime example for a perfect attribute: matching in almost every case, and matching with minimum ambiguity. We strive for high coverage of a keyword to be of maximum use in every case, and we need low ambiguity of the keyword to achieve precise classification.

We measured keyword quality using the simple formula $q = \frac{c}{a}$, where quality is proportional to coverage and inversely proportional to ambiguity. Figure 6b, displaying keyword qualities in descending order, shows that keyword quality

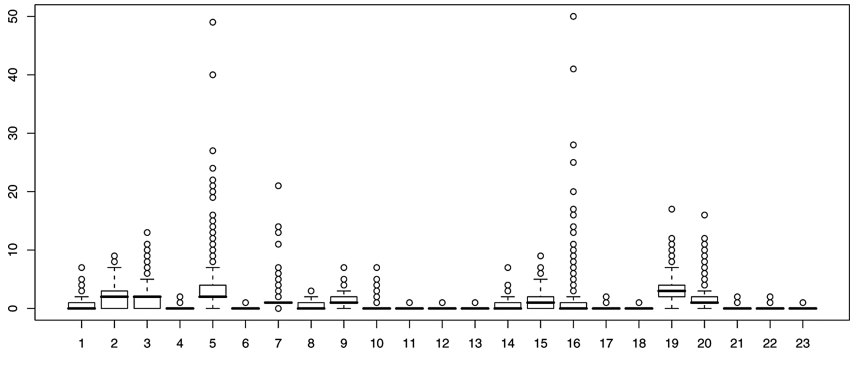


**Fig. 7.** Keyword ambiguity (see table 2 for keyword names)

decreases exponentially. Therefore, the top 1/3 keywords are responsible for most of the semantic annotation, while the remaining keywords are almost useless. This is a hint that those attributes were either underspecified in the domain knowledge, or that these attributes are really so infrequently mentioned to make them negligible. In any case, they should be used with caution in subsequent evaluations of the generated data.

**Table 2.** Keyword coverage

| | | | | | |
|---|---|---|---|---|---|
| 1 | p_rechargeable | 649 | 2 | p_resolution | 1380 |
| 3 | p_batterytype | 1465 | 4 | p_denomination | 242 |
| 5 | p_pixels | 1669 | 6 | p_display_size | 4 |
| 7 | p_movieclips | 1620 | 8 | p_firewire | 688 |
| 9 | p_weight | 1850 | 10 | p_guarantee | 140 |
| 11 | p_vendor | 11 | 12 | p_internal_memory | 11 |
| 13 | p_lcd_display | 3 | 14 | p_brand | 895 |
| 15 | p_optical_finder | 1432 | 16 | p_price | 866 |
| 17 | p_productdescription | 27 | 18 | p_slrcamera | 5 |
| 19 | p_storagemedia | 1805 | 20 | p_usb | 1854 |
| 21 | p_videoout | 7 | 22 | p_zoomfactor_digital | 7 |
| 23 | p_zoomfactor_optical | 6 | | | |

### 4.2 Evaluation Against Ground Truth

As explained in a previous section, the extraction stage is fed by a retrieval component that automatically retrieves domain relevant pages containing semi-structured data. We randomly selected 30 of these retrieved pages for manual annotation by a human domain expert. To make the annotation process less time consuming and error prone, we devised a ground truth annotation tool that we use to annotate relevant Web pages. We do not try to annotate all of the information on a page, but cover only a fraction of it by selecting a set of 5 attributes. We assume that the extraction quality for the other attributes will be comparable.

We need to provide ground truth to be able to verify results at different stages in our process: the location of tables on a page, the recognition of the table, and the interpretation of the function of table cells. Therefore, ground truth has to provide information about: left top and right bottom corners of the table, which word tokens in the table form a table cell, and the functional relations of table cells.

Several table models have been proposed in the literature [4,6,12]. We restrict our analysis to those table types that only contain a single level of table nesting, i.e., the nesting that is defined by intermediate headings. In addition to reducing the complexity of the problem, we can give more arguments for this restriction: Our system is not an isolated experiment in the table extraction field, but has to link the table interpretation results into our domain ontology. This ontology is centered around the concepts of products and attributes. If there are more

complex structural relationships contained in a particular table, it is very likely that these relationships, or the table as a whole, just are not appropriate for our extraction task.

The notion of subjectivity is an important factor in our considerations. When we want to extract product information from tables, we want it to be aligned with our (subjective) domain ontology, therefore we need to find those tables on the Web that share the same conceptualization basics. If an author describes a product from a completely unique perspective, this document cannot be included in our analysis, even when it is semi-structured. This is due to the fact that a common ontological understanding is missing in this case.

The ground truth generation tool we devised lets the user operate on the visual rendition of a Web page. We implemented an extension for the Mozilla Firefox browser that can be invoked for any Web page displayed in the browser. If activated, the user will be able to select any word on the page by pointing the mouse cursor over it; the selected word can then be annotated as a certain attribute keyword or value by performing some key strokes. In addition, there is a mode to indicate which of the word tokens belong together to form a functional unit. The results of the annotation process are stored in instances of an OWL ontology that allows for an easy comparison with the automatically generated results of our system.

**Table 3.** Results of evaluation against ground truth

| Average number of (per document) **ground truth** | | | Average number (per document) **identified** | | |
|---|---|---|---|---|---|
| keywords | values | attributes | keywords | values | attributes |
| 5.13 | 6.78 | 4.82 | 3.22 | 4.02 | 3.38 |

| **Recall** | | | **Precision** | | |
|---|---|---|---|---|---|
| keywords | values | attributes | keywords | values | attributes |
| 62.8% | 59.3% | 70.1% | 65.8% | 41.6% | 92.4% |

| **F-measure** | | |
|---|---|---|
| keywords | values | attributes |
| 64.27% | 48.90% | 79.72% |

### 4.3 Results

We asked different users to annotate 30 documents with 5 concepts from our domain ontology. We quickly found out that the annotation heavily depends on qualification of the user in the domain: Identifying CCD sensor sizes in documents is very difficult for users who do not have an appropriate background. On the other hand, even within the group of domain experts, there were differences in what users considered being related to a domain concept or not (e.g. in the case of sensor resolutions). For us, this proves our assumption that subjectivity plays a key role in the extraction process that has been underestimated so far.

Table 3 summarizes the results. The section "average number of ground truth" gives the average numbers for annotated `keywords`, `values` and `attributes` within a document. `Keyword` and `value` denote the respective parts of a keyword-value pair. Together, these elements form an `attribute`, i.e., a product feature. Note that the total number of keywords, values, and attributes is not equal. This is due to the fact that multiple values can exist for a single keyword, and attributes must comprise both a keyword and a value. The "average number identified" section gives the average numbers for the automatically identified attributes over all documents examined. For both recall and precision we get significantly better values for attributes than for keywords and values alone, showing the benefit of the effort to derive higher level domain concepts.

## 5 Conclusions and Outlook

We presented a system that uses ontology reasoning to integrate table extraction and table interpretation. A first evaluation has shown that the classification work done by the reasoner can significantly increase precision and recall of high level semantic product information.

Presently, the content spotters use regular expressions to match keywords and types. Experience has shown that tables frequently contain phrases that are not easily recognizable by regular expressions. Recognizing only simple phrases and assembling them with complex ontology concepts is computationally expensive. Simple grammars with a limited capability of recognizing natural language phrases could be used in place of the regular expressions.

Our current rectangular table model, while capturing the essential information, does not make use of the additional structural information that is present in more complex layouts. For example, many tables in our testing set used column spanning rows as sub–headers to segment a long table. The information in these sub–headers can give valuable context information to the interpretation of the row–attributes. We are currently working on an extended table model that can represent segmented nested tables. Such a model requires generalizing the containment–context axiom to multiple levels.

## References

1. Harith Alani, Sanghee Kim, David E. Millard, Mark J. Weal, Wendy Hall, Paul H. Lewis, and Nigel R. Shadbolt. "Automatic Ontology-Based Knowledge Extraction from Web Documents" In *IEEE Intelligent Systems, Vol. 18, No. 1, pages 14–21*, 2003.
2. Julien Carme, Michal Ceresna, Oliver Frölich, Georg Gottlob, Tamir Hassan, Marcus Herzog, Wolfgang Holzinger, and Bernhard Krüpl. "The Lixto Project: Exploring New Frontiers of Web Data Extraction" In *Proc. of the 23rd British National Conf. on Databases*, 2006.
3. David W. Embley, Cui Tao, Stephen W. Liddle. "Automatically Extracting Ontologically Specified Data from HTML Tables of Unknown Structure" In *Proc. of the 21st Int. Conf. on Conceptual Modeling (ER02), Tampere, Finland*, 2002.

4. David W. Embley, Daniel Lopresti, and George Nagy. "Notes on Contemporary Table Recognition" In *Proc. of the 2nd IEEE Int. Conf. on Document Image Analysis for Libraries*, 2006.
5. Wolfgang Gatterbauer and Paul Bohunsky. Table Extraction Using Spatial Reasoning on the CSS2 Visual Box Model In *Proc. of the 21st National Conf. on Artificial Intelligence*, 2006.
6. Matthew Hurst. "Layout and Language: Challenges for Table Understanding on the Web" In *Proc. of the 1st Int. Workshop on Web Document Analysis*, 2001.
7. Bernhard Krüpl and Marcus Herzog. Visually Guided Bottom-Up Table Detection and Segmentation in Web Documents. In *Proc. of the 15th Int. World Wide Web Conf.*, 2006.
8. Bijan Parsia, Evren Sivrin, Mike Grove, and Ron Alford. Pellet OWL Reasoner, 2003. Maryland Information and Networks Dynamics Lab *http://www.mindswap.org/2003/pellet/* (as of May 2006).
9. Chintan Patel, Kaustubh Supekar, and Yugyung Lee. "Ontogenie: Extracting Ontology Instances from WWW" In *Human Language Technology for the Semantic Web and Web Services*, ISWC'03, Sanibel Island, Florida, 2003.
10. Masahiro Tanaka and Toru Ishida. "Ontology Extraction from Tables on the Web" In *Proc. of the Int. Symposium on Applications on Internet*, 2006.
11. Yuri A. Tijerino, David W. Embley, Deryle W. Lonsdale, and George Nagy. "Ontology Generation from Tables" In *Proc. of the Fourth Int. Conf. on Web Information Systems Engineering*, 2003.
12. Xinxin Wang. "Tabular Abstraction, Editing, and Formatting" *PhD thesis, Univ. of Waterloo*, 1996.
13. Alan Wessman, Stephen W. Liddle, and David W. Embley. "A Generalized Framework for an Ontology-Based Data-Extraction System" In *Proc. of the 4th Int. Conf. on Information Systems Technology and its Applications, pages 239–253*, 2005.