ALLRIGHT: Automatic Ontology Instantiation from Tabular Web Documents

Kostyantyn Shchekotykhin, Dietmar Jannach, Gerhard Friedrich, and Olga Kozeruk

Universität Klagenfurt, Universitätsstrasse 65, 9020 Klagenfurt, Austria, Europe
{kostya,dietmar,gerhard,olga}@ifit.uni-klu.ac.at

Abstract. The process of instantiating an ontology with high-quality and up-to-date instance information manually is both time consuming and prone to error. *Automatic ontology instantiation from Web sources* is one of the possible solutions to this problem and aims at the computer supported population of an ontology through the exploitation of (redundant) information available on the Web.
In this paper we present ALLRIGHT, a comprehensive ontology instantiating system. In particular, the techniques implemented in ALLRIGHT are designed for application scenarios, in which the desired instance information is given in the form of *tables* and for which existing Information Extraction (IE) approaches based on statistical or natural language processing methods are not directly applicable.
Within ALLRIGHT, we have therefore developed new techniques for dealing with tabular instance data and combined these techniques with existing methods. The system supports all necessary steps for ontology instantiation, i.e. web crawling, name extraction, document clustering as well as fact extraction and validation. ALLRIGHT has been successfully evaluated in the popular domains of digital cameras and notebooks leading to a about eighty percent accuracy of the extracted facts given only a very limited amount of seed knowledge.

1 Introduction

The information gathering process for specialized *knowledge-based services* [1] can be a time consuming and error prone task. In order to at least partially automate this task, several *Web Mining* and *Information Extraction* techniques have been proposed over the last years as one cannot yet expect that Web documents are already sufficiently semantically annotated. Some of them, for instance, rely on the redundancy of information on the Web and use statistical methods [7, 4, 3, 5], others use natural language processing (NLP) techniques [19, 20] to extract the required knowledge from unstructured documents. It has also been shown that domain *ontologies* can be useful to support knowledge extraction, which in turn means that the information extraction problem can be generalized to the problem of finding and inserting information into the system's knowledge base that matches a given ontology [1], a process also referred to as *ontology instantiation* or *ontology population*.

The herein described ALLRIGHT system was designed based on the requirements that came up in the development of such a real-world knowledge-based service. In particular, the problem was to set up and maintain an up-to-date knowledge base of detailed

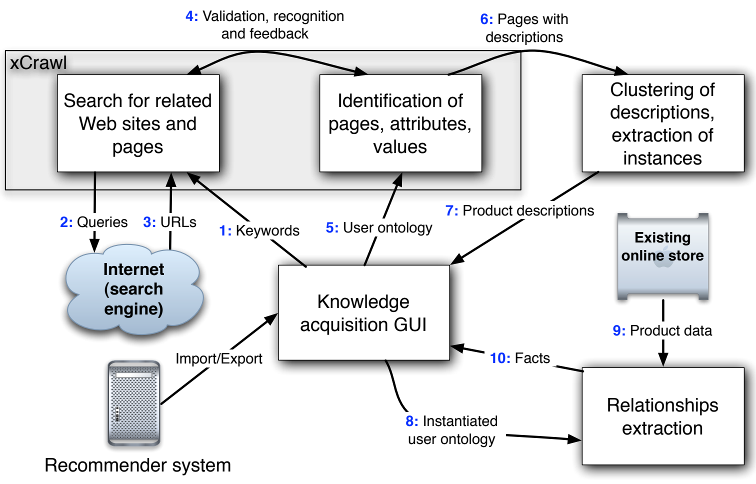

4: Validation, recognition and feedback
6: Pages with descriptions
xCrawl
Search for related Web sites and pages
Identification of pages, attributes, values
Clustering of descriptions, extraction of instances
2: Queries
3: URLs
1: Keywords
5: User ontology
7: Product descriptions
Existing online store
Internet (search engine)
Knowledge acquisition GUI
9: Product data
Import/Export
10: Facts
Relationships extraction
8: Instantiated user ontology
Recommender system

Fig. 1. ALLRIGHT ontology instantiation process in application scenario

item descriptions in an online store. These in-depth product descriptions should subsequently serve as a basis for building content-based recommender and product comparison applications as described in [9]. One can for example easily imagine that maintaining data sets for more than 400 digital camera models by hand and updating the knowledge base whenever new products are available is time consuming and costly. Our experiences also showed that the quality and level of detail of commercially available data sets is not sufficient for our purposes.

Knowing that in many domains the required item information is already available on the Web (e.g., on manufacturer homepages or community portals), the goal was thus to develop a Web Mining System (WMS) that is capable of finding and extracting this knowledge automatically. What was soon recognized, however, is that existing Web Mining techniques like the ones mentioned above are not directly applicable, mostly because the instance information is in many cases described in *tabular form*. Note that these tables are not necessarily constructed from underlying databases which means that neither SPARQL nor “Hidden-Web” techniques can be used.

In general, tables are well-suited for the human reader as they are a compact and precise form of representing information. The problem with tables in the context of existing information extraction techniques, however, is that they contain nearly no information that can be exploited by NLP methods. In addition, with respect to clustering methods, the problem is that pages with *the same sets of keywords/tokens* (from the same portal) describe *different items*. In the ALLRIGHT system these particularities have consequently been taken into account and new techniques for dealing with tabular information have been developed.

Nonetheless, ALLRIGHT was designed as a more general, domain independent ontology instantiation system in the sense of [1]: The input to the extraction system is thus a domain ontology that describes the structure of and the relations between the items to be searched for. The extraction process itself is structured in a series of tasks like Web crawling, name extraction, validation, document clustering, and fact extraction.

We see the contributions of ALLRIGHT to the state of the art as follows. First, we show how the approach of [1] to automatic ontology instantiation can also be applied to domains in which data is given in tables, which is a is common representation form for all kinds of data on the Web like personal information, geographical data and so forth. In addition, our usage of ontologies is broader than in previous approaches, as we also apply model-based diagnosis techniques [10] to automatically identify conflicts between the domain ontology and the extracted knowledge and are thus able to generate corresponding resolution proposals. Our contributions also include a new crawling method for fast location of tabular descriptions, a visual table identification technique, and a novel way of applying clustering algorithms to deal with tabular descriptions. Finally, given the promising evaluation results with an accuracy of about eighty percent, we show how Semantic Web technology, ontologies, and a combination of new and existing extraction techniques can help us to better deal with real-world, industrial problem settings.

In the subsequent sections of the paper we will - based on a continuing example problem from the domain of digital cameras - sketch the overall process flow, give technical details of the individual extraction steps, and report evaluation results for the individual subtasks.

2 Automatic Ontology Instantiation

Fig. 1 sketches the overall ontology instantiation process in one possible example setting, in which detailed product data should be extracted for a content-based recommender system in an online store. First, a domain ontology is developed, which describes the basic structure and characteristics of the items to be searched for. In our example configuration, some parts of the knowledge are imported from the ADVISOR SUITE system [8], in which for instance the list of characteristics for an item to be recommended can be defined. From the domain ontology (more details will be given in subsequent sections), the system generates keywords (1) that are used for crawling the Web for relevant Web pages (2,3) XCRAWL. Next, the downloaded pages are analyzed by the Identification Component (4) in order to determine whether they *really* contain item descriptions. Again, this analysis is done on the basis of the knowledge from the ontology (5). The filtered set of documents is then forwarded (6) to a module which first clusters those pages that describe the same products, then extracts the specific facts for each product and feeds the new knowledge back to the ontology (7). Note that in principle, also an alternative strategy for this step is possible, i.e., first perform fact extraction and then try to remove the duplicates [16]. Still such systems require more detailed seed knowledge and specialized and complex fact recognition techniques. In all phases, feedback and validation loops are contained which are used to refine the knowledge acquired so far. Within steps 8,9, and 10 relationships between the successfully identified ontology instances and other shop items are instantiated. In the digital camera domain, this could for instance be the problem of finding out which accessories are compatible with a certain camera model. However, these steps are only covered superficially in this paper as we want to focus on the extraction of product descriptions.

2.1 Ontologies in ALLRIGHT

In the ALLRIGHT system, three "levels" of ontologies are of interest, see Fig. 2 for example fragments. In the product data extraction scenario, the predefined *core ontology* for instance describes that we search for entities that have attributes of given types and certain units of measurement. Both the attributes and the units can be annotated with string-typed keywords.

When ALLRIGHT is to be configured for a certain domain, i.e., digital cameras in our scenario, a *domain ontology* has to be derived from the core ontology. In the example, we thus state that "resolution" is a property of interest for entities of type "digital camera". The resolution is given as a real-valued number and is measured in mega pixels. Appropriate keywords could be "effective pixels" and "million", respectively. Note that the task of defining the domain ontology corresponds to the definition of *seed knowledge*, which is required in every Web Mining System. Technically, the domain ontology is stored as a Web Ontology Language (OWL) document. The (graphical) definition and maintenance of the model can thus be accomplished with the help of any general purpose ontology editing tool. In our application scenario, however, we used the built-in graphical knowledge acquisition tools from ADVISOR SUITE [8].

The right-most part of Fig. 2 shows how one specific ontology *instance* in the domain could look like. In contrast to the definition of the domain ontology, which is done manually, the extraction of all corresponding instances that match the domain ontology is the automated task of the ALLRIGHT system. We will subsequently show how the domain ontology is exploited within the different subtasks and summarize the other advantages of using ontologies in that context.

2.2 Ontology-based data collection

Once the domain ontology is defined, the next task of the system is to locate as many Web documents as possible that contain information which matches the ontology (Steps 1 to 5 in Fig. 1). The resulting set of pages, which may contain multiple documents per instance, are then forward to the extraction component (Step 6).

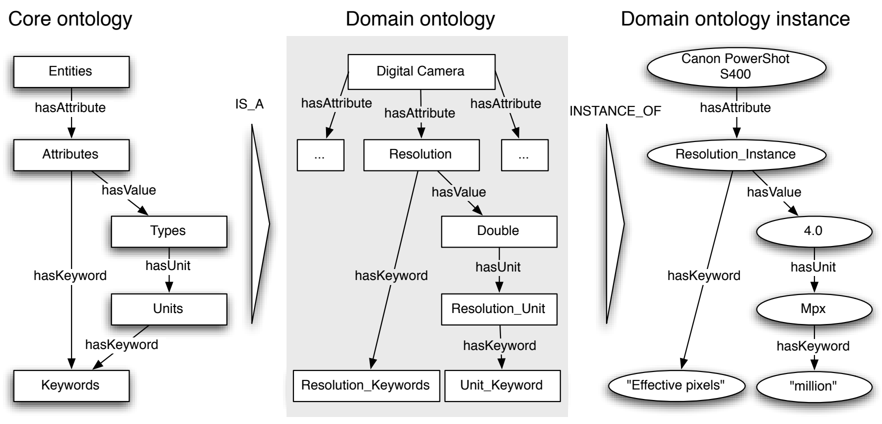


Fig. 2. Fragments of ontologies used in the ALLRIGHT system.

```
function FIND-AND-EXTRACT returns a collection of extracted tables
  inputs: domainOntology, an ontology derived by a user from the core ontology.
          maxIterations, (number) a convergency threshold of the crawler.
  local variables (initially empty):
                    Pages, a collection of pages downloaded by the crawler.
                    Hubs, a collection of pages that are linked with many authorities.
                    Authorities, a collection of pages with tabular descriptions of domain
                        ontology instances {
  Authorities := GETSEEDDATA( domainOntology )
  Sites := GETWEBSITES(Authorities);
  for each ( $site \in Sites$ ) {
    do {
      Authority := SELECTRANDOM( Authorities, site )
      newPages := PERFORMRANDOWWALK( Authority, Pages )
      Pages := Pages $\cup$ newPages
      Authorities := Authorities $\cup$ ANALYZEPAGES( domainOntology, newPages )
      Hubs := Hubs $\cup$ EXTRACT( Pages, Authorities )
    } while (NONEWHUBSFOUND( maxIterations ))
  }
  for each ( $hub \in Hubs$ ) {
    Authorities := Authorities $\cup$ GETALLAUTHORITIES( domainOntology, hub )
  }
  return STOREDOCUMENTS( Authorities )
}
```

Fig. 3. ALLRIGHT data collection algorithm

The ALLRIGHT system implements the subsequently described specific data collection algorithm (Fig. 3), which represents a novel combination of new and existing techniques in the area.

GETSEEDDATA: In the first step, a *seed* collection of relevant documents and web sites is created with the help of an *Automatic Query Generation* (AQG) technique (cf. [12]). Note that in general, there exist two commonly used approaches to automated data collection: A Focused Crawler efficiently seeks out documents about a specific topic and is guided by the link structure and the content of the Web [2]. In opposite to crawlers used by search engines that perform breadth-first search, a focused crawler employs depth-first search that is usually guided by some scoring algorithm. The other approach, Automatic Query Generation, retrieves documents by sending sophisticated queries to search engines and analyzes the retrieved documents with the help of domain-specific classifiers. The outcomes of this analysis process are then used to refine the queries in order to improve the results.

Note that we use AQG in the first phase because it helps us to retrieve the set of interesting Web documents and *sites* quickly (see [12] for a comparison of data collection strategies). Later on, however, we will use a focused crawling technique in combination with the "index-page heuristic" to achieve high recall: The basic assumption of the heuristic is that Web site creators typically arrange the documents of a site in a way that they can be easily accessed by users through index pages or hierarchies.

For the query generation task, our crawler produces an ordered list of all attribute keywords from the domain ontology whose order is based on the number of hits returned for search engine queries that consist only of the keyword itself. This ordering helps us to distinguish general keywords from very specific ones. Next, the AQG system generates further queries to a search engine, where each query is a combination of keywords with the maximum length of m query terms. The generation algorithm starts from the middle of the ordered list and proceeds in both directions simultaneously. Each generated query is sent to a search engine and the first n URLs are saved. In our experiments we used $m = 10$, because most of the search engines accept no more than 10 terms in one query, and $n = 30$, because 30 descriptions per instance is a statistically good sample size.

Also within the GETSEEDDATA method, a first *validation* phase (identification step in Fig. 1) takes place, i.e., the goal now is to check whether a downloaded page actually contains a tabular description of an ontology instance. The *validation process* begins with the analysis of individual Web pages with "content spotters" which is similar to systems presented in [13, 6]. Content spotters are used to identify and mark up interesting text positions for later processing (extraction). To some extent they also mimic human behavior when reading a product description: A document is first roughly skimmed for areas of interest detecting keywords or familiar text constructs like attribute values (numbers) that are are followed by units (keyword strings). Once such "context frame" is detected, the reader will take a closer look at these contents or page fragments.

In the ALLRIGHT system, content spotters are automatically constructed from the core ontology and the domain ontology, each one created to find a special keyword or structural feature. For example one spotter could look for definitions of number ranges (e.g. 10 to 12 cm) while another one seeks string enumerations and a third looks for numbers accompanied by units. Once a spotter finds an interesting region it will continue to examine the region in more detail and may for instance check whether the found number is within the allowed range as defined in the ontology. The content-spotting task itself is subdivided into several phases: First, simple spotters are applied on the document. Then the results are merged and potential conflicts resolved before more advanced content spotters operate on the matches found in the previous phase. The final output of the content spotters is an annotated text document in which interesting positions are marked and classified.

Based on these annotated documents, the ALLRIGHT system then uses a new *visual table recognition* method [15] to check whether the document contains tabular information which can be extracted. The method retrieves the coordinates of text boxes from a browser integrated into the crawling component and tries to draw lines that separate the text boxes (words) of a rendered page by using a system of heuristics of possible text alignments. Then the method analyzes those parts of the documents which were annotated by the content spotters and decides where the target table is placed on the page. The most plausible area is extracted for further processing and the page will be added to the set of *Authorities*, i.e., interesting pages.

GETWEBSITES: This small function takes the identified web pages as an input and extracts the basic URLs of the corresponding web sites like e.g. a manufacturer home page. These URLs serve as a starting point for the subsequent focused crawling phase.

FOR EACH SITE ∈ SITES: The main goal of this block is to identify *hubs* like index pages on each web site (as it is defined in [14]). This is done in an iterative process that continues until no new hubs are found over a predefined number of iterations. Our experiments showed that using an iteration limit of around 50 was appropriate in our setting. Lower values led to insufficient crawler performance and greater ones did not improved the crawler's result considerably and required much longer running times.

The crawler selects randomly one of the authorities (SELECTRANDOM) at the beginning of each iteration. This authority is used as a starting point for a *random walk with restart* [17] strategy to explore the Web site graph until the the first restart event happens. Note, that this strategy has the advantage that the search process does not get stuck in non-interesting or well-explored branches of the graph. The newly found pages are added to the local set of all *Pages* found so far in order to prevent the crawler from downloading the same page twice.

Next, ANALYZEPAGE searches for new authorities within the newly identified pages by using the same identification and validation techniques as described in the GETSEEDDATA function, i.e., content spotters and table recognition.

Finally, we analyze all pages found so far (EXTRACT) by applying the HITS algorithm [14] in order to detect new hubs for already known authorities.

FOR EACH HUB ∈ HUBS: Given all the hubs from the previous step, the algorithm now downloads all the pages that are referenced by each hub. All downloaded pages are then analyzed (GETALLAUTHORITIES) and if new authorities are found (again based on validation and identification), they are stored in the corresponding list.

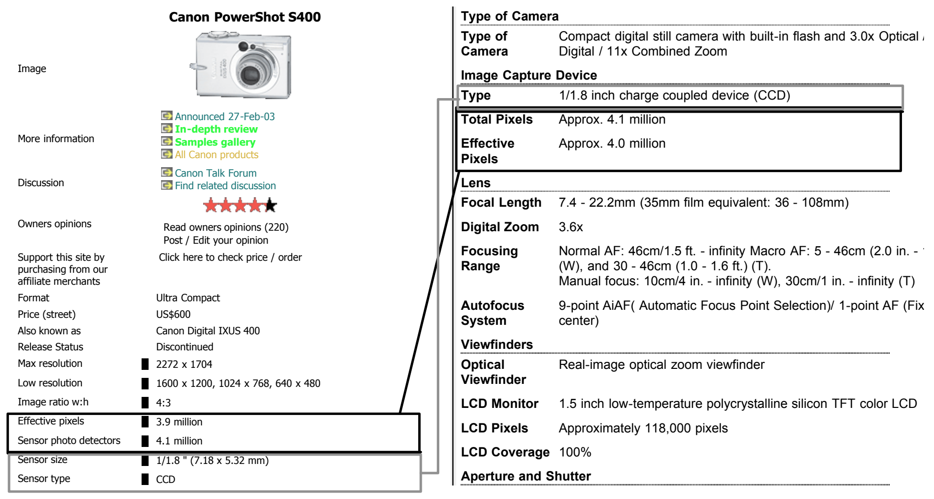
STOREDOCUMENTS: Finally, the full list of extracted Web pages is stored for the subsequent document clustering and fact extraction steps.

2.3 Document clustering & instance name recognition

Most of the latest statistics-based WMS like KnowItAll [7], Web→KB [4], C-Pankow [3], SemTag [5], and other systems exploit the redundancy of information in the Web to acquire knowledge. These systems apply different comparison and classification methods to analyze document collections that contain redundant descriptions of a target class and/or their instances: An instance description of a class, like for example a specification of a digital camera model, can be published on different Web sites. Our goal is thus to determine in the next step which pages describe the *same* instance in order to extract as much information as possible from different sources.

The main problem in this context however is that pages describing the same camera can significantly differ from each other in three main dimensions, see Fig. 4:

- *Presentation.* The left table is not structured like the original manufacturer's one, which contains *sections* like "Image capture device" or "Lens". Moreover, some distinct attributes of one table can, in fact, be described in on single attribute in the other one. For example, the "Type" attribute in the right table covers the information of the "Sensor type"and "Sensor size" attributes of the left table.
- *Detail coverage.* Review sites, like dpreview.com, often publish attributes of products that are not mentioned by manufacturers, but may be important for customers. Canon.com for instance does not report the capacity of the memory card shipped



Canon PowerShot S400

Image	
More information	Announced 27-Feb-03 In-depth review Samples gallery All Canon products
Discussion	Canon Talk Forum Find related discussion
Owners opinions	★★★★★ Read owners opinions (220) Post / Edit your opinion
Support this site by purchasing from our affiliate merchants	Click here to check price / order
Format	Ultra Compact
Price (street)	US$600
Also known as	Canon Digital IXUS 400
Release Status	Discontinued
Max resolution	2272 x 1704
Low resolution	1600 x 1200, 1024 x 768, 640 x 480
Image ratio w:h	4:3
Effective pixels	3.9 million
Sensor photo detectors	4.1 million
Sensor size	1/1.8 " (7.18 x 5.32 mm)
Sensor type	CCD

Type of Camera	
Type of Camera	Compact digital still camera with built-in flash and 3.0x Optical / Digital / 11x Combined Zoom
Image Capture Device	
Type	1/1.8 inch charge coupled device (CCD)
Total Pixels	Approx. 4.1 million
Effective Pixels	Approx. 4.0 million
Lens	
Focal Length	7.4 - 22.2mm (35mm film equivalent: 36 - 108mm)
Digital Zoom	3.6x
Focusing Range	Normal AF: 46cm/1.5 ft. - infinity Macro AF: 5 - 46cm (2.0 in. - (W), and 30 - 46cm (1.0 - 1.6 ft.) (T). Manual focus: 10cm/4 in. - infinity (W), 30cm/1 in. - infinity (T)
Autofocus System	9-point AiAF(Automatic Focus Point Selection)/ 1-point AF (Fix center)
Viewfinders	
Optical Viewfinder	Real-image optical zoom viewfinder
LCD Monitor	1.5 inch low-temperature polycrystalline silicon TFT color LCD
LCD Pixels	Approximately 118,000 pixels
LCD Coverage	100%
Aperture and Shutter	

Fig. 4. Different representation of information for the same camera

with a camera, which, however, can be found on dpreview.com in the "Storage included" attribute.

– *Content.* Attributes that are named differently can identify the same property of a camera, like "Total pixels" in the right table corresponds to "Sensor photo detectors" in the left one. In addition, there can be also a mismatch between the values of attributes that describe the same property of a camera. In Fig. 4, the values of the "Effective pixels" attribute are actually different.

Overall, the existing approaches to document clustering are designed to work with general sources written in natural language, i.e., they analyze documents or their parts "as is". Web→KB [4] for example uses a combination of three Naive Bayes classifiers trained to group the available documents based on full text, headers/titles and hyperlinks (anchor tags). Another technique is implemented in the KnowItAll [7] system, which uses web statistics acquired from search engines in order to determine the plausibility of extracted facts.

If, however, instances are described in tabular form as in our case, the applicability of these clustering methods is limited, because tables that describe *different* instances typically contain very common sets of words (like attribute names) and almost no grammar. On the other hand, it may also be possible that there are two documents that describe the same instance, but use non-overlapping sets of tokens, which means that standard clustering techniques would put them into different groups.

Therefore, in the ALLRIGHT system, we aim to cluster the documents indirectly by means of some previously extracted unambiguous instance identifiers like an *instance name*. These extracted names can consequently also be used to retrieve additional documents from the Web, as to increase redundancy and to improve the results of a statistics-based Web mining process. Unfortunately, such names or other identifiers are usually not explicitly given and there is no general rule where such identifiers typically appear on a Web page. Therefore, in our approach, we exploit the information contained in the *title* tag of the Web page that describes the instance as well as *hyperlinks* that point to that page. Of course, depending on the domain, also other tags can be considered even if they contain additional and irrelevant information.

Our current approach, which is more general in comparison to a previously reported implementation, see [11], uses the *X-Means* clustering algorithm [18] in order to extract the most significant part (name candidate) of strings contained in the title and hyperlinks. These name candidates are then used to group the instance descriptions so that each group contains only descriptions of one instance. Furthermore, a new name validation method is used to extract the real instance name from the candidate, which finally leads to high precision and recall (F-measure >0.9) for the instance name recognition process.

2.4 Fact extraction and instance generation

In this last step of the ontology instantiation process, we finally aim at the extraction of the exact or most plausible information from the many documents retrieved in the previous phases and insert the extracted instances into the ontology.

As a starting point we use the clustered and annotated documents (authorities) in which possible attribute-value pairs were marked up by the content spotters and for which instance names were determined in the previous step. What is thus left to do is to resolve those problematic situations in which we for instance have multiple value candidates for an individual attribute or have values that do not match the given domain ontology.

In a first step we therefore determine for each identified attribute-value pair if something has to be done and differentiate between the following situations.
(1) The found attribute name is valid (there exists an unambiguous match with the class defined in the domain ontology) and the value is consistent with the domain ontology.
(2) The attribute name is valid, but we have multiple value candidates that were found by the content spotter.
(3) The attribute is valid, but the value found is not consistent with the ontology, i.e. it violates a value restriction defined in the domain ontology.
(4) The found attribute name cannot be uniquely matched with the ontology; values however, have been found.
(5) The attribute cannot be uniquely matched, and no consistent values are found.
(6) The attribute name is not known at all in the ontology; possible values are given though.
(7) Neither the attribute name is known, nor have any values been found.

Situation (1) is obviously okay and nothing has to be done. Situation (7) on the other hand cannot be resolved. The only thing what we can do about it is to store this information as a hint to the knowledge engineer that maybe the domain ontology should be revised accordingly, i.e., extended with an additional feature.

For the other cases, we first try to resolve the conflict by simply looking at the attribute-value pairs of all documents in the same cluster, i.e., that describe the same instance. For case (2) with multiple value candidates, this would mean that we look if the content spotters detected unique values in the other documents. If for the majority of the other documents such a unique value was found, we accept it as the value for this attribute. The same technique is applied for situations (3) to (6).

When the conflicts cannot be resolved that way, because for instance none of the documents in one cluster contains a valid attribute-value pair as defined in the domain ontology, we apply a *fact extraction technique* similar to those described in [7] and [3]. In particular, we again use search engine queries to find out the "most plausible" values for the attributes and in parallel try to learn (and constantly improve) a suitable threshold value for each attribute that determines whether a value should be accepted or not.

The basic procedure can be summarized as follows. First, we the take the attribute-value pairs of Situation (1), i.e., those for which we are confident that the values are correct, and construct two search engine queries. Query 1 contains the instance name and the name of the attribute. Query 2, *the discriminator query*, in addition contains the value. From the number of hits returned for each query, we calculate the PMI (Pointwise Mutual Information) measure [21] : $PMI(q_1, q_2) = Hits(q_1 \cup q_2) \big/ Hits(q_2)$.

Roughly speaking, we compare the number of all documents that most probably describe the instance to the number of documents that describe this instance but also mention the same value. The basic idea now is to use PMI values, which we calculate for all completely recognized attribute-value pairs from Situation (1), in order to *learn a threshold* (acceptance level) for the more problematic cases. Based on this initial PMI value we are thus able to train a Naive Bayes classifier for each attribute. The classifier, which is used to determine whether a value should be accepted or not, is hence based on the PMI value and the information gained from the analysis of Situation (1) cases.

We then start to try to incrementally resolve the open problems starting with the easier cases from Situation (2) and proceed stepwise to Situation (6). Throughout, we exploit the information we gained from resolving previous cases and also continuously update the threshold values for each attribute.

We will here only sketch the basic idea of the approach using Situation (3) as an example. Let us assume that we have identified the attribute "memory type" but the value "CompactFlash" has been found, which is not consistent with the domain ontology (which may for instance specify that "SecureDigital" and "MemoryStick" are the allowed values). Let us also assume that the problem could not be resolved by looking at the other documents in the cluster. Therefore, we will send the two queries to the search engine as described above. Note that we will use the (inconsistent) value "CompactFlash" in the second query, which for instance leads to a PMI value of 0.002. In the background, however, we know from training the classifier with the assumedly correct attribute-value pairs for "memory type" that the acceptance threshold for this attribute is 0.001. As the newly determined PMI value is higher than the threshold, the system will accept "CompactFlash" as a value for the attribute. In addition, the classifier will be updated and will further on take this new PMI value into account. We apply the same technique - with slight differences - for all Situations (2) to (6).

Of course, adding for instance the value "CompactFlash" to the ontology will introduce an inconsistency in the knowledge base. Note however, that the user (or knowledge engineer) can in such situations use the model-based debugging facility of the ALLRIGHT system [10], which is not only capable of detecting these inconsistencies but also of generating corresponding repair proposals.

3 Evaluation

We evaluated our approach in the two popular domains of digital cameras and notebooks, for which we imported domain ontologies from two commercial knowledge-based advisory applications built with ADVISOR SUITE, see http://bitsuperstore.de. Both ontologies included definitions of around 40 attributes; an attribute definition comprises the attribute type, keywords, value restrictions and so forth. The ALLRIGHT system took those ontologies as the only input and performed the instantiation process fully automatically. In the data collection step, 3135 observations of digital camera instances from 18 Web sites and 2930 observations of the notebook domain have been extracted as described in Section 2.2. Note that after this step it is not guaranteed that there are no "false positives" in the observations like pages that describe photo printers which share similar attribute names with digital cameras. The digital camera observations contained 85 false positives, the notebook observations 34, which are to be removed in the clustering step. If only one page with an instance description was found it was also considered as a "false positive" since we cannot check the reliability of this data using redundancy.

To evaluate the accuracy of *instance name recognition* (Section 2.3), we analyzed inputs and outputs by hand and compared these manually defined groups with automatically created ones. This also gave us the number existing clusters, i.e., the real number of cameras in the observations. An automatically created cluster was considered correct if it did not contain false positives and there was no other cluster that contained a description of the same instance. For the domain of digital cameras the system created 498 groups, i.e., 498 camera models have been located for which more than one description existed. 234 groups were generated for the domain of notebooks. Note that during the creation of clusters, many "false positives", for which only one description existed, are filtered out. The clustering and name recognition results (Table 1) are presented in terms of standard information retrieval (IR) metrics: Precision, Recall and F-Measure (F1).

As described in Section 2.4, the clusters of the descriptions allow us to resolve conflicts in the content spotter mark-up. The size of attribute-value sets for the recognition Situations (1 - completely recognized) to (7 - no match) are presented on Fig. 5. It can be seen that the size of the set that corresponds to Situation 7 is rather big. This is mainly because a lot of attributes were not specified in the ontology. In the domain of digital cameras, for instance, some review sites contain over 60 attribute descriptions for their cameras, although only 38 of them were defined in the Web store's ontology. The same tendency could be observed for notebooks. A big number of observations in the 7th category is a good indication for a user to review the domain ontology and probably add some attribute definitions. Averages of IR metrics were calculated by comparing the results generated by automatic fact extraction with handcrafted ones. If a value was falsely accepted, had a wrong value or was not assigned at all, it was considered as incorrect. The resulting averages of IR measures for fact extraction are presented in Table 2.

The fact extraction system performed best with the single-valued features like the "Effective pixels" attribute from the Digital Camera domain. The learned threshold value was very effective leading to an average F1-value of 0.813. Even better performance was achieved in the Notebook domain. For the "Processor" attribute, values were extracted with an average F1-value of 0.859, "RAM" with an F1-value of 0.923. The av-

Domain	Precision	Recall	F-Measure
Digital Camera	0.964	0.965	0.964
Notebook	0.943	0.918	0.93

Table 1. Clustering and name extraction results

Domain	Precision	Recall	F-Measure
Digital Camera	0,738	0,91	0,815
Notebook	0,878	0,9275	0,902

Table 2. Fact Extraction: Average Precision, Recall and F-Measure for two evaluation domains.

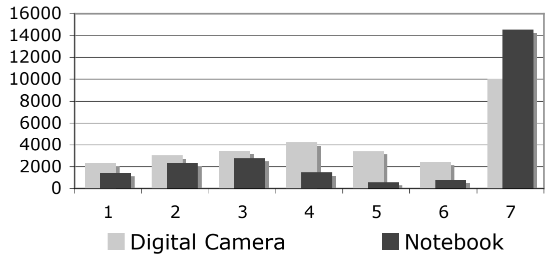


Fig. 5. Recognition situations: number of attribute-value pairs after cluster-based resolution.

erage F1-values for multi-valued attributes like the "ISO Rating" were typically lower (average F1: 0.766), mostly because the number of found values for such attributes varies from instance to instance. So, if for instance the average number of values in training data was smaller than the actual average in the whole data set, the threshold will be very restrictive.

The overall ALLRIGHT instantiation accuracy was assessed using *analysis of variance* (ANOVA) in order to reduce the amount of manual validation: Instead of reviewing nearly 19,000 attribute values extracted for the digital camera domain we reviewed only 946, which is a sufficient sample size according to the Cochran criteria with the level of acceptable error at 1%. For each of the attributes we thus created a random sample of extracted attribute-value pairs.

In order to calculate Precision, we used these samples to estimate the number of correct values and divided it by the number of all values that were found by the ALLRIGHT system. Recall was calculated as a ratio of the estimated number of correct values to the number of all ontology instances found by ALLRIGHT. This was basically done under assumption that the general goal is to instantiate all specified attributes. Also note that the final results are also influenced by "empty" attributes. Since we imported the domain ontology from an existing application, there were some out-of-date attribute definitions, which were never found on the Web pages. These empty values were considered as faulty in our evaluation strategy thus decreasing the Recall.

The obtained IR measures are shown in Table 3. Overall, over 77% of all instances of the digital camera ontology and over 85% of the notebook ontology were correctly instantiated. We view these results to be highly promising, in particular as we started with a very small amount of seed knowledge and only relied on domain-independent

Domain	Precision	Recall	F-Measure
Digital Camera	0,819	0,74	0,777
Notebook	0,834	0,883	0,858

Table 3. ALLRIGHT attribute-value extraction results.

heuristics and metrics, i.e., for instance no manually defined text patterns for content spotting and so forth have been used.

4 Summary and outlook

In this paper we have shown how the process of populating an ontology for *knowledge-based systems* with instance data can be automatized by retrieving and extracting the corresponding information from Web sources. The presented ALLRIGHT system relies on a novel combination of new and existing techniques from different areas and is in particular optimized for dealing with information which is contained in tabular form. The evaluation of the approach in two popular domains showed that highly precise instance information can be retrieved even if only a small amount of seed knowledge is available. We thus see our work as a contribution toward the broader application of ontology- or semantics-based applications whose adoption is in many cases hampered by the fact that larger amounts of ontology instance data have to be manually entered to the system.

We currently aim to extend and further improve the ALLRIGHT system in several dimensions. On the technical level, we evaluate additional crawling techniques and heuristics that go beyond the "index-page" approach and which can also better handle dynamic HTML pages. Also, the generated content spotters will be extended such that they are able to *learn* and thus improve their accuracy over time.

Beside these technical improvements, we are currently evaluating the results of the *relationship extraction* module, with which we for instance aim at detecting which kinds of accessories (like memory cards) are compatible with a certain camera model. The first evaluations, which are based on an ontology alignment technique and on similar fact extraction methods like the ones described herein, are already very promising but go beyond the scope of this paper.

Acknowledgments

We would like to thank the reviewers for their valuable comments.The research project is funded partly by the grant from the Austrian Research Promotion Agency, Programm Line FIT-IT Semantic Systems (Project AllRight, Contract 809261) and the Austrian Science Fund FWF (Project V-Know, Contract 19996).

References

1. H. Alani, S. Kim, D. E. Milard, M. J. Weal, W. Hall, P. H. Lewis, and N. R. Shadbolt. Automatic ontology-based knowledge extraction from web documents. *IEEE Intelligent Systems*, 18:14–21, 2003.
2. S. Chakrabarti, M. van den Berg, and B. Dom. Focused crawling: a new approach to topic-specific web resource discovery. *Comput. Networks*, 31(11-16):1623–1640, 1999.
3. P. Cimiano, G. Ladwig, and S. Staab. Gimme' the context: context-driven automatic semantic annotation with c-pankow. *Proceedings of the 14th international conference on World Wide Web*, pages 332–341, 2005.

4. M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. Learning to construct knowledge bases from the World Wide Web. *Artificial Intelligence*, 118(1):69–113, 2000.
5. S. Dill, J. Tomlin, J. Zien, N. Eiron, D. Gibson, D. Gruhl, R. Guha, A. Jhingran, T. Kanungo, S. Rajagopalan, et al. SemTag and seeker: bootstrapping the semantic web via automated semantic annotation. *Proceedings of the Twelfth International Conference on World Wide Web*, pages 178–186, 2003.
6. D. W. Embley, D. M. Campbell, Y. S. Jiang, S. W. Liddle, Y.-K. Ng, D. Quass, and R. D. Smith. Conceptual-model-based data extraction from multiple-record web pages. *Data Knowledge Engineering*, 31(3):227–251, 1999.
7. O. Etzioni, M. Cafarella, D. Downey, A. Popescu, T. Shaked, S. Soderland, D. Weld, and A. Yates. Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence*, 165(1):91–134, 2005.
8. A. Felfernig, G. Friedrich, D. Jannach, and M. Zanker. An integrated environment for the development of knowledge-based recommender applications. *International Journal of Electronic Commerce*, 11(2):11–34, Winter 2006-7 2007.
9. A. Felfernig, G. Friedrich, and L. Schmidt-Tieme. Recommender systems. *IEEE Intelligent Systems - Special Issue on Recommender Systems*, 22(3), May 2007.
10. G. Friedrich and K. Shchekotykhin. A General Diagnosis Method for Ontologies. *Proceedings of the 4 thInternational Semantic Web Conference (ISWC-05)*, pages 232–246, 2005.
11. G. Friedrich and K. Shchekotykhin. NameIt: Extraction of product names. *Data Mining Workshops, 2006. ICDM Workshops 2006. Sixth IEEE International Conference on*, pages 29–33, 2006.
12. P. G. Ipeirotis, E. Agichtein, P. Jain, and L. Gravano. To search or to crawl?: towards a query optimizer for text-centric tasks. In *SIGMOD '06: Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 265–276, New York, NY, USA, 2006. ACM Press.
13. T. Jayram, R. Krishnamurthy, S. Raghavan, S. Vaithyanathan, and H. Zhu. Avatar information extraction system. *IEEE Data Engineering Bulletin*, 29(1):40–48, 2006.
14. J. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632, 1999.
15. B. Krüpl, M. Herzog, and W. Gatterbauer. Visually guided bottom-up table detection and segmentation in web documents. In *The 15th International World Wide Web Conference (WWW2006)*, 2006.
16. F. Naumann, A. Bilke, J. Bleiholder, and M. Weis. Data fusion in three steps: Resolving schema, tuple, and value inconsistencies. *IEEE Data Eng. Bull*, 29(2):21–31, 2006.
17. J. Pan, H. Yang, C. Faloutsos, and P. Duygulu. Automatic multimedia cross-modal correlation discovery. *Proceedings of the 2004 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 653–658, 2004.
18. D. Pelleg and A. W. Moore. X-means: Extending k-means with efficient estimation of the number of clusters. In *Seventeenth International Conference on Machine Learning*, pages 727–734. Morgan Kaufmann, 2000.
19. G. Petasis, V. Karkaletsis, and C. Spyropoulos. Cross-lingual information extraction from web pages: the use of a general-purpose text engineering platform. In *4th International Conference on Recent Advances in Natural Language Processing (RANLP), Borovets, Bulgaria*, 2003.
20. A. Popescu and O. Etzioni. Extracting product features and opinions from reviews. *Proceedings of EMNLP 2005*, 2005.
21. P. Turney et al. Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 417–424, 2002.