ontoX - A Method for Ontology-Driven Information Extraction

Burcu Yildiz[1] and Silvia Miksch[1,2]

[1] Institute for Software Technology and Interactive Systems,
Vienna University of Technology, Vienna, Austria
{yildiz, miksch}@ifs.tuwien.ac.at
[2] Department of Information and Knowledge Engineering,
Danube University Krems, Krems, Austria
silvia.miksch@donau-uni.ac.at

Abstract. Information Extraction (IE) is an important research field within the Artificial Intelligence community, for it tries to extract relevant information out of vast amounts of data. In this paper, we propose an extraction method that utilises the content and pre-defined semantics of ontologies formulated in the Web Ontology Language (OWL) to perform the extraction task. We also propose our method to detect out-of-date constructs in the ontology to suggest changes to the user. After stating the results of our evaluation, we conclude that the use of ontologies in conjunction with IESs can indeed yield feasible results and contribute to the better scalability and portability of the system.

Key words: Ontology-Driven Information Extraction, Information Extraction, Ontology Engineering

1 Introduction

Since Berners-Lee proposed and started to endorse ontologies as the backbone of the Semantic Web in the nineties [1], a whole research field evolved around the fundamental engineering aspects of ontologies, such as the generation, evaluation, and management of ontologies. However, many researchers were curious about the use of ontologies within Information Systems (ISs) in 'ordinary' settings, performing 'ordinary' information processing tasks [2], where our focus lies on the use of ontologies within Information Extraction Systems (IESs) [3].

Information Extraction (IE) is defined as a form of natural language processing in which certain types of information must be recognised and extracted from text [4] [5]. It is difficult to describe what 'relevant information' actually is, and it is even harder to communicate this description to a computer system. For that purpose we analyse whether and how ontologies can be utilised to provide IESs with an unambiguous and formal description of relevant information. However, to enable the smooth integration of ontologies within IESs, several requirements have to be reconciled [6].

In this paper, we present a method for ontology-driven Information Extraction from natural language text, where the ontology is used as a formal and thus unambiguous "specification of the conceptualisation that represents the particular domain of interest" [7]. We analyse to what extent ontologies formulated in the Web Ontology Language (OWL), are suitable to drive the IE process and how a classical IES has to be augmented in order to deal with such an underlying ontology properly. Within this context we also analyse what kind of management issues regarding the ontology arise and what course of action would be appropriate to dissolve them.

2 Related Work

To use ontologies within the context of 'ordinary' IESs is a relatively new research field. We have to state that to our knowledge only the work of Embley [8] can be considered to be *ontology-driven*, whereas the other works are *ontology-based*, because therein the extraction process is not driven by the ontology rather the ontology is used during extraction as yet another component of the system.

Embley [8] presents an approach for extracting and structuring information from data-rich unstructured documents using extraction ontologies. With data-rich he means data that has a number of identifiable constants such as dates, names, times, and so forth. He proposes the use of the Object-oriented Systems Model (OSM) [9] to represent extraction ontologies, because it allows the formulation of regular expressions as descriptors for constants and context keywords. Both, the generation of the ontology and the generation of the regular expressions require human intervention, because they have to be performed manually. The generated ontology is then parsed to build a database schema that covers the required constants and to generate extraction rules to match constants and keywords. After that, recognisers are invoked that use these extraction rules to identify potential constant data values and context keywords. Finally, the generated database is populated using heuristics to determine which constants populate which records in the database.

Aitken [10] presents an approach to learn IE rules from natural language text using Inductive Logic Programming (ILP). In his approach an ontology is used only as a knowledge bearing artifact that represents the conceptualisation of interest and to which a human annotator can commit to while making annotations on a data corpus. His supervised induction algorithm then uses these annotations to generate extraction rules.

McDowell and Cafarella [11] present an automatic and domain-independent ontology-driven IES called OntoSyphon. Their system takes an ontology as input and uses its content for specifying web searches to identify possible semantic instances, relations, and taxonomic information. For example, for a concept Mammal in an ontology, the system specifies web searches using the phrase patterns introduced by Hearst [12], like "mammals such a", etc. The system then searches the Web for occurrences of these phrases and extracts candidate instances.

However, our aim is to provide an unsupervised ontology-driven IES, that is able to exploit the content of its underlying ontology on its own to extract relevant information from natural language text without being dependent on other knowledge sources.

3 ontoX - an Ontology-Driven IES

In this section, we will explain our method to IE from natural language text, that utilises the knowledge in an ontology. The input ontology is being used as a knowledge bearing artifact that represents the conceptualisation of a domain of interest and the task specification for the extraction task. Our aim by developing this method is to provide a means for 'common' people to perform IE in a way that requires neither skills in particular rule representation languages, nor any other resource but the ontology, such as lexicons, etc. The main architecture of our IES, which we named 'ontoX', is depicted in Fig. 1.

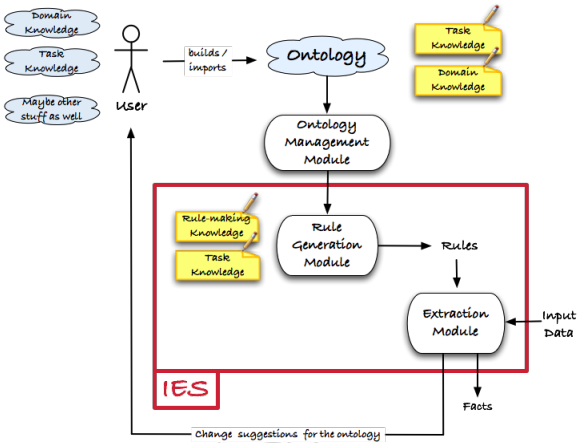

Fig. 1. Main architecture of ontoX

The Ontology Management Module (OMM) takes the input ontology and tries to exploit the knowledge in it to determine what exactly has to be extracted from the input data. Although our method can be applied to any other representation language, our implementation supports ontologies formulated in the Web Ontology Language (OWL) version 1.0 [13] [14]. The Rule Generation Module (RGM) uses the output of the OMM and performs several steps to formulate rules (in our case regular expressions) to locate candidate values that are relevant according to the input ontology. The Extraction Module (EM) takes these rules and determines candidate values in the input texts and applies

several heuristics to choose the most accurate values among them. This module finally returns the extracted values and also suggestions to the user regarding possible changes in the ontology.

To support the understandability of our extraction method and its implementation, we explain them by using an example. For that purpose we have chosen the domain of digital cameras, because the domain is widely known and its characteristics are suitable for extraction. Figure 2 shows a possible conceptualisation of the domain containing only relevant aspects of the domain we are interested in. In this sense, we can think of the ontology as the task specification where the properties of concepts represent the properties for which our extraction method has to determine appropriate values from input data.

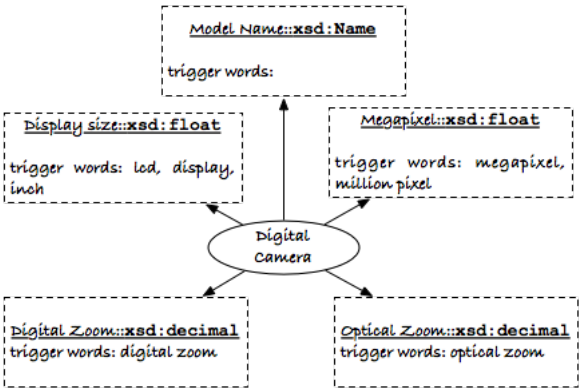

Fig. 2. Graphical representation of our digital camera ontology

3.1 Requirements

Although our method can process any valid ontology that is conform with OWL 1.0, several requirements have to be reconciled by the ontology to enable ontology-driven IE. In this section, we show how these additional requirements can be reconciled by extending ontologies with additional properties. All these properties can be attached to the ontological constructs in OWL using the `owl:AnnotationProperty` element.

Keywords The most important requirement to enable ontology-driven IE is to enrich the ontological constructs with keywords (i.e., trigger words) that indicate the presence of relevant information in the input text. If the user does not provide a property with corresponding keywords, our system tries to extract appropriate values considering frequent terms in the neighbourhood of other

properties' keyword occurrences. Note that this can only return feasible results when enough properties co-exist and not more than one such unknown property has the same data type.

Quality Properties Within our IE method we allow the user to enrich the constructs in the ontology by a property *confidence_level*, which can take values between $[0, 1]$ indicating the level of confidence of the ontology engineer that this construct is really relevant for the domain. This property helps our extraction method to make decisions when the same value is tried to be assigned to two different properties. In such a case, the property with the higher confidence level would be the winner.

Another property that can be attached to ontological constructs is the *relevance* property that can take one of the two values {true, false}. By marking a construct with this property the user can tell the system that she is not interested in the construct as far as the task specification is concerned, but merely that the construct is part of the domain of interest.

Value Constraint Properties Constraining properties are required to narrow the search range of possible values of properties that have to be found. In our method, this is possible as far as the OWL 1.0 specification allows it. This means, that our extraction system only takes the XML Schema data types into account whose usage is legal in OWL 1.0.

Temporal Properties Temporal properties can be useful in the context of IESs to enable two kinds of services. The first one is temporal extraction and the second is change management. With the first one, a user can state that she wants her input data to be extracted using ontological components that are valid at a certain point of time. With the second one, the user can be provided with suggestions regarding out-of-date concepts because they did not appear in the input texts anymore, so that she could adopt the ontology if necessary.

To enable both kinds of services, we suggest the use of *valid_time_begin* and *valid_time_end* properties that can be stated for every construct in the ontology. By doing this, the input data will be analysed using only ontological constructs that are valid at the given point of time, which must be provided by the user of the system in a way that is conform with the `xsd:Date` data type. If no date is given, all constructs in the ontology will be used for extraction.

To enable change management we also provide the user with the property *value_change_frequency* that can take one of the two values {stable, frequent}. The user can mark ontological components with this property in order to indicate that the component will appear in the input text consequently (i.e., stable) or not (i.e., frequent). Using this value an IES can compute accuracy levels for ontological components after each extraction operation, looking whether appropriate values for the component had appeared in the input text or not.

3.2 The Rule Generation Module

The Rule Generation Module (RGM) of ontoX is responsible for generating extraction rules that can be used to identify candidate values for the attribute-value pairs generated by the OMM. Because OWL 1.0 supports only a predefined set of data types, at the present this module merely has to look up appropriate regular expressions for the stated data types.

However, for other ontology representation languages the RGM would have to do a lot more. For the oncoming standard OWL 1.1, which will support user-defined data types for example, the RGM would have to parse the data type definitions in order to generate appropriate extraction rules.

3.3 The Extraction Module

The Extraction Module (EM) of ontoX is responsible for applying the rules generated by the RGM and using them to identify candidate values for properties in the ontology. The steps that are taken by the EM can be considered in two main parts: preprocessing step and extraction step.

Preprocessing is needed to transform the input data into a format that can be processed more easily by successive modules (e.g., eliminating noisy data that could affect the performance of the system). The preprocessing phase of our extraction method comprises the following particular steps.

Removing stop words Natural language texts are usually loaded with words, called stop-words, which do not convey relevant information but occur frequently in the text as constructs to build grammatically correct sentences. Because of their high frequency, they often cause outliers in term frequency histograms and have to be eliminated from the input text before starting the actual extraction phase. As we evaluate our method with texts in English, at this step we eliminate all stop words commonly used in English.

Locating Data Type Occurrences We stated earlier that data type properties represent our main interest w.r.t. extraction, for they define the actual values of interest to be extracted. However, the XML Schema data types have different value spaces, which range from integers, bytes, floats to more general data types such as strings. It is clear that the more constraints stated about a property, the easier it will be to identify a particular text as a value of that property. In order to rule out cases where the data types with wider value spaces overlap data types with narrower value spaces, we begin with locating occurrences of the narrower ones.

Values of data types derived from `xsd:string` are much more harder to identify than numbers and other data types that have fixed formats. Therefore, simple pattern matching methods cannot be applied on them. For these data types, especially for the data type `xsd:Name`, we use a heuristic that computes the *named_entity_probability* of a string. To compute this value, we consider anomalies in a string, like as numbers and characters appearing

in the same string, or mixed lower and upper case usage of characters that could indicate that the string at hand is not a proper word

Extraction The extraction phase is the actual phase where the located data type values are going to be assigned to corresponding properties in the ontology. Figure 3 depicts sample input data after the preprocessing phase. As you can see, all the stop words have been removed, all occurrences of data type values that are allowed in OWL 1.0 are underlined, and keyword occurrences are also highlighted by underlining them with two lines (i.e., 'megapixel', 'optical zoom', 'digital zoom', 'inch', 'LCD').

5.0-megapixel D-595 Zoom combines power manual control convenience easy-to-use design incredible results. Featuring manual control shutter speed aperture options, 3x optical zoom 19 shooting modes, D-595 Zoom combines manual settings ease of point-and-shoot, packing high-end advanced features affordable, easy-to-use compact camera.

3x optical zoom lens (38-114mm equivalent 35mm photography f2.8 - f4.9) combines 4x digital zoom deliver total 12x zoom so virtually photo opportunity out of reach. super macro mode possible capture amazingly small details, tiny date back penny, from close 0.8 inches.

TruePic TURBO super-charged image processor significantly enhances image quality processing speed. Users experience rapid startup, shutter release playback, capture sharper, more realistic images. One greatest benefits digital photography ability compose review images LCD, allows users accurately frame photographs make on-the-spot decisions about saving deleting them. ensure users can easily view subject frame best-shot possible, D-595 Zoom has, 85,000 pixel 1.8-inch Semi-Transmissive TFT LCD - easy viewing even bright sunlight.

Fig. 3. Sample input data after preprocessing

Now we have to assign appropriate values to the properties in the ontology that the identified keywords belong to. For that purpose, we are looking for values that are conform with the predefined data type of the property. In fact, we are looking for the first such value at the left side and the right side of a keyword occurrence, because it is more likely that the values are located near the keywords. For the keyword 'megapixel' that must have a value conform with `xsd:float`, we have the value '5.0' at its left and no appropriate value at its right, because there is a sentence boundary between the keyword and the next valid value. The heuristic used to choose appropriate candidate values for each keyword occurrence is formulated in the following algorithm.

Algorithm 1: Selecting Candidate Values

```
L_k := List of all keyword occurrences in Input
L_d := List of all data type occurrences in Input

for each keyword k in L_k do
 r := k.datatype  // look up the data type of k
 dp := k.property // look up the property to which k belongs

 repeat
   c_l := next token at the left
 until c_l is a valid value or a sentence boundary

 repeat
   c_r := next token at the right
 until c_r is a valid value or a sentence boundary

 if c_l == sentence boundary and c_r is not, then
   candidate_value = c_r
 else if cr == sentence boundary and c_l is not, then
   candidate_value = c_l
 else   // both, c_l and c_r are valid values
   c_l.evidence = compute evidence for c_l
   c_r.evidence = compute evidence for c_r

   if c_l.evidence > c_r.evidence then
     candidate_value = c_l
   else if c_r.evidence > c_l.evidence then
     candidate_value = c_r
   else   // both values have the same evidence
     if r == xsd:string or a derivative of xsd:string then
       candidate_value = c_r
     else
       candidate_value = c_l
     end if
   end if
 end if

 add candidate_value to the list of candidate values of dp
end for
```

A property may have been provided with more than one keyword, in which case every occurrence of each keyword will be encountered to collect candidate values in the input text. While collecting candidate values, our method marks them with a level of *evidence* that is computed as

$$evidence = \begin{cases} \frac{1}{d} \text{ if } d > 0 \\ 1 \text{ if } d = 0 \end{cases} \quad (1)$$

whereas d is the distance of the candidate value from the keyword occurrence in terms of the words that lie between them. This inverse distance function is used to favour data type values that are near to the keyword over values that are more far away.

If the same data type value appears more than one as a candidate for a certain property, the level of evidence of this value is being updated to the maximum between them:

$$evidence = max(evidence_{old}, evidence_{new}) \quad (2)$$

After having identified all candidate values for a certain property in the ontology we have to choose the final result among them. We already mentioned that OWL allows the definition of functional properties (i.e., `owl:FunctionalProperty`) that is, properties that can have only one value. For these kinds of properties our method chooses the candidate value with the highest computed evidence. For other properties it presents all candidate values whose evidence are above a user defined threshold.

For our sample input text in Figure 3 we would have the attribute-value pairs and their candidate values with decreasing order of their computed evidence as shown in Table 1.

Table 1. Candidate values and the final decisions for properties in the ontology

	Model	**Megapixel**	**Display**	**Optical zoom**	**Digital zoom**
Data type	xsd:Name	xsd:float	xsd:float	xsd:decimal	xsd:decimal
Candidate values	D-595 Zoom Semi-... TFT TruePic TURBO	5.0 1.8	1.8 0.8	3 19 4	4 12
Final decision	D-595 Zoom	5.0	1.8	3	4

3.4 Change Detection

We think that it is essential to provide ontology-driven systems with means to enable the detection of changes in the underlying conceptualisation. Therefore, we mentioned earlier that ontologies have to be enriched with several additional components, such as the *value_change_frequency*. We think that in our setting it is sufficient to generate a log-file that shows which ontological constructs became out-of-date over time to be suggested to the user for her to change the ontology

because it apparently contains constructs that do not occur in the input files anymore.

Our heuristic to detect out-of-date constructs works in a way that incorporates the property *value_change_frequency* and the amount of time over which a property in the ontology did not occur in the input files. The first idea was to decrease the value that indicates the probability that a certain construct in the ontology is still accurate (property *accuracy*) for let's say 20% every time a construct did not appear in the input data. It is clear that such a course of action would not lead us to satisfactory results, because it does not encounter the predicted value change frequency or the confidence level of the construct, which indicated the level of confidence for this construct at the first place. Therefore, we adjusted our idea and defined our function to determine the value of accuracy as follows:

$$accuracy_{new} = accuracy_{old} - \frac{(accuracy_{old}/5) * value\_change\_frequency}{confidence\_level} \quad (3)$$

This function ensures that the value of accuracy decreases at a faster pace for constructs that are said to be 'stable' than for constructs that are predicted to be 'frequent' anyway. The function also ensures that the value decreases at a slower pace for constructs of which the ontology engineer was more certain.

4 Experimental Results

In this section, we present the evaluation results of our proposed method to extract information from natural language text using ontologies formulated in OWL 1.0. To provide We use *Recall* and *Precision* [15] as established evaluation measures among the community. However, the newness of the field of ontology-driven IE and the fact that there are no freely accessible evaluation data sets that could be used by different IES developers, makes it hard to provide comparative results of different systems.

4.1 Evaluation of Performance

In the first part of our conducted evaluation we focus on the performance of our proposed extraction method that utilises only a relatively small ontology to extract relevant facts from natural language text documents. Further, we evaluate the usage of ontological relations during the extraction process when no keywords are attached to some properties.

We have chosen the domain of digital cameras, because of its popularity nowadays and the fact that its nature can be captured easily by ontologies. We collected a set of 137 digital camera reviews in natural language text from the Web[3] with over 57,000 words[4]. Table 2 depicts the results of our extraction

[3] http://www.steves-digicams.com/ [accessed last in April 2007]

[4] To download the data set and the corresponding result set visit our project page at http://ieg.ifs.tuwien.ac.at/projects/ontox/

method for the collected data corpus and the ontology as in Figure 2 in terms of the standard evaluation metrics recall and precision.

	Number of Facts	Correctly Identified Facts	Incorrectly Identified Facts	Recall	Precision
Model	137	110	28	0,79	0,79
Megapixel	137	70	63	0,51	0,52
Optical zoom	124	105	22	0,84	0,82
Digital zoom	13	6	6	0,46	0,46
Display size	113	93	23	0,82	0,80

Table 2. Evaluation results for the digital camera ontology in Figure 2

Analysing the results, we must admit that we had expect better results for the property 'Megapixel' because the keywords for that property are relatively clear and occur consistently in input documents and because its value space is also narrow. However, we figured that some reviews contained the megapixel information as decimals and sometimes even in letters (e.g., 'five') causing confusion in the extraction process.

On the other hand, the results for the property 'Model' were a pleasant surprise. We did not expect our method to locate appropriate values for this property that well, which was not provided with any keywords. The reason for this is that the model name of a digital camera appears relatively often in the text and therefore falls more often in the neighbourhood of other keywords' occurrences.

The number of incorrectly assigned values for the 'Digital Zoom' property is due to data type occurrences that could not be assigned to their original properties. Better means to state more complex data types can decrease the number of incorrect values, leading to better precision results.

4.2 Evaluation of Scalability and Portability

To evaluate how our method reacts to changes in the task specification, we changed our task specification so that it stated that the system has to extract also the type of storage medium a digital camera supports (e.g., xD-Picture Card, etc.), the kind of power supply it has (e.g., Lithium Ion Battery, etc.), and also information about supported video formats (e.g., MPEG format, etc.). Table 3 contains the results of our extraction method for the same data corpus and the changed ontology in terms of the standard evaluation metrics recall and precision.

The relatively bad results for the property 'Storage' show how the choice of inappropriate keywords for properties can affect the performance of the system. We figured that the keyword 'storage' is used apparently in another context as

well, leading to a lot of incorrectly identified values, which was fostered further by the rather large value space that the data type `xsd:Name` is allowing. The same can be said for the large number of incorrectly assigned values for the properties 'Movie Format' and 'Power Source'. The fact that all of them allow values from the `xsd:Name` value space, lead to a lot of candidate values with some of them apparently very close to the keywords.

	Number of Facts	Correctly Identified Facts	Incorrectly Identified Facts	Recall	Precision
Storage	61	15	56	0,25	0,22
Movie Format	56	41	59	0,73	0,41
Power Source	60	26	64	0,43	0,28

Table 3. Evaluation results for the changed task specification

However, the aim with this phase in our evaluation was to look how difficult it is to make the system extract information for a different task specification. As such a change only requires the change of the input ontology and not the IES itself, we conclude that the scalability and portability of ontology-driven IESs are indeed much better than of regular IESs.

4.3 Evaluation of Change Detection

Another aspect of our method was to enable basic change detection in the extraction interests of the user. We stated earlier the heuristic our method applies to determine the accuracy of properties to identify properties or concepts that do not appear in the input text over a certain amount of time (see Section 3.3). After every extraction operation the user is given a list of all ontological constructs in increasing order of their accuracy level, so that she can change the ontology if necessary.

During our described evaluation phase, our method returned the following accuracy list after the first ten input texts had been processed:

$0,129$ accuracy of 'Digital zoom'
$0,36$ accuracy of 'Optical zoom'
$0,6$ accuracy of 'Megapixel'
1 accuracy of 'Model'
1 accuracy of 'Display size'

Looking at such a list, the user can see that the 'Digital Zoom' property is not well represented in the input texts, indicating either that the property lost its relevance over time (i.e., the domain of interest had changed over time) or that the keywords or the data type of this property are not suitable to extract the desired information. Both cases require the user to look at the input texts and the ontology to figure out which one of these cases is actually true.

5 Conclusion and Future Work

In this work we presented an extraction method that can extract information from natural language text without using any knowledge resource except an input ontology. We thought that such a system would be useful for people with light-weight extraction demands and people who are not familiar with generating all kinds of knowledge resources (e.g., gazetteer lists, extraction rules, etc.) or do not have access to linguistic processing resources (e.g., part-of-speech tagger, etc.) that other state-of-the-art IESs require to yield feasible results.

Because of this focus of our work, we have to live with some limitations, which can be overcome if several requirements on the input ontologies are reconciled:

- The provided keywords have to be chosen carefully, because they are the trigger words using which our method locates candidate values.
- The ontology should contain as much information as possible regarding the constraints on property values. For example, if it is known that values of a particular property can have only values greater than 1, then its datatype should be `xsd:positiveInteger` instead of just `xsd:integer` or other data types with a wider value space.

In short we can say, that the performance of our proposed method highly depends on the quality of the input ontology. If the user can provide the system with an ontology that reconciles the requirements stated above, the results of the system will be as outlined in the experimental results (see Section 4.1).

The system as it is, can be seen as a scalable and portable IES, because to adopt the system to a changed specification or a new domain, only the ontology has to be changed. Some domains require better performing IESs and therefore it is likely that some IES developers will tailor their systems for only a particular domain at hand. But although this would cause a decrease in the portability of the system, it would be easier still to scale than with other approaches to IE.

However, a lot of work has to be done in this research field to convince users that it is indeed beneficial to use ontologies in conjunction with IESs. Possible directions for future work may include the following:

Developing an extension to OWL for IE In this paper we proposed several properties that should be used to enrich the existing components of OWL to represent conceptualisations. We think that the more OWL becomes the quasi standard for representing ontologies among the AI community, an extension to OWL for IE could be useful for researchers interested in IE and thus worth the effort. Such an extension to OWL for IE should contain means to represent user-defined data types, quality properties, temporal properties, and linguistic properties.

Utilising intentional knowledge for better extraction results In our proposed method we focused only on the extensional knowledge present in ontologies

but merely neglected the intensional knowledge (i.e., instances) if they were not directly related to properties of interest (i.e., `owl:ObjectProperty`). However, intentional knowledge can turn out to be useful for an IES to compare its identified candidate values with existing values of instances to make decisions based on similarity measurement.

References

1. Berners-Lee, T.: Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by Its Inventor. Harper San Francisco (1999)
2. Guarino, N.: Formal ontology and information systems. In Guarino, N., ed.: Proceedings of the First International Conference on Formal Ontologies in Information Systems (FOIS). (1998) 3–15
3. Yildiz, B., Miksch, S.: Motivating ontology-driven information extraction. In Prasad, A., Madalli, D., eds.: International Conference on Semantic Web and Digital Libraries. Indian Statistical Institute Platinum Jubilee Conference Series (2007) 45–53
4. Riloff, E.: Information extraction as a stepping stone toward story understanding. In: Understanding Language Understanding: Computational Models of Reading. MIT Press, Cambridge, MA, USA (1999) 435–460
5. Appelt, D.: Introduction to information extraction. AI Communications **12** (1999) 161–172
6. Yildiz, B., Miksch, S.: Ontology-driven information systems: Challenges and requirements. In Prasad, A., Madalli, D., eds.: International Conference on Semantic Web and Digital Libraries. Indian Statistical Institute Platinum Jubilee Conference Series (2007) 35–44
7. Gruber, T.: Toward principles for the design of ontologies used for knowledge sharing. International Journal of Human-Computer Studies **43** (1993) 907–928
8. Embley, D.: Toward semantic understanding: An approach based on information extraction ontologies. In: Proceedings of the Fifteenth Conference on Australasian Database Conference, Australian Computer Society, Inc. (2004) 3–12
9. Embley, D., Kurtz, B., Woodfield, S.: Object-oriented systems analysis: a model-driven approach. Yourdon Press, Upper Saddle River, NJ, USA (1992)
10. Aitken, J.: Learning information extraction rules: An inductive logic programming approach. In: Proceedings of the 15th European Conference on Artificial Intelligence (ECAI'02). (2002) 355–359
11. McDowell, L., Cafarella, M.: Ontology-driven information extraction with OntoSyphon. In: International Semantic Web Conference. (2006) 428–444
12. Hearst, M.A.: Automatic acquisition of hyponyms from large text corpora. In: Proceedings of the 14th International Conference on Computational Linguistics. (1992) 539–545
13. Horrocks, I., Patel-Schneider, P.F., van Harmelen, F.: From SHIQ and RDF to OWL: The making of a web ontology language. Journal of Web Semantics **1**(1) (2003) 7–26
14. Grigoris, A., van Harmelen, F.: A Semantic Web Primer. The MIT Press, Cambridge, Massachusetts, London, England (2004)
15. Cowie, J., Lehnert, W.: Information extraction. Communications of the ACM **39:1** (1996) 80–91