Bottom-Up Extraction and Trust-Based Refinement of Ontology Metadata

Paolo Ceravolo, Ernesto Damiani, *Member*, *IEEE*, and Marco Viviani

Abstract—We present a way of building ontologies that proceeds in a *bottom-up* fashion, defining concepts as clusters of concrete XML objects. Our rough bottom-up ontologies are based on simple relations like association and inheritance, as well as on value restrictions, and can be used to enrich and update existing upper ontologies. Then, we show how automatically generated assertions based on our bottom-up ontologies can be associated with a flexible *degree of trust* by nonintrusively collecting user feedback in the form of *implicit* and *explicit* votes. Dynamic trust-based views on assertions automatically filter out imprecisions and substantially improve metadata quality in the long run.

Index Terms—Semantic Web, bottom-up ontology, ad hoc conceptualization, metadata extraction and maintenance, fuzzy clustering techniques, trusted assertions.

1 Introduction

In the scenario of knowledge-based organizations, *virtual communities* are emerging as a new organizational form supporting knowledge sharing, diffusion, and application processes. Such communities do not operate in a vacuum; rather, they have to coexist with a huge amount of digital information, such as text or semistructured documents in the form of Web pages, reports, papers, and e-mails. Heterogeneous information sources often contain valuable information that can increase community members shared knowledge, acting as high-bandwidth information exchange channels.[1] Experience has shown that exchanging knowledge extracted from heterogeneous sources can improve community-wide understanding of a domain and, hence, facilitate cooperative building and maintenance of shared domain models such as domain ontologies [2]. On today's global information infrastructure, manual knowledge extraction is often not an option due to the sheer size and the high rate of change of available information. In this paper, we describe a bottom-up method for ontology extraction and maintenance aimed at seamlessly complementing current ontology design practice, where, as a rule, ontologies are designed top-down. Also, we show how metadata based on our bottom-up ontologies can be associated with a flexible *degree of trust* by nonintrusively collecting user feedback. Dynamic trust is then used to filter out unreliable metadata, improving the overall value of extracted knowledge. The paper is organized as follows: Section 2 gives an overview of related research, while Section 3 outlines our application of fuzzy techniques to efficient structure-based encoding and clustering of XML data. Section 3.4 describes how content-based clustering can be coupled with structure-based techniques to produce a finer-grained classification. Section 4 contains a worked-out example of XML clustering, while Section 5 describes how clusters can be used as a basis for generating ontology metadata suitable for enriching and updating an existing ontology. Section 6 illustrates how our automatically generated metadata can be given a *trustworthiness* value by collecting explicit and implicit user votes. Trustworthiness supports computing custom views on assertions, reflecting the views of the user community and improving the overall quality of metadata. Finally, Section 7 draws the conclusion.

1. For instance, manual inspection of thousands of posts at `comp.security.firewalls` newsgroup showed their potential of creating and maintaining shared competence [1].

• *The authors are with the Università degli Studi di Milano, Dipartimento di Tecnologie dell'Informazione, Via Bramante, 65-26013 Crema (CR), Italy. E-mail: {ceravolo, damiani, viviani}@dti.unimi.it.*

Manuscript received 3 Oct. 2005; revised 14 Apr. 2006; accepted 18 Aug. 2006; published online 19 Dec. 2006.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDESI-0461-1005.

2 Related Work

Traditionally, the approach to community-oriented ontology building has been a collaborative one aimed at valorizing the contribution of each community member in the knowledge creation activity. Metadata extraction and merging is carried out manually by individual users as a part of their everyday activities, possibly taking sample data items into account. A well-known example is *Ontolingua Server* [3], a Web space where members of a community of ontology developers can access ontologies, browse them, edit them, and propose modifications. This process enables an interdisciplinary vision as each member of the community contributes to the ontology with her own background. More importantly, cooperatively built ontologies can quickly incorporate new "operative knowledge" derived from job experience into existing domain models. However, some drawbacks do exist. Cooperatively building and maintaining an ontology takes a substantial amount of time, even in the presence of a careful process management. For this reason, recent research turned once again to automatic and semiautomatic (as opposed to cooperative) metadata construction and maintenance. Automatic techniques for building and integrating metadata have been studied for many years by the database community, especially in the area of federated databases and object-oriented systems, where extracting and merging inheritance hierarchies [4] is a time-honored problem. More recently, several techniques specifically aimed at learning ontologies from text corpora or database repositories were proposed [5]. Well-known research approaches to ontology learning [6] include:

- *Pattern-based extraction*: Concepts and relations are detected when corresponding word patterns are recognized [7], [8].
- *Conceptual clustering*: Concepts are grouped according to a semantic distance [9] between each other to make up hierarchies. Semantic distances for Web data are described in [10].
- *Association rules mining*: The frequency of an association of terms is computed in the information base. If the frequency of the association is close to the occurrence of individual terms, the association is transformed in an ontological relation. This technique was first introduced for text documents [11] and later generalized to data objects [12].
- *Ontology extraction and merging*: Ontologies extracted from heterogeneous data sources are concsolidated and merged [13].

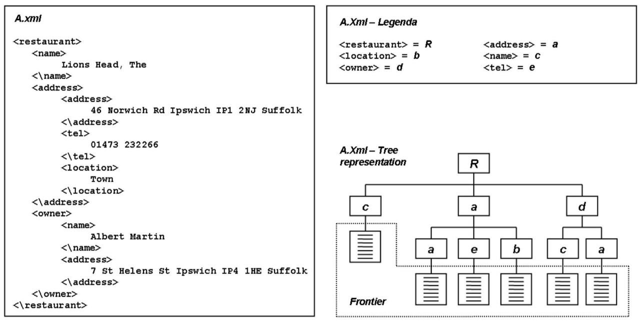
In this paper, we shall focus on quick-and-dirty ontology extraction and merging with existing ontologies. Several graph-theoretical approaches exist to ontology merging, most of them relying on suitable graph algebras. The *Onion* system [14] was born as an attempt to reconcile ontologies underlying different biological information systems. It heuristically identifies semantic correspondences between ontologies and then uses a graph algebra based on these correspondences for ontology composition. However, Onion is aimed at merging fully fledged, competing ontologies rather than at enriching and developing an initial ontology based on emerging domain knowledge. The FCAMERGE technique [15] is much closer to ours, inasmuch it follows a bottom-up approach, extracting instances from a given set of domain-specific text documents by applying natural language processing techniques. Based on the extracted instances, classical mathematical techniques like *Formal Concept Analysis* (FCA) are then applied to derive a lattice of concepts [16] to be later manually transformed into an ontology.

The extraction of ontology classes from data items such as documents is a crucial step of all bottom-up procedures. It bypasses a typical problem of top-down ontology design, where, often at design time, there are no real objects which can be used as a basis for identifying and defining concepts. Historically, automatic knowledge extraction from text documents started by indexing documents via vectors of (normalized) keyword occurrences. This encoding gives rise to a *vector space* where every document is seen as a vector in the term space (i.e., the set of document words). Documents are then clustered into (approximations of) concepts by means of a suitable *distance* function computed on vectors, e.g., Euclidean or scalar-product-based ones. Traditional approaches to content-based clustering [17] can be classified as follows:

- *Hierarchical Algorithms*, creating a tree of node subsets by successively subdividing or merging the graph's nodes. Typical examples are k-Nearest-Neighbor (k-NN), linkage, or Ward methods.
- *Iterative Algorithms*. The simplest and most widespread algorithm is k-Means, resembling a self-organizing Kohonen network whose neighborhood function is set to size 1.
- *Metasearch Algorithms*, treating clustering as an optimization problem where a given goal is to be minimized or maximized (genetic algorithms, simulated annealing, two-phase greedy strategy, etc.).

k-Nearest-Neighbor [18], [19] is one of the most used techniques for text categorization. It is a supervised classification method: Given a set of labeled prototypes (i.e., categories) and a test document, the k-NN method finds its k nearest neighbors among the training documents. The categories of the k neighbors are used to select the nearest category for the test document: Each category gets the sum of votes of all the neighbors belonging to it and the one with the highest score is chosen. Other strategies calculate these scores taking into account the distances between the k neighbors and the test document or, alternatively, using a similarity measure like the scalar product. On the other hand, the k-Means [18] algorithm is an unsupervised technique often used in document clustering applications. Hierarchical clustering algorithms (including k-Nearest-Neighbor) are generally considered better, although slower, than k-Means [20], [21]. For this reason, *hybrid approaches* involving both k-Means and hierarchical methods have been proposed. Research work on Web document analysis [18] has shown that applying text-based classification algorithms to Web data involves three major problems. First, text-oriented techniques require a high number of documents (typically, many thousand) to work properly. Second, they hardly take into account document structure and are therefore unsuitable for semistructured document formats used on the Web, such as HTML or the *eXtensible Markup Language* (XML). Third, the final step of identifying document clusters with concepts often gives raw results that contrast with human intuition. The conceptualization step can be significantly improved only through the effort of a domain expert, which introduces a delay not compatible with community-style Web interaction. Some research approaches tried to address these problems by defining ad hoc feature spaces for heterogeneous resources classification, independently of any specific data model. Focusing on feature taxonomies, Gupta et al. [22] recently proposed a bottom-up learning procedure called TAXIND for learning taxonomies; TAXIND operates on a matrix of asymmetric similarity values. Fuhr and Weikum's CLASSIX project [23] used a feature-based technique for constructing personal or domain-specific ontologies guiding users and XML search engines in refining queries. An important contribution toward bridging the gap between conventional text-retrieval and structure-aware techniques was given by Bordogna and Pasi [24], whose work, however, does not specifically address Web and XML data. "Pure" structure-based techniques were initially proposed by one of us as the basis for approximate XML queries [25]; more recently, they were applied by Carchiolo et al. to *semantic partitioning* of individual Web documents. The schema extraction algorithm described by [26] groups the contents of a single Web page into collections which represent logical partitions of the Web page. Carchiolo et al.'s technique takes into account subtree similarity and primary tags in the XML tree for identifying these collections, using the relative and absolute depth of primary tags from the root of the tree for calculating the similarity between them. However, these techniques are aimed at partitioning single Web pages rather than at large-scale resource classification. Also, they do not address clustering inside each partition. In this paper, we develop our previous work on structure mining and knowledge extraction [27], [28] to present a way of building the specification of a shared conceptualization [29] defining concepts as clusters of concrete objects. A major aspect of our approach [30] is *trust-based* enhancement of the extracted classifications. Most current approaches use a different technique; namely, they try to adapt ontology merge algorithms to support alignment as well. For instance, the seminal PROMPT system [31] used an incremental merge algorithm to detect inconsistencies in the state of the ontology due to user updates and suggested ways to remedy these inconsistencies. In principle, a PROMPT-style approach could be adopted in conjunction with ours; however, PROMPT admittedly requires substantial user intervention and, therefore, it is not likely to scale well. We follow a different line of research, considering that metadata can be generated by different sources other than automatic extraction (the data owner, other users) and may or may not be digitally signed. As a consequence, they have nonuniform *trustworthiness* [32]. In order to take full advantage of automatically extracted metadata, it is therefore fundamental that their trustworthiness be continuously checked, e.g., based on the reported view of the user community. Trustworthiness assessment is even more important when the original source of metadata is an automatic metadata generator whose error rate, though hopefully low, is in general not negligible. Traditionally [33], [34] research approaches distinguish between two main types of trust management systems, namely, *Centralized Reputation Systems* and *Distributed Reputation Systems*.[2] In centralized reputation systems, trust information is collected from members in the community in the form of *ratings* on resources. The central authority collects all the ratings and derives a score for each resource. In a distributed reputation system, there is no central location for submitting ratings and obtaining resources' reputation scores; instead, there are distributed stores where ratings can be submitted. In a "pure" peer-to-peer setting, each peer has its own repository of trust values. In both cases, initial trust values can be modified based on users' navigation activity. Information on user behavior can be captured and transformed in a metadata layer expressing the trust degree related to the single assertion.

2. There is a clear distinction between *trust* and *reputation*. In general, the trust T of a source can be computed based on its reputation R, that is, $T = \phi(R, t)$, where t is the time elapsed since when the reputation was last modified [35].



```
A.xml

<restaurant>
   <name>
        Lions Head, The
   <\name>
   <address>
        <address>
             46 Norwich Rd Ipswich IP1 2NJ Suffolk
        <\address>
        <tel>
             01473 232266
        <\tel>
        <location>
             Town
        <\location>
   <\address>
   <owner>
        <name>
             Albert Martin
        <\name>
        <address>
             7 St Helens St Ipswich IP4 1HE Suffolk
        <\address>
   <\owner>
<\restaurant>
```

```
A.Xml – Legenda

<restaurant> = R          <address> = a
<location> = b            <name> = c
<owner> = d               <tel> = e
```

Fig. 1. The tree representation of an XML fragment.

3 A Bottom-Up Fuzzy Knowledge Extraction Engine

The *eXtensible Markup Language* (XML) [36] is today the most widespread format for exchanging and representing data-in-transit. While not all available information items can be expected to be in XML format, we extensively rely on *software wrappers* [37], a well-understood technique for mapping foreign data-types to the XML data model. Therefore, without loss of generality, here we assume data items to be represented by XML instances. We use a simplified, abstract representation of the XML Infoset data model [38] as a *multilabeled, multisorted directed graph*, where, for the sake of simplicity, only two types of nodes (element and attributes) are considered. Labels correspond to element and attribute names. This simplified Infoset can be easily extended to coincide with the standard one simply by increasing the number of sorts and adding some additional constraints [38]. Also, in our representation of the XML Infoset, the subgraph representing (element and attribute) containment is always a tree (Fig. 1), where leaf nodes represent content and values, while nonleaf nodes correspond to tags.[3] Similarity between XML documents has been defined in many different ways, taking into consideration only documents' structure, their content, or both [39]. In this paper, we shall adopt a *structure-based approach* [38] based on a fuzzy representation of XML documents' structure. Our encoding allows us to compute similarity values between XML document structures and use them to create classes representing the domain of competence. Depending on the specific domain, our structure-based clustering will be carried out in a coordinated fashion with a content-based clustering, both based on an hybrid k-Means and NN-family algorithm. For instance, content-based clustering can be applied within each structure-based similarity class (Fig. 2).

3.1 The ClassBuilder

Our approach to the architecture illustrated in Fig. 2 is a suite of software tools aimed at domain analysis experts and end users alike. In particular, our *ClassBuilder* clustering tool uses fuzzy techniques in order to compare items in an incoming XML data flow, clustering them in a

3. Note that our XML data items are well-behaved: Only the frontier of the tree contains data. While nonleaf XML nodes can, in principle, hold content, this behavior is now deprecated for all Infoset-based data representation [36].

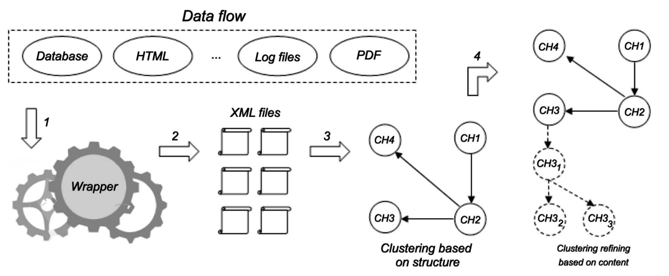


Fig. 2. Clustering an heterogeneous data flow based on structure and content information.

semiautomatic way. As we shall see, our ClassBuilder can be coupled with an auxiliary layer, called *OntoExtractor*, in order to build/update ontologies using Semantic-Web-style metadata formats. In the next sections, we will illustrate the ideas and the algorithms behind our tools.

3.2 Fuzzy Representation of XML Data

The first step of our metadata extraction process is encoding XML documents (i.e., instances of our simplified Infoset) as fuzzy multisets, i.e., *fuzzy bags*. A *bag* is simply a collection of elements which may contain duplicates. As we shall see, bags provide a good *flat encoding* for XML documents since they can express the fact that the same tag can be repeated at different levels and/or with different cardinalities at the same level. Set-based flat encodings for XML documents were first proposed in our previous work on approximate XML querying [25] as a way of avoiding carrying out computations on XML trees since tree matching problems have computational complexities ranging from polynomial to NP-complete [40]. Intuition tells us that adopting a model based on bags rather than sets will make our technique more sensitive to differences in cardinality between tags of the same name. Again, intuitively, the similarity between bags will decrease when differences in elements' cardinalities increase, but the amount of this decrease may selectively depend on the specific tags involved, fine-tuning similarity values. Our encoding of an XML tree as a fuzzy bag is computed as follows: Supposing that every tag x_i has an associated membership degree $\mu(x_i)$, initially equal to 1 ($\mu(x_i) = 1$), we divide every membership degree by its nesting level L (where $L_{\text{root}} = 1$), obtaining a "lossy" representation of the XML tree that only takes into consideration the original nesting level of tags.[4] Our flat encoding is also sensitive to structural differences between XML documents containing the same group of tags in different positions. Applying our encoding to the tree representation of the XML document A.xml (Fig. 1), we obtain the fuzzy bag:

$$\begin{aligned} A =&\{1/R, 0.5/a, 0.5/c, 0.5/d, 0.33/a, \\ &0.33/a, 0.33/b, 0.33/c, 0.33/e\}. \end{aligned}$$

We use the following representation to indicate the presence of content information connected to leaf nodes:

$$\begin{aligned} A =&\{1/R, 0.5/a, 0.5/c[data], 0.5/d, \langle 0.33, 0.33\rangle/a[data], \\ &0.33/b[data], 0.33/c[data], 0.33/e[data]\}. \end{aligned}$$

3.3 Structure-Based Document Classification

A major feature of our flat encoding technique is that, unlike XML trees, fuzzy bags lend themselves to fast and efficient clustering. When comparing fuzzy bags representing XML documents several measures of object similarity can be used [41], some of them aimed at specific domains [42]. Choosing the "right" similarity usually depends on domain and even data-set-dependent criteria.[5] Also, applying the extension principle, it is possible to extend to fuzzy bags most object comparison measures originally defined for fuzzy sets [44]. When performing this extension, it is important to note that, while the (single) occurrence of an element x in a fuzzy set has a fuzzy membership value $0 < \mu \leq 1$, the (possibly multiple) occurrences of an element x in a fuzzy bag A [45] constitute a fuzzy set whose *fuzzy cardinality* is not a single value, but a *fuzzy number of occurrences* denoted $\Omega_A(x)$ [46]. The cardinality $|\hat{A}|$ of an ordinary fuzzy set A is defined by

$$\forall n \in \mathbb{N}, \quad \mu_{|\hat{A}|}(n) = \sup\{\alpha : |A_\alpha| \geq n\}.$$

By the extension principle, the cardinality of a bag is the arithmetic sum of the occurrences of its elements, including those which appear multiple times. The cardinality of a fuzzy bag A is defined as follows:

$$|\hat{A}| = \sum_{x \in X} \Omega_A(x).$$

For example, for elements a and b in the bag $A = \{\langle 1, 0.1, 0.1\rangle/a, \langle 0.5\rangle/b\}$, we get:

$$\begin{aligned} \Omega_A(a) =&\ \{1/0, 1/1, 0.1/2, 0.1/3\}, \\ \Omega_A(b) =&\ \{1/0, 0.5/1\}, \\ |\hat{A}| =&\ \Omega_A(a) + \Omega_A(b) \\ =&\ \{1/0, 1/1, 0.5/2, 0.1/3, 0.1/4\}. \end{aligned}$$

As far as structural similarity is concerned, we need one that is monotonic with respect to element addition and removal [41], such as the *Jaccard* norm:

$$S(A, B) = \frac{|A \cap B|}{|A \cup B|}.$$

Bags are clustered using S as a similarity by means of a hybrid *k*-Means and NN clustering algorithm. Specifically, our ClassBuilder uses a tunable α threshold for the clustering process, in order to avoid suggesting an initial number of clusters (k). This way, some well-known problems of the *k*-Means algorithm are entirely avoided. Our clustering algorithm compares all items with the centroid of each cluster, considering only the top similarity value. If this value is bigger than α, the data item is inserted in the cluster; otherwise, a new empty cluster is generated

4. The content of leaf tags is contextually saved independently from the fuzzy bag representation.

5. A data-dependent technique for choosing between nonsymmetrical or symmetrical similarities can be found in [43].

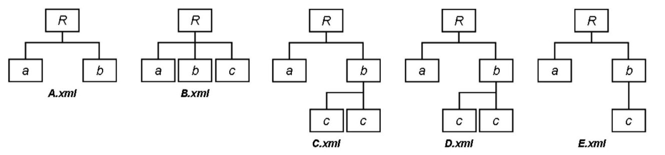


Fig. 3. Representation of XML documents in a cluster.

and the item is inserted into it. ClassBuilder offers two different ways to calculate the *centroid* of each cluster, called *clusterhead*. The former method chooses the items represented by the smallest fuzzy bag so that a cluster centroid always corresponds to a real data item. The latter generates a new fuzzy bag as the union of all fuzzy bags in the cluster. This way, a clusterhead does not necessarily coincide with a real data item since each tag is taken with the cardinality and membership degree it has in the bag where it appears with the highest cardinality. Fig. 3 shows the XML trees corresponding to the following encodings:

$$\begin{aligned} A &= \{1/R, 0.5/a, 0.5/b\},\\ B &= \{1/R, 0.5/a, 0.5/b, 0.5/c\},\\ C &= \{1/R, 0.5/a, 0.5/b, 0.3/c, 0.3/c\},\\ D &= \{1/R, 0.5/a, 0.5/b, 0.3/c, 0.3/c\},\\ E &= \{1/R, 0.5/a, 0.5/b, 0.3/c\}. \end{aligned}$$

The clusterhead is:[6]

$$CH = \{1/R, 0, 5/a, 0.5/b, 0.5/c, 0.3/c\}.$$

3.4 Content-Based Document Classification

While traditional content-based clustering operates on document-centric data, our content-based clustering technique must work on a collection of semistructured tagged documents. So, we start by computing content-based similarity *at the tag level*, comparing contents of the same tag in two different documents. Content-based similarity *at the document level* can then be obtained by aggregating tag-level similarity values. Since XML documents are encoded as fuzzy bags, our comparison works on content data attached to bag elements, whose original nesting level (in the XML tree structure) is encoded by the membership degree. Fig. 4 shows the tree representations of two structurally identical fuzzy sets A and B. Since we are evaluating content similarity, we limit our interest to subsets containing leaf nodes only (i.e., nodes where content information can be found). With $\{x[data]\}_D$, we designate content data delimited by tag x in document D. In order to evaluate content similarity, we combine two different functions. The first one is a tag-level similarity f comparing data belonging to tags with the same name in different documents:

$$f_x(\{x[data]\}_{D_1}, \{x[data]\}_{D_2}).$$

6. After this clustering process, if new documents join the same data flow, in order to cluster them, it is sufficient to compare their distance with respect to final centroids obtained previously and insert them in the cluster represented by its closest centroid. This is the behavior of NN-family algorithms.

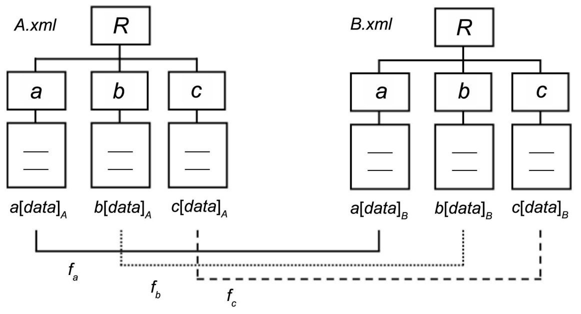


Fig. 4. Aggregation between data belonging to the same tag in different documents.

Referring to Fig. 4, we compute:

$$\begin{aligned} &f_a(\{a[data]\}_A, \{a[data]\}_B),\\ &f_b(\{b[data]\}_A, \{b[data]\}_B),\\ &f_c(\{c[data]\}_A, \{c[data]\}_B). \end{aligned}$$

Since terms may have different informative value depending on the tag they belong to, we also take into account the membership value $\mu(x_i)$ associated to the ith tag. When dealing with text data, we represent the content of each tag by the well-known *Vector Space Model* (VSM), which represents natural language documents as vectors in a multidimensional space: Every document d is considered to be a vector $\boldsymbol{d}$ in the term space (set of document words). Each document is represented by the (*TF*) vector $d_{f_t} = (f_{t_1}, f_{t_2}, \ldots, f_{t_n})$, where f_{t_i} is the frequency of the ith term in the document. Then, we can use a standard *cosine distance*—expressed by the equation $\cos(\boldsymbol{d}_1, \boldsymbol{d}_2) = (\boldsymbol{d}_1 \bullet \boldsymbol{d}_2)/\|\boldsymbol{d}_1\| \cdot \|\boldsymbol{d}_2\|$, where $\bullet$ denotes the scalar product and $\|\boldsymbol{d}\|$ is the *Euclidean Length*. However, text comparison hardly captures the real similarity between typed values such as dates and numerical types. Our XML content can belong to one of the basic XML Schema data-types (URLs, numbers, dates, etc.) or be a text blob. In the first case, it is useful to apply *type-driven matching*, i.e., compute data similarity by taking into account their estimated XML Schema type T. When type information is available, we use ad hoc measures for computing similarity between basic datatypes, as illustrated in Example 1. ClassBuilder supports a number of string distances [47] like the *Levenshtein* (or *edit*) *distance* L, defined for two strings s and r of arbitrary length as the minimum number of character inserts, deletes, and changes needed to convert r into s.[7]

The second function F aggregates the individual f_x values, $F(f_a, f_b, f_c)$ and expresses the overall similarity between two XML documents. Using F as a comparison measure, we can perform content-based clustering using the same algorithm used for structural clustering in Section 3.3. When choosing F, we need to take into consideration the tags' different semantic relevance. At one extreme, there is the situation in which *all* tag-level similarity values contribute to the overall similarity via a conjunction of their individual values. Here, F will belong

7. For string and URI/date distances, more details are given in [47] and [48].

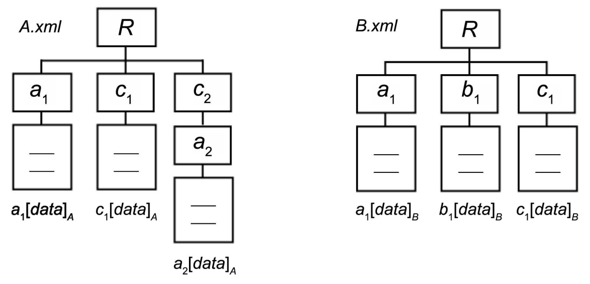


Fig. 5. Computing similarity between documents featuring multiple tags with the same name and in different positions.

to the t-norm [49], [50] class of operators. At the other extreme lies the situation in which the overall result considers *exactly one* of the individual similarity values, operating a disjunction via a t-conorm. The aggregation depicted in Fig. 4 represents the simplest possible situation, when two documents have exactly the same tags (with no duplicates) and the same membership degrees (the same positions). In order to deal with the general case, we need to answer two additional questions not addressed by classical techniques of information clustering and retrieval (see Fig. 5). Namely, 1) what happens when a tag is present in one of the documents being compared and not in the other and 2) what are the "right" tag-pairs to be compared?[8] When a tag appears in a document (with single or multiple cardinality) and not in the other, we simply consider tag-level similarity to be zero. Referring to Fig. 5:

$$f_b(null, b[data]_B) = 0.$$

Question 2 is a bit more tricky, because tag x may appear at different nesting levels in the two documents. In this case, we consider the maximum value resulting from the comparison of all pairs of content data connected to it within the two documents under comparison. Namely, we get:

$$f_x = \max_{p,k} \frac{1}{1+\Delta_{p,k}} f_{x_{p,k}}(\{x_p[data]\}_{D_1}, \{x_k[data]\}_{D_2}),$$

where p ranges over the occurrences of tag x in D_1 while k ranges over the occurrences of x in D_2 in any suitable order. Δ represents the difference between the tag's occurrences membership degrees, that is, $\Delta = |\mu(x_p) - \mu(x_k)|$. In this way, in the case of multiple tags with the same name at the same level, we take into consideration the maximum value of f resulting from all comparisons, while, when the same tag is repeated multiple times in the two documents but at different levels, we weight f by the difference between membership values. In other words, we take into account the intuitive fact that two identical data items located at different nesting levels are less likely to have the same meaning.

Example 1. For the sake of simplicity, in this example, we assume data connected to leaf tags to be of `string` type, and we use the *N-gram distance N* [51], that compares the N-grams from the first word with the N-grams for the second word. An N-gram is a vector representation including all of the N-letter combinations in a string. The N-grams comparison function generates the set of all substrings of length n for each string. Referring to Fig. 5, supposing

$$a_1[data]_A = \clubsuit\clubsuit\clubsuit\clubsuit\clubsuit, a_2[data]_A = \clubsuit\clubsuit\diamondsuit\clubsuit\clubsuit,$$
$$c_1[data]_A = \heartsuit\heartsuit\spadesuit\spadesuit\spadesuit, a_1[data]_B = \clubsuit\clubsuit\diamondsuit\diamondsuit\clubsuit,$$
$$b_1[data]_B = \diamondsuit\diamondsuit\diamondsuit\diamondsuit\diamondsuit, c_1[data]_B = \heartsuit\heartsuit\heartsuit\spadesuit\spadesuit,$$

and using 3-grams, we obtain:

$$\begin{aligned}
f_{a_{1,1}}(\{a_1[data]\}_A, \{a_1[data]\}_B) &= N(\clubsuit\clubsuit\clubsuit\clubsuit\clubsuit, \clubsuit\clubsuit\diamondsuit\diamondsuit\clubsuit) \\
&= 0.44, \\
f_{a_{2,1}}(\{a_2[data]\}_A, \{a_1[data]\}_B) &= N(\clubsuit\clubsuit\diamondsuit\clubsuit\clubsuit, \clubsuit\clubsuit\diamondsuit\diamondsuit\clubsuit) \\
&= 0.85, \\
f_{b_{null,1}}(null, \{c_1[data]\}_B) &= 0, \\
f_{c_{1,1}}(\{c_1[data]\}_A, \{c_1[data]\}_B) &= N(\heartsuit\heartsuit\spadesuit\spadesuit\spadesuit, \heartsuit\heartsuit\heartsuit\spadesuit\spadesuit) \\
&= 0.89.
\end{aligned}$$

It follows that

$$f_a = \max\left(0.44, \frac{1}{1+|0.33-0.5|} \cdot 0.85\right) = \max(0.44, 0.73)$$
$$= 0.73, f_b = 0 \text{ and } f_c = 0.89.$$

In order to obtain the final similarity value between documents, we apply the aggregator F to the individual fs. Using the arithmetic average, we get:[9]

$$F(f_a, f_b, f_c) = \frac{0.73 + 0 + 0.89}{3} = 0.54.$$

4 Metadata Extraction: A Worked-Out Example

In this section, we apply our technique to a set of documents taken from a real application about restaurants. As we have seen, our clustering method consists of two independent phases: a structural and a content-based one. While there is no preset order for executing them, in this example, content analysis follows the structural one.

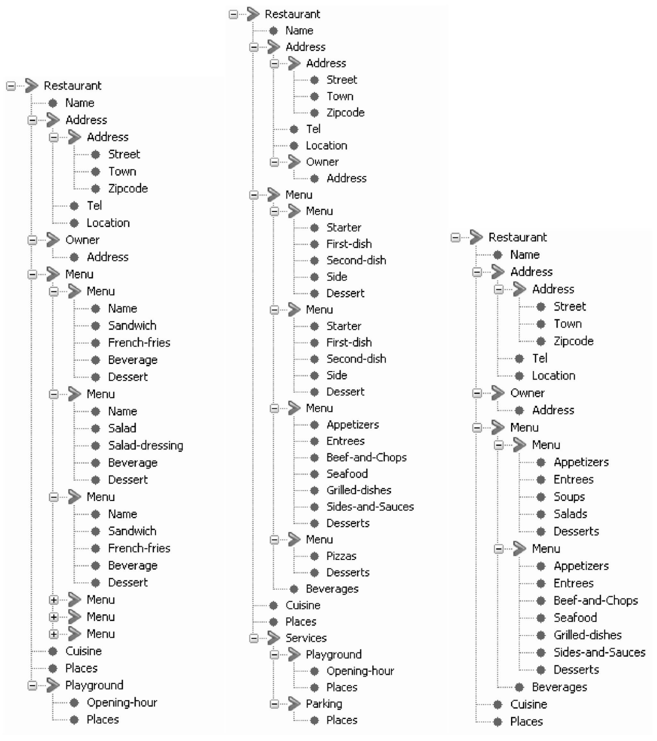
4.1 Structure Analysis

We rely on the educated guess that categories of restaurants may be identified based on the number and the type of menus they propose (and, consequently, on their different cardinality and/or structure). Let us examine the clusterheads obtained by the procedure explained in Section 3.3.[10] The "typical" structure referring to a *Fast Food* is characterized by a "high" number of different *Menu*s more or less with the same structure. Fig. 6a shows the structure of this clusterhead with our ClassBuilder's layout. Much in the same way, a *Tourist Restaurant* usually offers one or a

8. Due to the semistructured nature of XML data, there may be multiple tags with the same name within both documents at the same or at a different nesting level.

9. It is important to underline that, had we used traditional text-based encoding, we would have obtained a much lower similarity value, namely, $N(\clubsuit\clubsuit\clubsuit\clubsuit\clubsuit\ \heartsuit\heartsuit\spadesuit\spadesuit\spadesuit\ \clubsuit\clubsuit\diamondsuit\clubsuit\clubsuit,\ \clubsuit\clubsuit\diamondsuit\diamondsuit\clubsuit\ \diamondsuit\diamondsuit\diamondsuit\diamondsuit\diamondsuit\ \heartsuit\heartsuit\heartsuit\spadesuit\spadesuit) = 0.21$.

10. Of course, the names for the clusterheads have been picked manually. Our ClassBuilder does not attempt to give names to clusters automatically.



(a) (b) (c)

Fig. 6. The clusterheads: (a) fast food, (b) tourist restaurant, and (c) fine restaurant.

couple of predefined *Special Menu*s with the same structure, {*Starter, First course, Second course, Side, Dessert*}, plus the possibility of choosing among the dishes of a classic *à la carte Menu* or a simple *Pizza Menu* (Fig. 6b). Finally, a *Fine Restaurant* shares with a Tourist Restaurant the same structure for the *à la carte Menu*, characterized by the structure {*Appetizers, Entrees, Steaks-and-Chops, Seafood, Grilled-dishes, Sides-and-Sauces, Desserts*}; it usually does not contain predefined menus, except when we can recognize in the {*Appetizers, Entrees, Soups, Salads, Desserts*} structure a classical *Vegetarian Menu* (Fig. 6c).

4.2 Content Analysis

After the structural analysis phase, we run ClassBuilder again to subdivide each structure cluster in terms of one or more elements' content. Depending on the type of the chosen element(s), it is necessary to choose a suitable distance. Finer-grained clustering leads to the subdivision of each structural cluster in a number of content-based clusters, each containing documents where designated elements take values in a given value range; once again, depending on the type of the chosen feature, ClassBuilder will extract a *cluster descriptor* encoding the value range for each content-based cluster. Our tool will express content-based cluster descriptors in the form of XML `ComplexTypes`, later to be used as a basis for defining Semantic-Web-style metadata. This is rather straightforward in the case of numeric values (for example, the number of places in a restaurant expressed by the tag `<Places>`), where the standard XML Schema `range` type restriction can be used. Extracting a concise descriptor for a content-based cluster whose feature type can assume several values (e.g., the tag `<Cuisine>`) or a zipcode type (the tag `<Zipcode>`) is far less simple. In the first case, we can solve the problem using an `enum` type restriction as descriptor in the second case, the descriptor is obtained as an instance of Semantic-Web-style metadata formats by querying a suitable remote agent [52]. These techniques will be described in detail in Section 5.2.

5 Ontology Construction Process

We are now ready to organize extracted information in a standard knowledge representation format. *OntoExtractor* is an additional layer of our system, fully integrated with the ClassBuilder tool. It takes as inputs the clusterheads resulting from the overall clustering process and organizes them into an ontology using a standard Semantic Web format. This ontology is then used to complement and develop an existing upper-level domain ontology. While traditional ontology learning approaches focus on hyperonymic and association relations,[11] our technique takes advantage of structural information, including the elements' cardinality. In particular, our approach is aimed at detecting four ontological relations:

1. `subclass-of`,
2. `value restriction`,
3. `cardinality restriction`, and
4. `associated-to`.

These relations are typical of a number of knowledge representation languages, and are usually denoted via *Description Logic* (DL) formalism used for setting up the Semantic Web knowledge representation standards. In DL, the semantics of ontological relations is defined by means of an extensional definition, i.e., evaluating instances membership of classes. For instance, a class A is a subclass of a class B if all the instances belonging to A also belong to B. In our application, individual documents' membership cannot be used because no document can be classified under different clusters (Section 3). For this reason, the notion of subsumption used here is *intentional*: The way for checking relations among classes is to compare clusterheads.[12] Let us give an informal description of our procedure before illustrating it in detail. The basic idea we rely on for identifying candidate classes is organizing clusterheads (such as, in our example, *Tourist Restaurant* or *Fine Restaurant*) in a hierarchy. A classic technique used to build the lattice of binary relations over a pair of sets is *Formal Concept Analysis* (FCA). Starting from the structure of our clusterheads in terms of attributes, our OntoExtractor works in three steps:

1. First, it uses an extension of the classic FCA algorithm to build the lattice of clusterheads intersections; in turn, these intersections give rise to

11. This is due to the fact that available techniques for knowledge acquisition are mainly based on semantic distances.

12. Of course, here we once again use our representation of clusterheads in terms of fuzzy bags for performing comparisons and matching operations. For the sake of conciseness, we shall loosely use the term "clusterhead" rather than "fuzzy bag encoding of a clusterhead" whenever no confusion may arise.

ontology classes connected by `subclass-of` relations.

2. Second, class attributes are used in order to generate `part-of` relations, whose data type is defined on the basis of the content-based clustering results.
3. Third, `value restriction` relations are added to the hierarchy, on the basis of the blocks composing the content-based partition of each cluster.[13]

The final phase of our process is generating standard metadata assertions linking resources and concepts. XML documents belonging to the same cluster are associated to the corresponding ontology class and to all its superclasses, according to the `subclass-of` hierarchy. This association is not a Boolean predicate; rather, it is a matter of degree inasmuch as in computing a similarity matching among the fuzzy bags representation of documents and concepts, we obtain a measure of the level of fulfillment between a class and a document. This value can be interpreted as a degree of reliability of our document classification and is used for setting an initial trust value for our *Trust Layer* (Section 6).

A Fuzzy Extension to FCA. FCA is a time-honored mathematical theory designed for identifying conceptual structures based on data sets [16]. FCA techniques are based on the notion of a formal context $\mathcal{K}$, defined as a triple $(\mathcal{D}, \mathcal{A}, \mathcal{R})$, where $\mathcal{D}$ represents the data items (called *documents* in the FCA terminology), $\mathcal{A}$ the attributes, and $\mathcal{R}$ is a binary relation $\mathcal{R} \subseteq \mathcal{D} \times \mathcal{A}$. A concept lattice can be obtained evaluating two functions: f and g. Function f maps a set of documents into the set of common attributes, whereas g does the same for attribute sets. If X and Y are, respectively, sets of documents and attributes, a *formal concept* is a couple (X, Y), where $X = g(Y)$ and $Y = f(X)$. X is called the *extent* and Y the *intent* of the concept. These definitions allow us to build a lattice organizing, in a partial order, all formal concepts belonging to a formal context. Here, we extend FCA to handle our encoding of XML documents as fuzzy bags. Our extended notion of formal context is a triple $\mathcal{K}_{ex} = (\mathcal{D}, \mathcal{A}, \mathcal{R}_{ex})$, where $\mathcal{D}$ and $\mathcal{A}$ are sets and $\mathcal{R}_{ex}$ is a fuzzy bag on domain $\mathcal{D} \times \mathcal{A}$, where each relation $(d, a) \in \mathcal{R}_{ex}$ is a fuzzy integer $\Omega_d(a)$. In classic FCA, the codomain of the function f, i.e., the set of attributes common to a given set of documents X, is equivalent to the intersection of the documents in X and identifies a set of common attributes Y. In our extended FCA, the function f_{ex} returns the intersection among the fuzzy bags associated to the documents in X. The fuzzy bag obtained by this intersection must be included in all the fuzzy bags of X. According to [53], (1) gives us the definition of intersection among two fuzzy bags using fuzzy integers

$$\Omega_{A \cap B}(x) = \min(\Omega_A(x), \Omega_B(x)). \tag{1}$$

Because *min* is an associative operator, we can apply it to a set of attributes. In other words, function f_{ex} maps a set of documents to the fuzzy bag generated by the intersection of all the fuzzy bags of X. For each attribute $a \in Y$, the fuzzy integer of a is computed according to (2):

13. Note that, taking advantage of our representation expressed in terms of fuzzy bags, `cardinality restriction` relations are also generated if clusterheads members of the same candidate class show relevant differences in cardinality.

TABLE 1
The Context of Our Clusterheads

	R	A	M	O	P	FM	AM	SM	PM	VM
CH_1	1/1	0.7/1	0.7/1	0.7/1	0.7/1	0.58/6	0	0	0	0
CH_2	1/1	0.7/1	0.7/1	0.58/1	0.58/1	0	0.58/1	0.58/2	0.58/1	0
CH_3	1/1	0.7/1	0.7/1	0.7/1	0	0	0.58/1	0	0	0.58/1

$$\Omega_{f_{ex}(X)}(a) = \min_{i \in X} \Omega_i(a). \tag{2}$$

Similarly, function g_{ex} maps a fuzzy number $\Omega_{g_{ex}(Y)}$ generated by the intersection of the fuzzy bags composed of a set of attributes Y to all the documents described by a fuzzy bag including $\Omega_{g_{ex}(Y)}$

$$\Omega_{g_{ex}(Y)}(a) = \min_{i \in Y} \Omega_i(a). \tag{3}$$

5.1 Constructing the Concept Hierarchy

We are now ready to generate our bottom-up ontology and to use it to complement an existing one built in the usual top-down fashion. As a reference, we will use the well-known upper-level restaurant ontology developed by the Imperial College (UK) [54]. Our examples refer to the clusterheads in Fig. 6 that can be linked to the upper-level ontology as subclass of the generic class *Restaurant*. Of course, terminological differences can be among some classes or properties of the upper-level ontology and the tags in our XML files. For instance, in the restaurant ontology, we have a class *BabyLaundry* where, in our documents, we have a property *Playground*. These differences must be consolidated using well-known conceptual distance methods [55]. To illustrate what happens when our OntoExtractor is used after ClassBuilder with no human intervention, here we use machine-attributed cluster names like CH_1 for Fast Food, CH_2 for Tourist Restaurant, and CH_3 for Fine Restaurant and the fuzzy bag encoding explained in Section 3.2, with the difference that, in this case, we divide every tag's initial membership degree (equal to 1) by $\sqrt{L}$, i.e., the square root of its nesting level.

$$\begin{aligned}
CH_1 = \{&1/R, 0.7/Address, 0.7/Menu,\\
&0.7/Owner, 0.7/Playground,\\
&\langle 0.58, 0.58, 0.58, 0.58, 0.58, 0.58\rangle/Fast\text{-}food\text{-}Menu\},\\
CH_2 = \{&1/R, 0.7/Address, 0.7/Menu, 0.58/Owner,\\
&0.58/Playground, \langle 0.58, 0.58\rangle/Set\text{-}meal\text{-}Menu,\\
&0.58/A\text{-}la\text{-}carte\text{-}Menu, 0.58/Pizza\text{-}Menu\},\\
CH_3 = \{&1/R, 0.7/Address, 0.7/Menu, 0.7/Owner,\\
&0.58/A\text{-}la\text{-}carte\text{-}Menu, 0.58/Vegetarian\text{-}Menu\}.
\end{aligned}$$

Table 1 shows the context formed by our clusterheads. For the sake of conciseness, fuzzy bags' elements are denoted only by the initial letters of the element name; also, some attributes are omitted.

Applying f_{ex} and g_{ex} to our clusterheads, we obtain the hierarchy shown in Fig. 7. This hierarchy identifies a concept c_1 as the subset of attributes common to all the clusterheads, while c_2 groups restaurants having playground facilities.

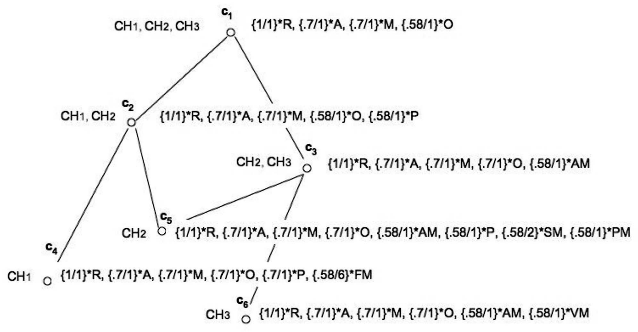

c1
CH1, CH2, CH3
{1/1}*R, {.7/1}*A, {.7/1}*M, {.58/1}*O
c2
CH1, CH2
{1/1}*R, {.7/1}*A, {.7/1}*M, {.58/1}*O, {.58/1}*P
c3
CH2, CH3
{1/1}*R, {.7/1}*A, {.7/1}*M, {.7/1}*O, {.58/1}*AM
c5
CH2
{1/1}*R, {.7/1}*A, {.7/1}*M, {.7/1}*O, {.58/1}*AM, {.58/1}*P, {.58/2}*SM, {.58/1}*PM
c4
CH1
{1/1}*R, {.7/1}*A, {.7/1}*M, {.7/1}*O, {.7/1}*P, {.58/6}*FM
c6
CH3
{1/1}*R, {.7/1}*A, {.7/1}*M, {.7/1}*O, {.58/1}*AM, {.58/1}*VM

Fig. 7. Classes organized in a hierarchy.

Also, c_3 includes restaurants allowing ordering "à la carte," while c_4, c_5, and c_6 are formal concepts associated to a single clusterhead. While this hierarchy could have been obtained by standard FCA, our extension enriches it with additional information about the relevance and cardinality of attributes by means of fuzzy integers. This information will support generating metadata assertions that associate hierarchy concepts to XML documents according to a trust degree, as explained in Section 5.4.

5.2 Generating the Ontology

In order to translate our hierarchy into OWL standard knowledge representation format, we start by taking each concept as a candidate class connected by a `subclass-of` relation to all concepts included in it. Then, our translation algorithm visits the hierarchy top-down, adding to each class the properties it has not inherited from its super classes. Properties' data types are decided based on the results of content-based clustering. Namely, whenever the content of a tag allows it, our algorithm defines a value range restriction on the corresponding property by declaring an XML `ComplexType` and using it as a basis for a class declaration in the OWL code.

In the case of our example, we obtain different range classes for describing values of the `<Places>` tag in the different documents of a cluster. Each time the algorithm generates a *Places*, attribute it will set its type to the appropriate range class. In Fig. 8, two code fragments are shown: (a) an example of an interval type defined by means of XML and (b) an OWL property defined using this data type. In other cases, the data type can be defined using a standard knowledge representation, such as the ISO 3166 geographical taxonomy [56].

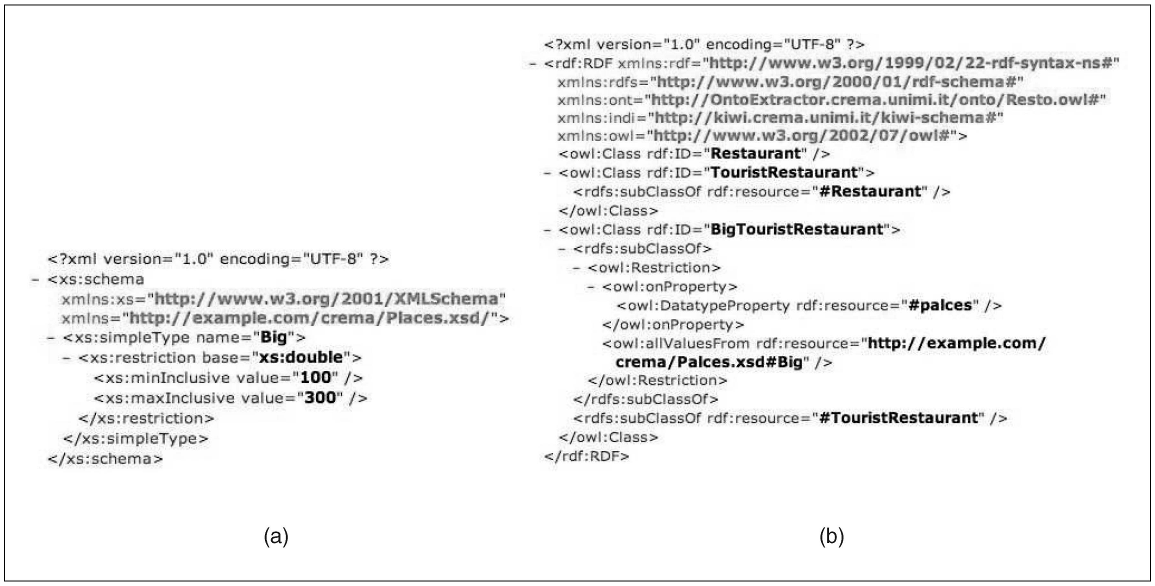
When the content of a tag does not readily generate range subdivisions, e.g., because of high heterogeneity of data values, we simply set a data type property accepting any type of character data. Fig. 8 shows some examples of OWL classes and properties generated by our OntoExtractor. These class and property names used by our tool are taken from the original restaurant ontology [54], making integration and comparison straightforward.

5.3 Cardinality Restrictions

A major feature of our approach is taking advantage of cardinality information in order to define *cardinality restrictions* on the ontology. For instance, let us consider a clusterhead with the form:

$$\begin{aligned}CH_4 = \{&1/R, \langle 0.7, 0.7, 0.7, 0.7, 0.7\rangle/Address, 0.7/Menu,\\ &0.7/Owner, \langle 0.7, 0.7, 0.7\rangle/Playground,\\ &\langle 0.58, 0.58, 0.58, 0.58, 0.58, 0.58, 0.58,\\ &0.58, 0.58, 0.58\rangle/Fast\text{-}food\text{-}Menu\}.\end{aligned}$$

This clusterhead has the same attributes as CH_1 and the FCA algorithm will group both in the same formal concept. On the other hand, CH_4 shows some differences in the cardinality of the attributes *Playground* and *Address*. For this reason, we generate a specific `cardinality restriction`, expressing this difference quantitatively by means of fuzzy integers. In ordinary arithmetics, two numbers are equal if, by dividing one by the other, we get



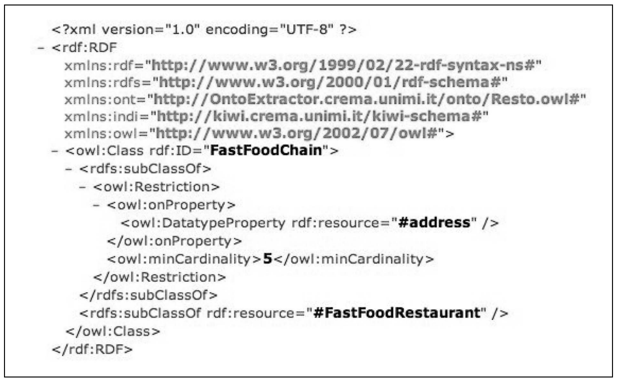
```
<?xml version="1.0" encoding="UTF-8" ?>
- <xs:schema
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns="http://example.com/crema/Places.xsd/">
  - <xs:simpleType name="Big">
    - <xs:restriction base="xs:double">
        <xs:minInclusive value="100" />
        <xs:maxInclusive value="300" />
      </xs:restriction>
    </xs:simpleType>
  </xs:schema>
```

(a)

```
<?xml version="1.0" encoding="UTF-8" ?>
- <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:ont="http://OntoExtractor.crema.unimi.it/onto/Resto.owl#"
    xmlns:indi="http://kiwi.crema.unimi.it/kiwi-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#">
    <owl:Class rdf:ID="Restaurant" />
  - <owl:Class rdf:ID="TouristRestaurant">
      <rdfs:subClassOf rdf:resource="#Restaurant" />
    </owl:Class>
  - <owl:Class rdf:ID="BigTouristRestaurant">
    - <rdfs:subClassOf>
      - <owl:Restriction>
        - <owl:onProperty>
            <owl:DatatypeProperty rdf:resource="#palces" />
          </owl:onProperty>
          <owl:allValuesFrom rdf:resource="http://example.com/
            crema/Palces.xsd#Big" />
        </owl:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf rdf:resource="#TouristRestaurant" />
    </owl:Class>
  </rdf:RDF>
```

(b)

Fig. 8. (a) A customized data type and (b) an OWL property defined using the same data type.



```
<?xml version="1.0" encoding="UTF-8" ?>
- <rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:ont="http://OntoExtractor.crema.unimi.it/onto/Resto.owl#"
    xmlns:indi="http://kiwi.crema.unimi.it/kiwi-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#">
  - <owl:Class rdf:ID="FastFoodChain">
    - <rdfs:subClassOf>
      - <owl:Restriction>
        - <owl:onProperty>
            <owl:DatatypeProperty rdf:resource="#address" />
          </owl:onProperty>
          <owl:minCardinality>5</owl:minCardinality>
        </owl:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf rdf:resource="#FastFoodRestaurant" />
    </owl:Class>
  </rdf:RDF>
```

Fig. 9. An example of cardinality restriction.

a result equal to 1. The same principle can be applied to fuzzy integers: By dividing a fuzzy integer by another [57], we obtain a quotient that is again a fuzzy quantity. By the extension principle, the result of a division among fuzzy integers is a rational fuzzy number. This fuzzy quantity can be represented by a scalar number applying a σ-*count* function [58]. Comparing this scalar result to 1, we obtain a measure of the difference between two fuzzy integers as the complement of the division result obtained in the division. Formally, we have:

$$1 - \sigma\text{-}count\left(\frac{\Omega_i}{\Omega_{f_{ex}(X)}}\right), \tag{4}$$

where i is a document belonging to a given formal concept: $i \in f_{ex}(X)$. In our example, the formal concept c_4 has an extent composed by the documents CH_1 and CH_4. Dividing the intent of c_4 by the intent of CH_4, we obtain the following result:

$$\begin{aligned}1 - \sigma\text{-}count\left(\frac{\Omega_{c_4}}{\Omega_{CH_4}}\right) &= 1 - \sigma\text{-}count\left(\frac{\{1/1, 0.7/5, 0.58/11\}}{\{1/1, 0.7/11, 0.58/21\}}\right)\\ &= 1 - \sigma\text{-}count(\{1/1, 0.7/0.4, 0.58/0.52\})\\ &= 1 - 0.64 = 0.36.\end{aligned}$$

In other words, CH_4 cardinality difference has a degree of 0.36 with respect to c_4. Based on this value, OntoExtractor suggests the generation of a `cardinality restriction`. Fig. 9 shows a cardinality restriction produced by our system.

5.4 Generating Metadata Assertions

After the ontology has been extracted, we generate other assertions that are essentially links between XML documents classified in a clusterhead and the corresponding ontology class. For instance, a document D belonging to the CH_2 will be associated to the class c_5 and, therefore, to c_2, c_3, and c_1. Note that the fulfillment degree of these associations is not uniform. In general, membership values of fuzzy bags associated to classes at the top of the hierarchy are lower than in the ones at the bottom. This is due to the fact that the intersection among fuzzy bags is computed by means of the *min* operator. For instance, documents D_1 and D_2, belonging to CH_2, are encoded by the following fuzzy bags:

TABLE 2
Class/Document Matching

	c_5	c_2	c_3	c_1
D_1	0.82	0.83	0.83	0.75
D_2	0.88	0.77	0.71	0.71

$$\begin{aligned}D_1 = \{&1/R, 0.7/Address, 0.7/Menu, 0.58/Owner,\\ &0.58/Playground, 0.58/Set\text{-}meal\text{-}Menu,\\ &0.58/A\text{-}la\text{-}carte\text{-}Menu\},\\ D_2 = \{&1/R, 0.7/Address, 0.7/Menu, 0.58/Owner,\\ &0.58/Playground, \langle 0.58, 0.58\rangle/Set\text{-}meal\text{-}Menu,\\ &0.58/Pizza\text{-}Menu\}.\end{aligned}$$

Using the similarity described in Section 3.3, we can evaluate how much the encoding of each document matches the encoding of each class, as follows:

$$\begin{aligned}&\sigma\text{-}count\Bigg(S(D_1, CH_2) = \frac{D_1 \cap CH_2}{D_1 \cup CH_2}\\ &\quad = \frac{\{1/1, 0.7/3, 0.58/7\}}{\{1/1, 0.7/3, 0.58/9\}} = \{1/1, 0.7/1, 0.58/0.7\}\Bigg) = 0.82,\\ &\sigma\text{-}count\Bigg(S(D_2, CH_2) = \frac{D_2 \cap CH_2}{D_2 \cup CH_2}\\ &\quad = \frac{\{1/1, 0.7/3, 0.58/8\}}{\{1/1, 0.7/3, 0.58/9\}} = \{1/1, 0.7/1, 0.58/0.8\}\Bigg) = 0.88.\end{aligned}$$

These values are computed for all the classes connected to D_1 and D_2, as shown in Table 2.

Each `rdf:Description` element in Fig. 10 refers to a resource by means of the `rdf:about` property, as well as to a class present in the domain ontology.

6 Incorporating Trust-Based Feedback

For validating automatically generated assertions, we do not rely on human inspection as it would require an effort comparable to manually writing metadata from scratch. Rather, we leave it to community members to express their views on the trustworthiness of each assertion. While interacting with documents, users can provide important feedback on metadata trustworthiness. This information is captured and transformed in a new metadata layer composed of *trust assertions* expressing the level of trust of the assertions of the first layer.[14] This second layer can be computed by a central server or by distributed clients; in both cases, the trust degree specified by each trust assertion must be aggregated and the result provided to all interested clients. On the basis of this aggregated trust degree, clients can compute their *custom views* on the original metadata base using a movable threshold to discard untrusted assertions.

14. Although we shall not deal with digital signatures in this paper, it is important to remark that meta-assertions could be enriched with a *digital signature*, providing a way of indicating clearly and unambiguously their author.

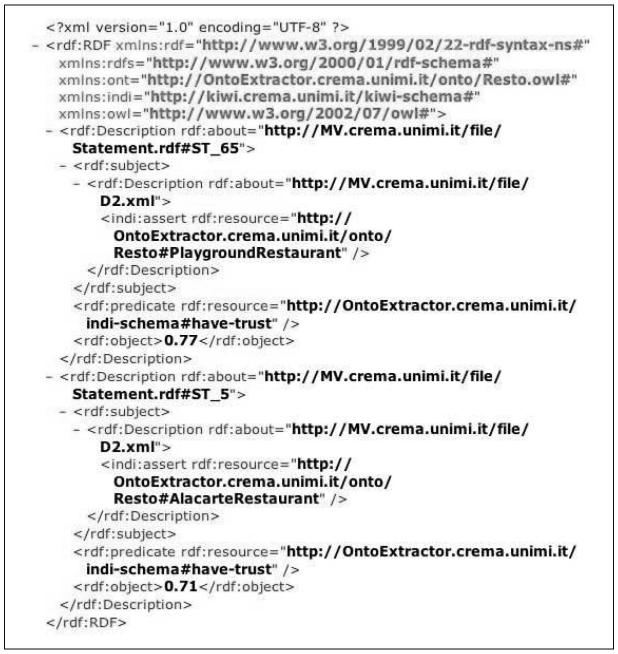

```
<?xml version="1.0" encoding="UTF-8" ?>
- <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:ont="http://OntoExtractor.crema.unimi.it/onto/Resto.owl#"
    xmlns:indi="http://kiwi.crema.unimi.it/kiwi-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#">
  - <rdf:Description rdf:about="http://MV.crema.unimi.it/file/
      Statement.rdf#ST_65">
    - <rdf:subject>
      - <rdf:Description rdf:about="http://MV.crema.unimi.it/file/
          D2.xml">
          <indi:assert rdf:resource="http://
            OntoExtractor.crema.unimi.it/onto/
            Resto#PlaygroundRestaurant" />
        </rdf:Description>
      </rdf:subject>
      <rdf:predicate rdf:resource="http://OntoExtractor.crema.unimi.it/
        indi-schema#have-trust" />
      <rdf:object>0.77</rdf:object>
    </rdf:Description>
  - <rdf:Description rdf:about="http://MV.crema.unimi.it/file/
      Statement.rdf#ST_5">
    - <rdf:subject>
      - <rdf:Description rdf:about="http://MV.crema.unimi.it/file/
          D2.xml">
          <indi:assert rdf:resource="http://
            OntoExtractor.crema.unimi.it/onto/
            Resto#AlacarteRestaurant" />
        </rdf:Description>
      </rdf:subject>
      <rdf:predicate rdf:resource="http://OntoExtractor.crema.unimi.it/
        indi-schema#have-trust" />
      <rdf:object>0.71</rdf:object>
    </rdf:Description>
  </rdf:RDF>
```

Fig. 10. The metadata format.

The architecture of our *Trust Layer* (Fig. 11) is composed of a centralized *Publication Center* that collects and displays metadata assertions manually added by the ontology engineer or produced by the OntoExtractor.[15] Clients interact with assertions by navigating them and providing implicitly (with their behavior) or explicitly (by means of an explicit vote) an evaluation about their trustworthiness. Trust-related information is passed by the Publication Center on to the *Trust Manager* in the form of new assertions of a special type. *Trust assertions* are built using the well-known technique of *reification*.[16] Then, trust values are made available to all interested parties and clients can use them to custom compute views on the original metadata (e.g., filtering out all metadata whose trust value lies below a threshold). Our trust ratings are computed either by collecting and aggregating the users' explicit votes or by nonintrusively inferring users preferences from their behavior (e.g., by monitoring the time spent by each user working on a document). Here, we focus on the former technique, where users can cast a vote on each assertion's trustworthiness. We assume that only a small subset of the users will vote, depending on their role or their expertise. For aggregating these votes, it is necessary to take into account the level of anonymity provided by the system. If all users are anonymous, all votes contribute in the same way to the overall trust value on metadata assertions about a resource. When users have an identity, ratings have to be aggregated at the level of the individual user first, and then globally.

15. Of course, our Center will assign different trust values to assertions depending on their origin: Assertions written by a domain expert are much more reliable than the automatically generated ones submitted by OntoExtractor.

16. This choice allows our Trust Layer to interact with other sources of metadata than OntoExtractor because the Trust Metadata syntax is *not dependent* on the format of the original assertions.

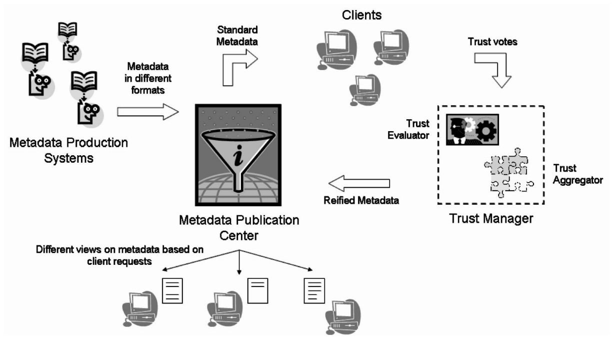

Standard Metadata
Clients
Trust votes
Metadata in different formats
Metadata Production Systems
Trust Evaluator
Trust Aggregator
Reified Metadata
Trust Manager
Metadata Publication Center
Different views on metadata based on client requests

Fig. 11. The Trust Layer architecture.

6.1 The Reputation Computation Problem

In the last few years, several techniques for computing reputation and trust measures in nonanonymous environments have been proposed [34]:

- *Summation or average of ratings*, computing the sum of positive and negative ratings separately. The total score is obtained by subtracting negative votes from positive ones.
- *Bayesian Systems*, taking binary ratings as input and computing reputation scores by statistically updating *beta probability density functions*. They are particulary used in *recommender systems*[59].
- *Discrete Trust Models*, relying on human verbal statements to rate assertions (e.g., *Very Trustworthy*, *Trustworthy*, *Untrustworthy*, *Very Untrustworthy*).
- *Belief Models*. Belief theory is a framework related to probability theory, but where the sum of beliefs over all possible outcomes does not necessarily add up to 1. The missing part is interpreted as uncertainty.
- *Fuzzy Models*, where linguistic variables are used to represent trust; in this case, membership functions describe to what degree an assertion can be described as trustworthy or not trustworthy.
- *Flow Models*, computing trust by transitive iteration through looped or arbitrarily long chains.

Our approach computes the *level of trust* of an assertion, that we indicate with p_M, as the aggregation of multiple *fuzzy values* p_{M_i} representing human interactions with metadata assertions. Namely, we set $p_M = \mathcal{A}(p_{M_i})$, where $\mathcal{A}$ is an aggregation operator [60]. The initial value of each p_{M_i} is assigned automatically or semiautomatically during the process of metadata creation.[17] Once assertions have been generated, users may cast 1) *explicit* or 2) *implicit* votes on them, as follows:

$$1)\ \hat{p}_{M_i} = (p_{M_i})^{\delta} \text{ and } 2)\ \check{p}_{M_i} = (p_{M_i})^{\frac{1}{\delta}},$$

where $0.1 \leq \delta < 1$ (a smaller value of δ produces a higher difference between explicit and implicit votes and vice versa). Obviously, the value of explicit and implicit votes can also be statically defined a priori.

17. When OntoExtractor is used to generate assertions semiautomatically, it is reasonable to assign a higher trustworthiness than when assertions are generated without human intervention.

6.2 Aggregating Trust Values

The choice of the operator used to aggregate votes is, of course, a crucial and difficult one.[18] Our solution, based on a *compensatory* notion of aggregation, is an alternative to probabilistic approaches (e.g., Bayesian Systems or Belief Methods).[19] Several aggregation functions are available; for instance, plain *Weighted Mean* [63] aggregates votes from different sources, taking into account the reliability of each source. The *Ordered Weighted Averaging* (OWA) operator [50], [64], [65] weights votes in relation to their value, without taking into account which users have cast them. In nonanonymous environments, however, different categories of users can cast their votes on an assertion; therefore, it is important to weight each vote according to the reliability of the user who cast it (e.g., based on her role). Furthermore, votes can be aggregated depending on a number of other criteria (e.g., user location, connection medium, etc.). For this reason, we adopt a *Weighted OWA* (WOWA) [66] aggregation function combining the advantages of the OWA and WM operators. WOWA uses two sets of weights: $\mathbf{p}$ corresponding to the *relevance of the sources* and $\mathbf{w}$ to the *relevance of the values*.

Definition 1. *Let* $\mathbf{p}$ *and* $\mathbf{w}$ *be weight vectors of dimension* n *(*$\mathbf{p} = [p_1 p_2 \dots p_n]$, $\mathbf{w} = [w_1 w_2 \dots w_n]$*) such that 1)* $p_i \in [0,1]$ *and* $\sum_i p_i = 1$ *and 2)* $w_i \in [0,1]$ *and* $\sum_i w_i = 1$. *In this case a mapping* $f_{\mathrm{WOWA}} : \mathbb{R}^n \rightarrow \mathbb{R}$ *is a* Weighted Ordered Weighted Averaging *(WOWA) operator of dimension* n *if*

$$f_{\mathrm{WOWA}}(a_1, a_2, \dots, a_n) = \sum_i \omega_i a_{\sigma(i)},$$

where $\{\sigma(1), \sigma(2), \dots, \sigma(n)\}$ *is a permutation of* $\{1, 2, \dots, n\}$ *such that* $a_{\sigma(i-1)} \geq a_{\sigma(i)}$ *for all* $i = 2, \dots, n$. *With* $a_{\sigma(i)}$, *we indicate the* i*th largest element in the collection* $\{a_1, a_2, \dots, a_n\}$ *and the weight* ω_i *is defined as*

$$\omega_i = w^*\left(\sum_{j \leq i} p_{\sigma(j)}\right) - w^*\left(\sum_{j < i} p_{\sigma(j)}\right) \quad (5)$$

with w^* *a monotonic function (e.g., a polynomial) that interpolates the points* $(i/n, \sum_{j \leq i} w_j)$ *together with the point* $(0,0)$.[20] ω *is used to represent the set of weights* $\{\omega_i\}$: $\omega = \{\omega_1, \omega_2, \dots, \omega_n\}$.

Example 2. For the sake of simplicity, we initially assume the array $\mathbf{w}$ to be composed of values in the form $w_i = \frac{k}{n}$.[21] Intuition suggests that the impact of an explicit vote on an assertion must be much higher than the impact of an implicit vote. The difference between explicit and implicit votes was already taken into account when defining them, but the WOWA operator and a proper $\mathbf{w}$ vector can further increase it when needed. Much in the same way, weights in vector $\mathbf{p}$ can give higher importance to an explicit vote coming from a reputable source with respect to one whose source is unknown.

Let us suppose $\mathbf{a} = [\hat{p}_{M_k}\, \hat{p}_{M_u}\, \check{p}_{M_k}\, \check{p}_{M_u}] = [.9 .9 .5 .5]$, where k represents a known and reputable voter and u an unknown one. In order to privilege explicit votes with respect to implicit ones, the vector $\mathbf{w}$ is expressed as $\mathbf{w} = [.4 .4 .1 .1] = \left[\frac{2}{5}, \frac{2}{5}, \frac{1}{10}, \frac{1}{10}\right]$. The function w^* interpolating the points $\left(\left(\frac{1}{4}, \frac{2}{5}\right), \left(\frac{2}{4}, \frac{4}{5}\right), \left(\frac{3}{4}, \frac{9}{10}\right), (1,1)\right)$ with the point $(0,0)$ is $w^*(x) = \frac{32}{5}x^4 - \frac{64}{5}x^3 + \frac{34}{5}x^2 + \frac{3}{5}x$.

Now, in order to assign an higher impact to reputable voters, we assume $\mathbf{p} = [.4 .1 .4 .1] = \left[\frac{2}{5}, \frac{1}{10}, \frac{2}{5}, \frac{1}{10}\right]$, obtaining

$$\omega_1 = w^*\left(\frac{2}{5}\right) = \frac{2,102}{3,125},$$
$$\omega_2 = w^*\left(\frac{1}{2}\right) - w^*\left(\frac{2}{5}\right) = \frac{398}{3,125},$$
$$\omega_3 = w^*\left(\frac{9}{10}\right) - w^*\left(\frac{1}{2}\right) = \frac{362}{3,125},$$
$$\omega_4 = w^*(1) - w^*\left(\frac{9}{10}\right) = \frac{263}{3,125}.$$

The final trust value is $p_M = \mathcal{A}(.9, .9, .5, .5)$, where, in this case, $\mathcal{A} = f_{\mathrm{WOWA}}$. Therefore, we get:

$$p_M = f_{\mathrm{WOWA}}(.9, .9, .5, .5) = \sum_{i=1}^{4} \omega_i a_i = 0.82.$$

6.3 Trust-Based Evaluation: A Worked-Out Example

In this section, we show the result of a simulation executed with our Trust Layer tool, based on metadata produced by the OntoExtractor and described in Section 5.4. Our sample population is composed of three user categories, 1) *Senior Specialist*, 2) *Junior Specialist*, and 3) *Employee*, interacts with metadata produced semiautomatically and each group of users modifies over the time metadata visibility voting on them according to its *expertise* level. According to Example 2 in Section 6.2, here we assign a high expertise value to the first group of users and a low expertise level to the last one. Expertise level is encoded as the relevance of the sources vector $\mathbf{p}$, according to Definition 1.

The vector $\mathbf{a}$ containing users (explicit/implicit) votes is generated at every iteration, based on the users' *skills*, regarded as an estimator of the probability that they will vote correctly on an assertion. The values of the votes are linked to the initial trust value given to the metadata, according to the method described at the end of Section 6.1. In this example, we will take into consideration the initial trust values connected to metadata describing document D_2 showing how much (the encoding of) D_2 matches (the encodings of) classes c_2 and c_3, namely, 0.77 and 0.71, as shown in Table 2 in Section 5.4.

Finally, vector $\mathbf{w}$ (the relevance of the values vector) contains values generated directly from $\mathbf{a}$, depending on the attitude ("diffident" or "confident") adopted by the user aggregating the votes. We show the results of two different simulations, differing in the population composition only:

18. Experience has shown that, when votes are cast on data (see [61]) rather than on assertions, data semantics may not be sufficiently clear to suggest the correct aggregation operator to use.

19. Here, we describe our aggregation technique without evaluating it; in [35], we showed that the global convergence speed of our technique is faster than the EigenTrust algorithm described in [62].

20. It is adequate for any method that defines a bounded monotone function from monotone data and bounded in the unit interval.

21. In [67], a first case is illustrated, called *diffident approach*, where $k = 1, 2, \dots, n$, and a second one, the *confident approach*, where $k = n, n-1, \dots, 1$. Considering the fact that the WOWA operator is applied to a permutation of data values a_i (where $a_{\sigma(i-1)} \geq a_{\sigma(i)}, \forall i = 2, \dots, n$), it is easy to understand why the first approach is called "diffident:" It lessens the impact of high trust values. On the contrary, with the "confident" approach, we increase the impact of higher values.

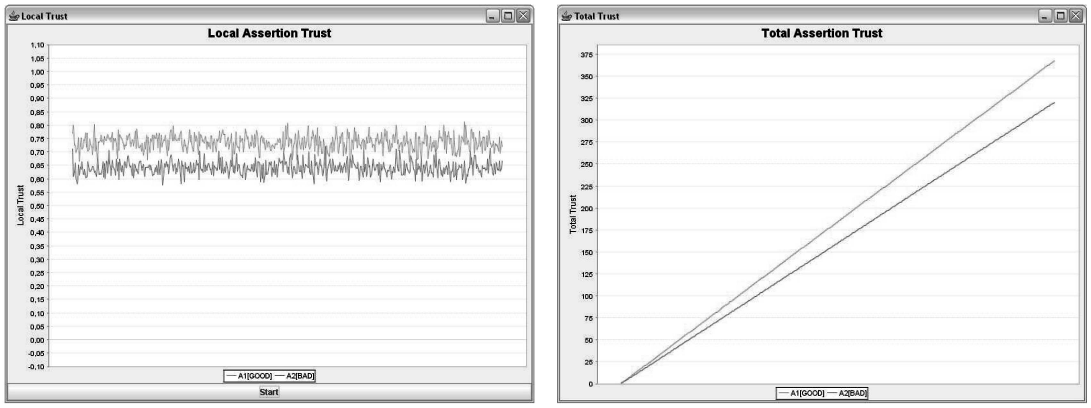


Fig. 12. The result of Simulation 1: The left graph shows the result of the WOWA aggregation for each iteration and the right one shows the global level of trust for each assertion.

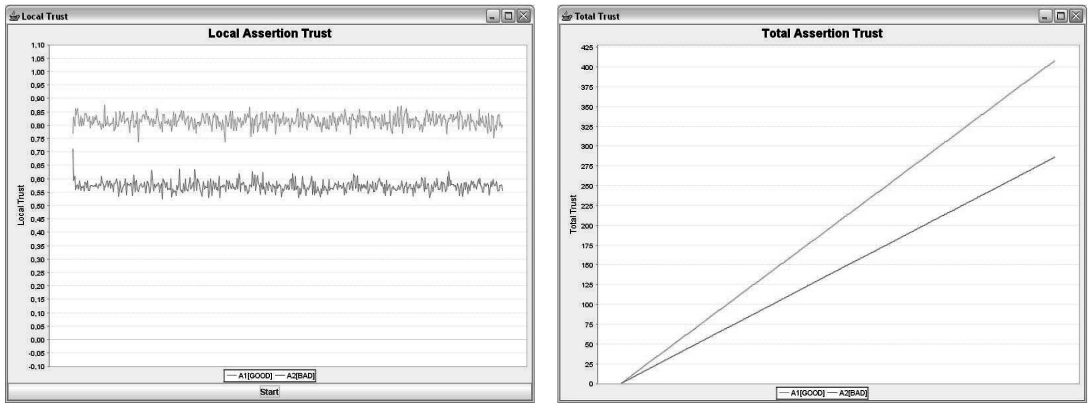


Fig. 13. Increasing the number of Senior Specialists rapidly increases the difference between the level of trust between assertions.

1) Senior $= 20\%$ Junior $= 30\%$ Employee $= 50\%$,
2) Senior $= 60\%$ Junior $= 20\%$ Employee $= 20\%$.

POPULATION: 100 users NUMBER OF ITERATIONS: 500

ASSERTIONS: $A1$—Trust Initial Value = 0.77
$A2$—Trust Initial Value = 0.71

p VECTOR: $p_i = 0.9$ for Senior
$p_i = 0.5$ for Junior
$p_i = 0.3$ for Employee

w VECTOR: $w_i = 0.8$ if $a_i > 0.5$
$w_i = 0.2$ if $a_i \leq 0.5$ $0.1 \leq a_i < 1$

a VECTOR$_{A1}$: $a_{i(\text{explicit})} = 0.77^{\delta}$
$a_{i(\text{implicit})} = 0.77^{\frac{1}{\delta}}$
$\delta = 0.4$

a VECTOR$_{\text{A2}}$: $a_{i(\text{explicit})} = 0.71^{\delta}$
$a_{i(\text{implicit})} = 0.71^{\frac{1}{\delta}}$
$\delta = 0.5$

SKILLS: 0.9 for Senior
0.5 for Junior
0.3 for Employee

This simple simulation shows the efficiency of our method in rapidly aggregating trust values connected to metadata and its sensitivity in parameter changing. Fig. 12 and Fig. 13 show, over 500 iterations, the result of the WOWA aggregation for each iteration (left-hand side) and the global level of trust for each assertion (right-hand side). Comparing Fig. 12 with Fig. 13, we see that the level of trustworthiness of the two different assertions visibly increases or decreases depending on the population composition. Note that, for the sake of conciseness, here we kept the composition of the voters at any given time consistent with the one of the community. This simplifying assumption allowed using a system *without memory*, i.e., taking into account only votes cast at the time of aggregation. It is possible to avoid this assumption by choosing, at every iteration, explicit and implicit votes depending on the last obtained local trust value for a resource, for example, as $a_{i(\text{explicit})}(t+1) = (\texttt{local\_trust}(t))^{\delta}$ and $a_{i(\text{implicit})}(t+1) = (\texttt{local\_trust}(t))^{\frac{1}{\delta}}$.

7 CONCLUSIONS

Although a huge literature on knowledge extraction is available, developing and maintaining ontology-based metadata is still more an art than a science. Rapid evolution of available information is difficult to control and often potentially useful, emerging concepts remain unnoticed. This paper addressed this problem by a bottom-up, semiautomatic fuzzy technique for generating ontologies from semistructured data flows. We consider trust-based validation of extracted metadata based on user community's feedback as the only viable alternative to both semiautomatic and fully automatic merging of ontology updates, the former being, in our opinion, still too cumbersome for practical applications, and the latter being too error prone for large-scale adoption. Of course, our community-based validation will require careful tuning and research on suitable voting algorithms [68]. Future developments of our work include a complete framework for community-based representation and update of trust-related assertions expressing a community's view over an automatically constructed ontology.

Acknowledgments

This work was partly funded by the Italian Ministry of Research Fund for Basic Research (FIRB) under project "Knowledge Management for the Web Infrastructure" (KIWI).

References

[1] G. Burnett, M.H. Dickey, M.M. Kazmer, and K.M. Chudoba, "Inscription and Interpretation of Text: A Cultural Hermeneutic Examination of Virtual Community," *Information Research*, vol. 9, no. 1, 2003.

[2] E. Lesser and K. Everest, "Using Communities of Practice to Manage Intellectual Capital," *Ivey Business J.*, vol. 65, no. 4, pp. 37-41, 2001.

[3] A. Farquhar, R. Fikes, and J. Rice, "The Ontolingua Server: A Tool for Collaborative Ontology Construction," *Int'l J. Human-Computer Studies*, vol. 46, no. 6, pp. 707-727, 1997.

[4] E. Damiani, M.G. Fugini, and C. Bellettini, "A Hierarchy-Aware Approach to Faceted Classification of Objected-Oriented Components," *ACM Trans. Software Eng. Methodologies*, vol. 8, no. 3, pp. 215-262, 1999.

[5] B. Omelayenko, "Learning of Ontologies for the Web: The Analysis of Existent Approaches," citeseer.ist.psu.edu/omelayenko01learning.html, 2001.

[6] M. Cristani and R. Cuel, "A Survey on Ontology Creation Methodologies," *Int'l J. Semantic Web and Information Systems*, vol. 1, no. 2, pp. 49-69, 2005.

[7] E. Morin, "Using Lexico-Syntactic Patterns to Extract Semantic Relations between Terms from Technical Corpus," *Proc. Fifth Int'l Congress on Terminology and Knowledge Eng. (TKE '99)*, pp. 268-278, 1999.

[8] P. Cimiano, A. Pivk, L. Schmidt-Thieme, and S. Staab, "Learning Taxonomic Relations from Heterogeneous Sources of Evidence," *Ontology Learning from Text: Methods, Applications, Evaluation*, 2005, http://www.uni-koblenz.de/~staab/Research/Publications/2005/OL-book-chapter-cimiano.pdf.

[9] A. Budanitsky, "Semantic Distance in Wordnet: An Experimental, Application-Oriented Evaluation of Five Measures," 2001, citeseer.ist.psu.edu/budanitsky01semantic.html.

[10] M.T. Pazienza, M. Pennacchiotti, and F.M. Zanzotto, "Identifying Relational Concept Lexicalisations by Using General Linguistic Knowledge." *Proc. 16th Eureopean Conf. Artificial Intelligence (ECAI '04)*, pp. 1071-1072, Aug. 2004.

[11] P. Adriaans and D. Zantinge, *Data Mining.* Addison-Wesley Longman, 1997.

[12] A. Maedche and S. Staab, "Ontology Learning for the Semantic Web," *IEEE Intelligent Systems*, vol. 16, no. 2, pp. 72-79, 2001.

[13] R. Volz, R. Studer, A. Maedche, and B. Lauser, "Pruning-Based Identification of Domain Ontologies," *J. Universal Computer Science*, vol. 9, no. 6, pp. 520-529, 2003.

[14] P. Mitra and G. Wiederhold, "An Algebra for Semantic Interoperability of Information Sources," *Proc. IEEE Int'l Conf. Bioinformatics and Biomedical Eng.*, pp. 174-182, 2001.

[15] I. Schmitt and G. Saake, "Merging Inheritance Hierarchies for Database Integration," *Proc. Third IFCIS Int'l Conf. Cooperative Information Systems*, pp. 322-331, 1998.

[16] R. Wille, "Restructuring Lattice Theory: An Approach Based on Hierarchies of Concepts," *Ordered Sets*, pp. 445-470, 1982.

[17] B. Stein and S. Meyerzu Eißen, "Document Categorization with MAJORCLUST," *Proc. 12th Workshop Information Technology and Systems (WITS '02)*, pp. 91-96, 2002.

[18] M. Steinbach, G. Karypis, and V. Kumar, "A Comparison of Document Clustering Techniques," 2000, citeseer.ist.psu.edu/steinbach00comparison.html.

[19] E.-H. Han, G. Karypis, and V. Kumar, "Text Categorization Using Weight Adjusted K-Nearest Neighbor Classification," *Proc. Fifth Pacific-Asia Conf. Knowledge Discovery and Data Mining*, pp. 53-65, 2001.

[20] A.K. Jain and R.C. Dubes, *Algorithms for Clustering Data.* Prentice-Hall, 1988.

[21] G. Hamerly and C. Elkan, "Alternatives to the K-Means Algorithm that Find Better Clusterings," *Proc. 11th Int'l Conf. Information and Knowledge Management*, pp. 600-607, 2002.

[22] K.M. Gupta, D.W. Aha, and P. Moore, "Learning Feature Taxonomies for Case Indexing Advances in Case-Based Reasoning," *Proc. Seventh European Conf.*, 2004.

[23] N. Fuhr and G. Weikum, "Classification and Intelligent Search on Information in XML," *IEEE Data Eng. Bull.*, vol. 25, no. 1, pp. 51-58, 2002.

[24] G. Bordogna and G. Pasi, "A User-Adaptive Indexing Model of Structured Documents," *Proc. 10th Int'l Conf. Fuzzy Systems*, pp. 984-989, 2001.

[25] E. Damiani, L. Tanca, and F. Arcelli Fontana, "Fuzzy XML Queries via Context-Based Choice of Aggregations," *Kybernetika*, vol. 36, no. 6, pp. 605-616, 2000.

[26] V. Carchiolo, A. Longheu, and M. Malgeri, "Hidden Schema Extraction in Web Documents," *Proc. Third Int'l Workshop Databases in Networked Information Systems (DNIS '03)*, pp. 42-52, 2003.

[27] P. Ceravolo and E. Damiani, "Fuzzy Mining Class Hierarchies from XML-Based Authentication Data," *Ontology Learning from Text: Methods, Applications, Evaluation*, Sept. 2004.

[28] P. Ceravolo, A. Corallo, E. Damiani, G. Elia, M. Viviani, and A. Zilli, "Bottom-Up Extraction and Maintenance of Ontology-Based Metadata," *Fuzzy Logic and the Semantic Web, Computer Intelligence*, Elsevier, to appear.

[29] T.R. Gruber, "Toward Principles for the Design of Ontologies Used for Knowledge Sharing," *Int'l J. Human-Computer Studies*, vol. 43, nos. 5-6, pp. 907-928, 1995.

[30] E. Damiani, R. Khosla, and W.I. Grosky, *Human Centered e-Business.* Kluwer Academic, 2003.

[31] N.F. Noy and M.A. Musen, "PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment," *Proc. 17th Nat'l Conf. Artificial Intelligence (AAAI/IAAI)*, pp. 450-455, 2000, citeseer.ist.psu.edu/noy00prompt.html.

[32] P. Ceravolo, E. Damiani, and M. Viviani, "Adding a Trust Layer to Semantic Web Metadata," *Soft Computing for Information Retrieval on the Web*, to appear.

[33] M. Blaze, J. Feigenbaum, and A.D. Keromytis, "The Role of Trust Management in Distributed Systems Security," *Secure Internet Programming*, pp. 185-210, 1999, citeseer.ist.psu.edu/23045.html.

[34] A. Jøsang, R. Ismail, and C. Boyd, "A Survey of Trust and Reputation Systems for Online Service Provision," *Decision Support Systems*, 2005.

[35] R. Aringhieri, E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, and P. Samarati, "Fuzzy Techniques for Trust and Reputation Management in Anonymous Peer-to-Peer Systems," *J. Am. Soc. Information Science and Technology (JASIST)*, to appear.

[36] Extensible Markup Language (XML), http://www.w3.org/XML/, 2005.

[37] Z. Cui, E. Damiani, M. Leida, and M. Viviani, "OntoExtractor: A Fuzzy-Based Approach in Clustering Semistructured Data Sources and Metadata Generation," *Proc. Ninth Int'l Conf. Knowledge-Based Intelligent Information and Eng. Systems (KES '05)*, Sept. 2005.

[38] E. Damiani and L. Tanca, "Blind Queries to XML Data," *Proc. 11th Int'l Conf. Database and Expert Systems Applications (DEXA '00)*, pp. 345-356, 2000.

[39] D.R. Cutting, D.R. Karger, J.O. Pedersen, and J.W. Tukey, "Scatter/Gather: A Cluster-Based Approach to Browsing Large Document Collections," *Proc. 15th Ann. Int'l ACM Conf. Research and Development in Information Retrieval (SIGIR '92)*, pp. 318-329, 1992.

[40] P. Kilpeläinen, "Tree Matching Problems with Applications to Structured Text Databases," PhD dissertation, Dept. of Computer Science, Univ. of Helsinki, 1992.
[41] B. Bouchon-Meunier, M. Rifqi, and S. Bothorel, "Towards General Measures of Comparison of Objects," *Fuzzy Sets Systems,* vol. 84, no. 2, pp. 143-153, 1996.
[42] P. Bosc, E. Damiani, and M.G. Fugini, "Fuzzy Service Selection in a Distributed Object-Oriented Environment," *IEEE Trans. Fuzzy Systems,* vol. 9, no. 5, Oct. 2001.
[43] E. Damiani, M.C. Nocerino, and M. Viviani, "Knowledge Extraction from an XML Data Flow: Building a Taxonomy Based on Clustering Technique," *Proc. EUROFUSE 2004: Eighth Meeting EURO Working Group on Fuzzy Sets,* pp. 133-142, 2004.
[44] P. Ceravolo, M.C. Nocerino, and M. Viviani, "Knowledge Extraction from Semistructured Data Based on Fuzzy Techniques," *Proc. Eighth Int'l Conf. Knowledge-Based Intelligent Information and Eng. Systems (KES '04),* pp. 328-334, 2004.
[45] D. Rocacher, "On Fuzzy Bags and Their Application to Flexible Querying," *Fuzzy Sets and Systems,* vol. 140, no. 1, pp. 93-110, 2003.
[46] L. Zadeh, "A Computational Approach to Fuzzy Quantifiers in Natural Languages," *Computing and Math. with Applications J.,* no. 9, pp. 149-184, 1983.
[47] G. Cormode and S. Muthukrishnan, "The String Edit Distance Matching Problem with Moves," *Proc. 13th Ann. ACM-SIAM Symp. Discrete Algorithms (SODA '02),* pp. 667-676, 2002.
[48] E. Damiani, N. Lavarini, B. Oliboni, and L. Tanca, "An Approximate Querying Environment for XML Data," *Fuzzy Logic and the Internet,* vol. 137, Jan. 2004.
[49] D. Dubois, "Triangular Norms for Fuzzy Sets," *Proc. Second Int'l Seminar Fuzzy Set Theory,* pp. 39-68, 1980.
[50] R.R. Yager, "On Ordered Weighted Averaging Aggregation Operators in Multicriteria Decisionmaking," *IEEE Trans. Systems, Man, and Cybernetics,* vol. 18, no. 1, pp. 183-190, 1988.
[51] J.A. Hylton, "Identifying and Merging Related Bibliographic Records," Technical Report MIT/LCS/TR-678, MIT Lab for Computer Science, Cambridge, Mass., http://ltt-www.lcs.mit.edu/ltt-www/People/jeremy/thesis/, 1996.
[52] Zip Distance Calculator Web Service, http://www.webservicex.net/uszip.asmx, Sept. 2005.
[53] P. Bosc and D. Rocacher, "About $\mathbb{Z}_f$, the Set of Fuzzy Relative Integers, and the Definition of Fuzzy Bags on $\mathbb{Z}_f$," *Proc. 10th Int'l Fuzzy Systems Assoc. World Congress on Fuzzy Sets and Systems (IFSA '03),* T. Bilgiç, B.D. Baets, and O. Kaynak, eds., pp. 95-102, June-July 2003.
[54] Restaurant Ontology, Imperial College of Science, Technology and Medicine, Jan. 2006. http://www-agentcities.doc.ic.ac.uk/ontology/restaurant.
[55] R. Richardson, A.F. Smeaton, and J. Murphy, "Using WordNet as a Knowledge Base for Measuring Semantic Similarity Between Words," Technical Report CA-1294, Dublin, Ireland, citeseer.ist.psu.edu/richardson94using.html, 1994.
[56] Iso 3166 Maintenance Agency, http://www.iso.org/iso/en/prods\-services/iso3166ma/index.html, Sept. 2005.
[57] D. Rocacher and P. Bosc, "The Set of Fuzzy Rational Numbers and Flexible Querying," *Fuzzy Sets and Systems,* vol. 155, pp. 317-339, 2005.
[58] M. Wygralak, "An Axiomatic Approach to Scalar Cardinalities of Fuzzy Sets," *Fuzzy Sets Systems,* vol. 110, no. 2, pp. 175-179, 2000.
[59] G. Adomavicius and A. Tuzhilin, "Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions," *IEEE Trans. Knowledge and Data Eng.,* vol. 17, no. 6, pp. 734-749, June 2005.
[60] R. Mesiar and M. Komorníková, "Aggregation Operators," *Proc. XI Conf. Applied Math. (PRIM '96),* pp. 193-211, 1996.
[61] A. Ceselli, E. Damiani, S. DeCapitanidiVimercati, S. Jajodia, S. Paraboschi, and P. Samarati, "Modeling and Assessing Inference Exposure in Encrypted Databases," *ACM Trans. Information Systems Security,* vol. 8, no. 1, pp. 119-152, 2005.
[62] S.D. Kamvar, M.T. Schlosser, and H. Garcia-Molina, "The Eigentrust Algorithm for Reputation Management In P2P Networks," *Proc. 12th Int'l Conf. World Wide Web (WWW '03),* pp. 640-651, 2003.
[63] J. Aczél, "On Weighted Synthesis of Judgements," *Aequationes Math.,* vol. 27, pp. 288-307, 1984.
[64] J. Fodor, J.L. Marichal, and M. Roubens, "Characterization of the Ordered Weighted Averaging Operators," *IEEE Trans. Fuzzy Systems,* vol. 3, no. 2, pp. 236-240, 1995.
[65] M. Grabisch, "Fuzzy Integral in Multicriteria Decision Making," *Fuzzy Sets Systems,* vol. 69, no. 3, pp. 279-298, 1995.
[66] V. Torra, "The Weighted Owa Operator," *Int'l J. Intelligent Systems,* vol. 12, no. 2, pp. 153-166, 1997.
[67] E. Damiani, S. De Capitani di Vimercati, P. Samarati, and M. Viviani, "A Wowa-Based Aggregation Technique on Trust Values Connected to Metadata," *Proc. First Int'l Workshop Security and Trust Management (STM '05),* Sept. 2005.
[68] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, and P. Samarati, "Managing and Sharing Servents' Reputations in P2P Systems," *IEEE Trans. Knowledge and Data Eng.,* vol. 15, no. 4, pp. 840-854, July/Aug. 2003.
[69] M.J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li, "New Algorithms for Fast Discovery of Association Rules," *Proc. Third Int'l Conf. Knowledge Discovery and Data Mining (KDD '97),* 1997.
[70] P. Ceravolo, "Extracting Role Hierarchies from Authentication Data Flows," *Int'l J. Computer Systems Science and Eng.,* vol. 4, no. 6, 2004.
[71] I. Horrocks, U. Sattler, and S. Tobies, "Practical Reasoning for Expressive Description Logics," *Proc. Sixth Int'l Conf. Logic Programming and Automated Reasoning (LPAR '99),* pp. 161-180, 1999.
[72] OWL Web Ontology Language—Overview, World Wide Web Consourtium, http://www.w3.org/TR/owl-features/, Dec. 2003.

Paolo Ceravolo received the Laurea degree in philosophy from the Philosophy Department at the Catholic University of Milan, Italy, and the PhD degree in computer science from the Department of Information Technologies at the University of Milan, Italy. Currently, he is an assistant professor in the same department. His research interests include ontology-based knowledge extraction and management, semantic Web technologies, trust/reputation, and soft computing. On these topics, he has published several scientific papers and book chapters.

Ernesto Damiani is a full professor in the Department of Information Technology at the University of Milan, Italy. He has held visiting positions at several international institutions, including George Mason University, Virginia, and is an adjunct professor at the University of Technology, Sydney, Australia. He coordinates several research projects funded by the Italian Ministry of Research, the European Commission, and by a number of private companies, including Cisco, ST Microelectronics, Siemens Mobile, and BT Exact. His research interests include knowledge extraction and metadata design, advanced software architectures, and soft computing. On these topics, he has filed international patents and published more than 200 refereed papers in international journals and conferences. He is the vice chair of the IFIP WG on Web Semantics (WG 2.12) and the Secretary of the IFIP WG on Open Source Systems (WG 2.13). He is the author, together with W. Grosky and R. Khosla, of the book *Human-Centered e-Business* (Kluwer, 2003). In 2000, he was the recipient of ACM SIGAPP Outstanding Service Award. He is a member of the IEEE.

Marco Viviani received the Laurea degree in computer science from the Department of Information Technologies at the University of Milan, Italy. Currently, he is a research collaborator and a PhD student in the same department. His research interests include knowledge extraction and management, semantic Web technologies, data mining techniques, pattern recognition and clustering, trust/reputation, and fuzzy logic. On these topics, he has published several scientific papers and book chapters.