An Ontology-Driven Annotation of Data Tables

Gaëlle Hignette[1], Patrice Buche[1], Juliette Dibie-Barthélemy[1], and Ollivier Haemmerlé[2]

[1] UMR AgroParisTech/INRA MIA - INRA Unité Mét@risk
AgroParisTech, 16 rue Claude Bernard, F-75231 Paris Cedex 5, France
{hignette,buche,dibie}@agroparistech.fr

[2] IRIT - Université Toulouse le Mirail, Dpt. Mathématiques-Informatique
5 allées Antonio Machado, F-31058 Toulouse Cedex
ollivier.haemmerle@univ-tlse2.fr

Abstract. This paper deals with the integration of data extracted from the web into an existing data warehouse indexed by a domain ontology. We are specially interested in data tables extracted from scientific publications found on the web. We propose a way to annotate data tables from the web according to a given domain ontology. In this paper we present the different steps of our annotation process. The columns of a web data table are first segregated according to whether they represent numeric or symbolic data. Then, we annotate the numeric (resp.symbolic) columns with their corresponding numeric (resp. symbolic) type found in the ontology. Our approach combines different evidences from the column contents and from the column title to find the best corresponding type in the ontology. The relations represented by the web data table are recognized using both the table title and the types of the columns that were previously annotated. We give experimental results of our annotation process, our application domain being food microbiology.

Keywords: ontology-driven data integration, semantic annotation.

1 Introduction

In the scientific world, many experimental data are produced and continually published on the web. It is often hard to keep track of all the experiments that are conducted in a specific domain, and even harder to compile all the results from different experiments at the time when one needs them. Our work is applied to the domain of food microbiology. In a first system called MIEL [1], data on food microbiology coming from scientific publications or industrial sources is manually entered into a relational database: this database is indexed with a domain ontology, so that data coming from different sources is represented with a certain uniformity. Users can query the database using an interface that allows them to select, within the domain ontology, the food product, microorganism and relations they are interested in. The database is then queried to find data corresponding to or close to the users selection criteria, and the answers are ordered according to how close they are to the users criteria. However, manually feeding this database is very time-consuming, resulting in a largely incomplete database. Our work aims at building an XML data warehouse that completes the database with data gathered on the web that is automatically annotated with the same ontology, so as to provide a uniform way of querying. We focus on data that are presented in tables, since it is a usual way of presenting synthetic information in many scientific or economic domains, and in particular in food microbiology. Our goal is to annotate the table contents and recognize the relations represented in the tables. The annotated data tables can then be queried using the ontology. Our whole system relies on the domain ontology, so that changing the ontology is enough to change the application domain, provided that we are looking for data presented in tables.

There has been a lot of research work on table recognition, both for the recognition of tables within text and for the detection of the table orientation [2]. Our work is not focused on table recognition or orientation, but on annotating a table that has already been recognized and put in a standard orientation where relations are represented in lines, the first line of the table being the column titles and the next lines being the actual data we want to extract. In [3], frames are constructed from tables using semantic tools such as the WordNet ontology or GoogleSets. However, in their approach they create new relations with names according to the relation signature and a generic ontology, whereas we want to recognize predefined relations in an ontology specific to the targetted domain. In a more recent work [4], relations from an ontology are instanciated using various HTML structures including tables. However, they only identify binary concept-role relations between instances that are assumed to be already annotated (manually or using another information extraction system). Our work differs as we focus on the recognition of n-ary relations and we propose a step-by-step algorithm including the recognition of element types. From this point of view, the work presented in [5] is nearer to ours, as they transform tables of different structures into a common relational database schema with n-ary relations. However, their approach is dependent on the manual definition of an "extraction ontology" that defines extraction rules for each set of objects, giving all synonyms for an attribute name or defining the context of apparition of a string to extract. Our work differs as the extraction rules we build are independant from the ontology: the domain experts who build the ontology only define which data are of interest for their purpose, they don't have to concentrate on the way to find them. Other annotation techniques applicable to tables, based on learning wrappers using textual context and / or structure, such as BWI [6] or Lixto [7], are not interesting in our application for several reasons: there is no textual context for the apparition of terms in the tables, as we consider the entire content of a table cell for annotation; the table structures are not homogeneous; data is rare in our domain, so it is very difficult for us to rely on learning techniques, as obtaining data to train learning algorithms is a real challenge.

We first present the structure of the ontology that we use during our annotation process (Sect. 2). In the tables that we consider for annotation, semantic relations are represented as rows. Our aim is to recognize the signature of a known relation by looking at the different column types of the table. Our way of finding the type of a column differs depending on whether the column is numeric or symbolic. Thus, the first step of our annotation process consists in classifying columns as numeric or symbolic (Sect. 3). For the annotation of both numeric and symbolic columns, we use a similarity measure between the terms from the web and the terms from the ontology (Sect. 4). Then we present the way of finding the type of a numeric column (Sect. 5) and the type of a symbolic column (Sect. 6). We then show how the knowledge of the different column types is used to find the relations represented in the table (Sect. 7).

2 Structure of the Ontology

Figure 1 shows the structure of a simplified version of our domain ontology. The ontology is built in collaboration with domain experts, who ideally are the future users of the querying system that exploits the annotated tables. They define the numeric types, symbolic types and relations that are interesting for their domain.

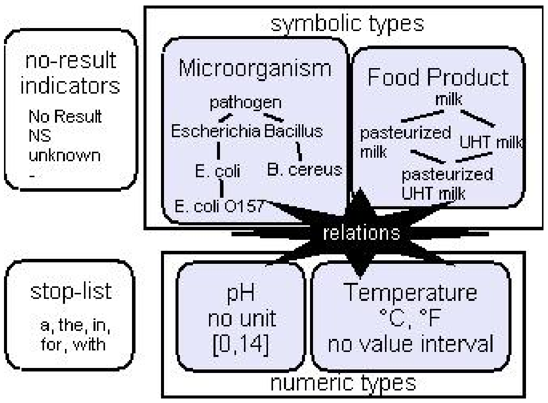

Fig. 1. Structure of the ontology used for our annotation process

Numeric types are used to define the numeric data that we want to extract from the tables. A numeric type in the ontology is described by the name of the type, the units in which data of this type is usually expressed, and the interval of possible values for this type. For example, the type named "Temperature" can be expressed in the units {°C, °F} and has a range of $]-\infty, +\infty[$ while the type "pH" has no unit and has a range of $[0, 14]$.

Symbolic types are used when the data of interest is represented as a string. A symbolic type in the ontology is described by the name of the type and the type hierarchy (which is the set of possible values for the type, partially ordered by the subsumption relation). Such a hierarchy can be either a tree or a lattice, depending on whether it has sense for the domain that a concept has several parents: for example the type hierarchy attached to “Microorganism” is a biological taxonomy of selected microorganisms, additionally divided between the categories “pathogen”, “spoiler” and “useful for food process”.

Relations are used to represent semantic links between different types. A relation in the ontology is described by the name of the relation and its signature. Relations originate from a relational database schema: they were n-ary with a similar role for all types in the signature. However, we were able to redesign the relation definition to correspond to an application having for domain the set of controlled factors (numeric or symbolic types) in the experimental plan and for range the resulting experimental measure (always a numeric type). For example, the relation “Growth kinetics” of our ontology, which measures the growth of a microorganism over time, has for domain the set of types “Microorganism”, “Food product”, “Temperature”, “Time”, and for range the type “Colony count” (the microorganism concentration). As several measures of non-controlled factors can be taken in one experiment, tables often represent several relations. Note that the relations are not functions: due to biological variations and to other factors not taken into account in the model of a relation, the same set of values in the domain types can lead to distinct values in the range type.

In addition to the numeric or symbolic types and the relations defined in collaboration with domain experts, the ontology also presents two lists that are not really specific to the domain, but that present lexical knowledge useful for our annotation process. The stopword list contains a list of words that have more a grammatical role than a semantic one, such as articles and conjunctions. The “no-result indicator” list is a list of terms that are used to represent the absence of data, for example “No Result” or “NS” (for Not Specified).

3 Distinction Between Numeric and Symbolic Columns

The first step of our annotation process is to distinguish between numeric and symbolic columns. For that purpose, we have built a set of rules, using the ontology. Let *col* be a column of the table we want to annotate. We search *col* for all occurrences of numbers (in decimal or scientific format) and of units of numeric types described in the ontology. We also search *col* for all words, which are defined as alphabetic character sequences that are neither units nor “no result indicators”.

Let c be a cell of the column *col*. We apply the following classification rules:

– if c contains a number immediately followed by a unit, or a number in scientific format, then c is numeric;
– else, if c contains more numbers and units than words, then c is numeric;
– else, if c contains more words than numbers and units, then c is symbolic;
– else (equal amount of words and numbers+units) the status of c is considered as unknown.

Once all cells of the column *col* have been classified using the above rules, *col* is classified as symbolic if there are more cells classified as symbolic than numeric.

Table 1. Classification results for numeric and symbolic column detection over 349 columns (as there is no unclassified column, precision is equal to recall)

manual \ computed	classification using ontology		naive classification	
	numeric	symbolic	numeric	symbolic
numeric	261	2	229	34
symbolic	5	81	13	73
precision/recall	98%		87%	

Else, the column is classified as numeric (we have experimentally shown that when numbers of symbolic and numeric cells are equal, it usually corresponds to a high rate of absent data, which is more frequent in numeric columns).

We have experimented our method on 60 tables taken from publications in food microbiology. We have compared our classifier with a naive classifier, for which a cell is numeric if and only if it contains a number. Results are shown in Tab. 1. Our method gives much better results than the naive classifier because it is able to consider as non-numeric a cell that contains numbers (for example a microorganism with a strain number) as well as it is able to deal with unknown data: the "no result indicators" are not considered as words, thus leading to an unrecognized cell, whereas the naive method considers them as symbolic.

4 Similarity Measure Between a Term from the Web and a Term from the Ontology

Throughout our whole annotation process, we will be using a similarity measure that allows to compare a term from the web with a term from the ontology, a term being a set of consecutive words. Several similarity measures to compare two terms are presented in [9]. Semantic similarity measures imply the use of an ontology that contains both terms to compare. In our case, there is not such an ontology: we have tested both Wordnet[1] and AgroVoc[2], but the terms used on the web to represent food products are too specific and do not appear in those thesauri. Instead of using a semantic similarity, we thus use a word-by-word similarity measure.

Words in terms from the ontology are manually weighted, according to their importance in the meaning of the term. The non-informative words listed in the stopword list are given a weight of 0, and other words are weighted with only two possible weights: 1 for the most informative word(s) of the term, 0.2 for secondary words (the weight of 0.2 has been chosen after preliminary experiments that showed the results were better than when using 0.5). Determining the importance of a word in a term meaning is done by a domain expert (for example, "cooked"

[1] http://wordnet.princeton.edu/

[2] Thesaurus for agriculture and food industry used by the FAO, http://www.fao.org/aims/ag_intro.htm

Table 2. Vector representation of the term from the web "ground meat" and two terms from the ontology

terms \ coordinates	ground	meat	fresh	beef	pork
ground meat	1	1	0	0	0
fresh meat	0	1	0.2	0	0
ground beef	0.2	0	0	1	0

in "cooked meat" is important if we are interested in microorganisms, while not if we are looking at chemical contaminants). We have shown in [10] that manual weight definition an improvement in annotation quality compared to giving a weight of 1 to each word, even if this improvement is quite small. Weighted terms are represented as vectors, in which the coordinates represent the different possible lemmatised words and their weight in the term (0 if the word does not belong to the term). Table 2 shows the vector representation of a term from the web and two terms from the ontology. The similarity between a term from the web and a term from the ontology is then a similarity between two weighted vectors: we have tested several similarity measures, such as the Dice coefficient [9] or the cosine similarity measure [11]. The choice of the similarity measure had no big impact on our results in the following steps of the annotation. We decided to keep the cosine similarity measure as it is the most popular one.

Definition 1. *Let w be a term from the web, represented as the weighted vector $\boldsymbol{w} = (w_1, \ldots, w_n)$ and o a term from the ontology, represented as the weighted vector $\boldsymbol{o} = (o_1, \ldots, o_n)$. The similarity between w and o is computed as:*

$$sim(w, o) = \frac{\sum_{k=1}^{n} w_k \times o_k}{\sqrt{\sum_{k=1}^{n} w_k^2 \times \sum_{k=1}^{n} o_k^2}} \tag{1}$$

Example 1. The similarity measures between the term "ground meat" from the web and the two terms from the ontology presented in table 2 are respectively:

- $sim(ground\ meat, fresh\ meat) = \frac{1\times0+1\times1+0\times0.2+0\times0+0\times0}{\sqrt{(1^2+1^2)+(1^2+0.2^2)}} \approx 0.57$
- $sim(ground\ meat, ground\ beef) = \frac{1\times0.2+1\times0+0\times0+0\times1+0\times0}{\sqrt{(1^2+1^2)+(0.2^2+1^2)}} \approx 0.11$

5 Numeric Column Annotation

When a column has been recognised as numeric (Sect. 3), we look for the numeric type of the ontology that corresponds to the column. For that purpose, we compute the score of each numeric type for the column. The score of a numeric type for the column is a combination of:

- the score of the numeric type for the column according to the column title. This score is the similarity measure (def. 1) between the column title and the numeric type name.

– the score of the numeric type for the column according to the units present in the column.

To compute the numeric type for the column according to the units in the column, we first compute the score of the numeric type for each unit that is present in the column: a numeric type has a score for each unit, depending on the number of numeric types that can be expressed in this unit.

Definition 2. *Let u be a unit and T_u the set of numeric types that can be expressed in this unit, the score of a type t for the unit u is $score(t, u) = 0$ if $t \notin T_u$, and $score(t, u) = \frac{1}{|T_u|}$ if $t \in T_u$.*

The score of a numeric type for the column is computed as the maximum of the scores of the type for each unit present in the column. If no unit from the ontology was identified in the column, then the column is considered as presenting the unit "no unit", which is treated as a normal unit in the ontology.

The final score of a numeric type t for the column col is a combination of the score of t for col according to the title of the column ($score_{title}(t, col)$), and the score of t for col according to the units in the column ($score_{unit}(t, col)$). However, types are filtered according to the numeric values presented in the column:

– if the column contains a numeric value outside the range of values for the type t, then $score_{final}(t, col) = 0$.
– if all values in the column are compatible with the range of type t, then $score_{final}(t, col) = 1 - (1 - score_{title}(t, col))(1 - score_{unit}(t, col))$. This method of combination of several scores is inspired by [12]: the score of the type t for the column col is much greater if there are several evidences (title of the column **and** units) that the type corresponds to the column.

Once the final score of each numeric type has been computed for the column, we choose the correct type for the column. Numeric types are ordered according to their final score for the column: let *best* be the type with the best score and *secondBest* be the type with the second best score. We compute the proportional advantage of *best* over *secondBest* on the column *col*:

$$advantage(best, col) = \frac{score_{final}(best, col) - score_{final}(secondBest, col)}{score_{final}(best, col)} \quad (2)$$

If $advantage(best, col)$ is greater than a threshold θ_{num} fixed by the user, then the column *col* is annotated with the type *best*. Else the type of *col* is considered as unknown.

We have experimented our method of numeric column annotation on the 261 columns that were correctly recognized as numeric (Sect. 3). The columns were manually annotated with the 18 numeric types of our domain ontology, and we compared the manual annotation with the types computed by our method, using the threshold $\theta_{num} = 0,1$. On the 261 columns, 243 were correctly annotated, 9 were considered as unknown and 9 were annotated with a wrong type (i.e. 96% precision, 93% recall). By comparison, when considering the score from title as the final score with no use of the units defined in the ontology (as was done in [8]), precision was 96% but recall was only 83%. We consider the results of our annotation of numeric columns as good enough, the use of the units defined in the ontology allowing better recall with no more errors.

6 Symbolic Column Annotation

As it has just been presented for numeric columns, we have to choose the correct symbolic type for a column that has been recognised as symbolic during the first step of our annotation process (Sect. 3). We first compute the score of each symbolic type for the column according to the title of the column, as the similarity measure (def. 1) between the column name and the names of the symbolic types. Then we compute the score of each symbolic type for the column according to the contents of the column. For that purpose, we compute the similarity measure (def. 1) between each term in the cells of the column and each term in the hierarchies of the symbolic types. Let c be the term in a cell of a symbolic column col, let t be a symbolic type and $hier(t)$ the set of all terms in the hierarchy of the type t. The score of the type t for the cell c is:

$$score(t, c) = \sum_{x \in hier(t)} sim(c, x) \tag{3}$$

We choose the type for the cell using the proportional advantage (Sect. 5). If the proportional advantage of the type having the best score for the cell is higher than a threshold $\theta_{symbCell}$, then the cell is considered as having this type, else the cell is considered as having an unknown type. When a type has been chosen for every cell in the column, the score of a type t for the column is the proportion of its cells that have been assigned the type t: let n be the number of cells in the column col and n_t the number of cells having the type t, then the score of t for the column col according to the column contents is:

$$score_{contents}(t, col) = \frac{n_t}{n} \tag{4}$$

As for the numeric columns, the final score of a symbolic type t for the symbolic column col is a combination of the score according to the column title and the score according to the column contents:

$$score_{final}(t, col) = 1 - (1 - score_{title}(t, col))(1 - score_{contents}(t, col)) \tag{5}$$

Once the final score of each symbolic type has been computed for the column, we choose the correct type for the column using the proportional advantage method (Sect. 5): if the proportional advantage of the best type for the column is greater than a threshold $\theta_{symbCol}$, then the column is annotated with this best type, else the column is considered as of unknown type.

We have experimented our symbolic column annotation using the 81 columns that were correctly recognized as symbolic (Sect. 3). In order to assess the quality of our results, we wanted to compare our method with a “standard” classifier. To our knowledge, there is no classifier that is dedicated to the classification of symbolic data using a domain ontology. We have thus decided to use the SMO classifier [13] implemented in Weka[3]: as it is an optimised version of the well-known SVM, it allows comparing our results with a “standard” method. It is thus a comparison between two alternatives: SMO uses no domain knowledge *but* uses learning, while our method is based on domain knowledge *but* has no learning phase. The 81 columns were manually annotated with the three symbolic types of the ontology (“Microorganism”, “Food product”, “Response”) and a type “Other” for additional information too specific to be entered in the ontology. Our annotation method was run using the following thresholds: $\theta_{symbCell} = \theta_{symbCol} = 0.1$, columns of unknown type being classified under the type “Other”. For the SMO classifier, we used the following pre-treatment: each distinct lemmatised word present in a column results in an attribute; the value of this attribute for a given column is the frequency of the word in the column. The SMO classifier was evaluated using a leave-one-out cross-validation, with default parameters of the Weka implementation. Results of this experiment are given in Tab. 3.

Table 3. Classification results on 81 symbolic columns

	our method using the ontology				SMO			
manual \ computed	Food	Micro.	Resp.	Other	Food	Micro.	Resp.	Other
Food	19	0	0	27	45	0	0	1
Micro.	0	6	0	10	4	12	0	0
Response	0	0	0	1	0	0	0	1
Other	2	0	0	16	7	0	0	11

Over the three symbolic types of the ontology (we are not interested in the “Other” column type), we obtain a precision of 93% and a recall of 66% with our annotation method, while the SMO classifier gives a 84% precision and a 90% recall. Our method, which uses domain knowledge but no learning phase, gives a better precision but a lower recall than the learning classifier. This is mainly because we have biased our annotation technique towards precision: whenever we do not know for sure the type of a column, it is considered as unknown.

7 Finding the Semantic Relations Represented by the Table

Once the types of all columns of a table have been recognized, we look for the relation(s) of the ontology that are represented in the table. As for the column types recognition, the final score of a relation for the table is the combination of two scores: the score of the relation for the table according to the table title, and the score of the relation for the table according to the table signature (the set of its recognized columns).

[3] http://www.cs.waikato.ac.nz/ml/weka

The score of a relation for the table according to the table title is computed as the similarity measure (def. 1) between the table title and the relation name.

The score of a relation *rel* for the table *tab* according to the table signature is computed as follows:

- if the numeric type that constitutes the range of the relation *rel* was **not** recognized as a type of a column of the table, then $score_{signature}(rel, tab) = 0$
- else, the score of the relation for the table is the proportion of types in its signature that were recognized in the table columns. Let $Sign_{rel}$ be the set of types in the signature of relation *rel* (i.e. the types that constitute the domain and the type that cpnstitutes the range), and $Sign_{tab}$ the set of types that were recognized for the table columns, then $score_{signature}(rel, tab) = \frac{|Sign_{rel} \cap Sign_{tab}|}{|Sign_{rel}|}$

Then the final score of a relation *rel* for the table *tab* is computed as:

$$score_{final}(rel, tab) = 1-(1-score_{title}(rel, tab))(1-score_{signature}(rel, tab)) \quad (6)$$

When the scores of all relations of the ontology have been computed for the table, we choose the relation(s) with which to annotate the table. A table can represent several relations at a time: for example, if a table gives the pH and the water activity of a food product, we will consider it as two separate relations: food pH and food water activity. If a relation has a non-zero score for the table and has no *concurrent relation*, this relation is used to annotate the table. Two relations are called *concurrent* if they have the same range. If there are several *concurrent relations* with non-zero scores for the table, then we only keep the one with the highest score for the annotation of the table (if several concurrent relations have the same highest score, we keep them all for the table annotation).

We have experimented this annotation method on the 60 tables that were used for all preceding experiments. Those tables were manually annotated with the 16 relations of the ontology: one table was typically annotated with 1 to 5 relations, which gives a total of 123 relations. We ran the different steps of our annotation system without validating the intermediate steps, i.e. even columns that were wrongly recognized in the symbolic versus numeric classification were further annotated and used for the relation annotation. Over the 123 relations in the manual annotation, 117 were correctly annotated with our annotation system, 6 were not recognized and there were 52 relations in the annotations that should not have been recognized. This gives a precision of 69% for a recall of 95%. Unfortunately, we cannot build a comparison of this method with another method to annotate tables, as we did not find other works on annotation of n-ary relations in tables using a domain ontology: the nearest work to ours is the one of [3] and it uses WordNet which is too general for our purpose.

In order to improve precision, we have tried to apply a score threshold on the relations: only relations with score greater than a threshold θ_{rel} are used for the annotation. The variation of precision and recall with the value of θ_{rel}, and the according F1-value is shown in Fig. 2. For $\theta_{rel} = 0.5$, there is a jump in precision to reach 95%, but also a fall in recall to 34%: increasing precision for our method means a too important drop in recall.

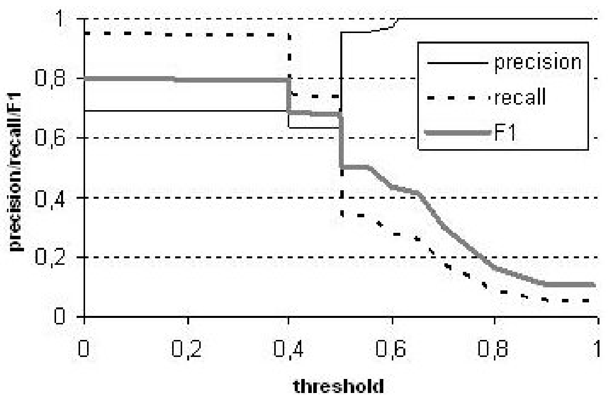

Fig. 2. Precision, recall and F1-value of relation annotation given the value of the threshold θ_{rel}

8 Conclusion and Perspectives

In this paper, we have shown the different steps of an annotation process that allows one to annotate data tables with the relations of a domain ontology. This annotation is entirely based on the domain knowledge given in the ontology, with no learning phase. The columns of a table are first segregated according to whether they represent numeric or symbolic data. Then, on top of the column titles, we use the units and interval of possible values defined in the ontology for numeric types to recognize the type of numeric columns, and we use the terms from the taxonomies of the symbolic types to recognize the type of symbolic columns. The relations represented by a table are recognized using both the table title and the types of the columns. When all steps are run one after the other, we obtain a high recall on relation recognition, with an acceptable precision level.

Our future works will aim at instanciating the relations according to the content of the cells in the table, not only annotating the symbolic contents with terms from the hierarchies of the corresponding types, but also analysing numeric values, dealing with intervals, standard deviations etc. Missing types that are in the relation signature but not in the table signature will be searched for: first we will try to identify them in the columns that have been annotated of unknown type, then we will also try and find if they are expressed as constants in the table title or in the sentences which reference the table. For example, a table named “Growth parameters for E. coli” is not likely to present a column of type “Microorganism” because the studied microorganism is a constant already given in the table title.

Fuzzy sets [14] will be used to represent similarities between symbolic cells and terms from the ontology, and to represent imprecise data for numeric cells. Then we will focus on the querying of the annotated data tables: the querying system must be integrated to the one already used in the MIEL system, which means that we will have to deal with user preferences, also expressed as fuzzy sets. Our querying system will have to take into account the different scores that we have computed during the annotation, that give hints about how sure we are of these annotations.

Acknowledgements

Financial support from the French National Research Agency (ANR) for the project WebContent in the framework of the National Network for Software Technology (RNTL) is gratefully acknowledged.

References

1. Buche, P., Dervin, C., Haemmerlé, O., Thomopoulos, R.: Fuzzy querying of incomplete, imprecise, and heterogeneously structured data in the relational model using ontologies and rules. IEEE T. Fuzzy Systems 13(3), 373–383 (2005)
2. Zanibbi, R., Blostein, D., Cordy, J.R.: A survey of table recognition: Models, observations, transformations, and inferences. International Journal on Document Analysis and Recognition 7, 1–16 (2004)
3. Pivk, A., Cimiano, P., Sure, Y.: From tables to frames. In: Third International Semantic Web Conference, pp. 116–181 (2004)
4. Tenier, S., Toussaint, Y., Napoli, A., Polanco, X.: Instantiation of relations for semantic annotation. In: International Conference on Web Intelligence, pp. 463–472 (2006)
5. Embley, D.W., Tao, C., Liddle, S.W.: Automatically extracting ontologically specified data from html tables of unknown structure. In: Spaccapietra, S., March, S.T., Kambayashi, Y. (eds.) ER 2002. LNCS, vol. 2503, pp. 322–337. Springer, Heidelberg (2002)
6. Freitag, D., Kushmerick, N.: Boosted wrapper induction. In: 17th National Conference on Artificial Intelligence, pp. 577–583 (2000)
7. Baumgartner, R., Flesca, S., Gottlob, G.: Visual web information extraction with Lixto. In: International Conference on Very Large Data Bases, pp. 119–128 (2001)
8. Gagliardi, H., Haemmerlé, O., Pernelle, N., Saïs, F.: An automatic ontology-based approach to enrich tables semantically. In: AAAI Context and Ontologies Workshop (2005)
9. Lin, D.: An information-theoretic definition of similarity. In: International Conference on Machine Learning, pp. 296–304 (1998)
10. Hignette, G., Buche, P., Dervin, C., Dibie-Barthélemy, J., Haemmerlé, O., Soler, L.: Fuzzy semantic approach for data integration applied to risk in food: an example about the cold chain. In: Proceedings of the 13th World Congress of Food Science and Technology, Food is Life (2006)
11. Van Rijsbergen, C.J.: Information Retrieval, 2nd edition. Dept. of Computer Science, University of Glasgow (1979)
12. Yangarber, R., Lin, W., Grishman, R.: Unsupervised learning of generalized names. In: International Conference on Computational Linguistics, pp. 1–7 (2002)
13. Platt, J.C.: In: Fast training of support vector machines using sequential minimal optimization, pp. 185–208. MIT Press, Cambridge (1999)
14. Zadeh, L.: Fuzzy sets. Information and control 8, 338–353 (1965)