Reasoning about Social Semantic Web Applications using String Similarity and Frame Logic

Markus Schatten
Faculty of Organization and Informatics
University of Zagreb
Pavlinska 2, 42000 Varaždin, Croatia
markus.schatten@foi.hr

Vijayalakshmi Kakulapati
JNT University
CSE
500 076 Hyderabad, Andhrapradesh, India
kakulapati.vijayalakshmi@gmail.com

Mirko Čubrilo
Faculty of Organization and Informatics
University of Zagreb
Pavlinska 2, 42000 Varaždin, Croatia
mirko.cubrilo@foi.hr

Abstract. *Social semantic Web or Web 3.0 application gained major attention from academia and industry in recent times. Such applications try to take advantage of user supplied meta data, using ideas from the semantic Web initiative, in order to provide better services. An open problem is the formalization of such meta data, due to its complex and often inconsistent nature. A possible solution to inconsistencies are string similarity metrics which are explained and analyzed. A study of performance and applicability in a frame logic environment is conducted on the case of agent reasoning about multiple domains in TaOPis – a social semantic Web application for self-organizing communities. Results show that the NYSIIS metric yields surprisingly good results on Croatian words and phrases.*

Keywords. social semantic web, intelligent agent, reasoning, frame logic, string similarity metric

1 Introduction

Web 3.0 or the social semantic Web should provide us with meta data that will be usefull to harvest the knowledge of large user bases. This knowledge is the result of a so called collective intelligence which has already been put to use in various applications like social bookmarking sites, recommendation systems, and semantic wikis. An inevitable property of such meta data is its internal inconsistency which is due to inconsistencies in the very social system which generated it [13, 14].

Various methods like web mining web [4, 18], context analysis through the interconnections [11], social tagging, group management, network management, social network analysis, auto completion mechanisms [15], built-in ontologies or simple taxonomies, pattern recognition, advanced audio/video processing algorithms, neural networks, or image tagging [14] have been proposed to deal with such inconsistencies. However, there hasn't been any effort to deal with syntactic inconsistencies in a frame logic semantic wiki environment that would address the very nature of meta data – strings.

Herein we will analyze the possibility of applying string similarity metrics in order to reason about semantic wiki systems. String similarity denotes the degree to which two strings are similar. We hypothesize that by using string similarity metrics we can deal with inconsistencies emerged due to inadequate or erroneous spelling. In order to test the hypothesis we will analyze metadata gathered on the ŦAOPīS system [10, 16, 17] gathered during its use in the past three years

by applying 10 different similarity metrics (namely Jaro distance, Jaro-Winkler distance, Dice's coefficient, Levenshtein distance, Damerau-Levenshtein distance, Hamming distance, Overlap coefficient, Soundex, NYSIIS and Needleman-Wunsch alignment score) and evaluate the results by using two intelligent agents implemented in the deductive frame-based language $\mathcal{F}$LORA-*2* . We will accept the hypothesis if there is at least one metric that yields better query results than the agents without it. The results of this study are constrained due to the nature of the analyzed meta data which is in most cases supplied in Croatian language.

2 Reasoning about Social Semantic Web Applications

Through the development of the World Wide Web as well as the OpenSource paradigm a lot of new information technologies were introduced that are of great interest to knowledge management [20]. Such technologies are often hidden under the term Web 2.0 or the social web, even if there is no clear agreement of what technologies build up Web 2.0. In most cases, however, lists of such technologies include wikis, folksonomies, blogs and micro blogs, social networking, podcasting, forums, social tagging or social bookmarking, as well as others.

On the other hand there is the semantic web movement [1] which tries to foster the creation of machine readable data that will allow for automated search and reasoning through the use of intelligent agents. The main idea is to provide structured semantic web ontologies written in some formalized language like OWL (the Web Ontology Language) based on description logic or $\mathcal{F}$LORA-*2* based on frame logic.

By merging these two perspectives one obtains social semantic web applications, often denoted with Web 3.0. By reasoning about such a social semantic application we mean to deduce unknown facts from the user generated knowledge base across different (often separated) systems. In order to do so, one needs to have an adequate query language like SPARQL or $\mathcal{F}$LORA-*2* to implement intelligent agents that will be able to access and query various systems. In the following we will use $\mathcal{F}$LORA-*2* (with a little help of the Python scripting language) to implement such agents.

2.1 The Case of ŧaOPīs

ŧaOPīs , which is a Web 3.0 application has a semantic wiki subsystem[1] that is based on frame logic [8], and particularly uses the $\mathcal{F}$LORA-*2* reasoning engine [22] to allow its users to query the dynamically created knowledge base. ŧaOPīs uses a syntax entitled **niKlas** that comprises the possibility to cast dynamic queries inside any wiki page. It allows users to create semantic linkages between pages as well as to tag these pages using attribute-value tags. Special attributes are used to enhance possible semantics (like class, subclass, rule etc.). Such additional tags allow the creation of metainformation in an object-oriented manner. The ŧaOPīs systems and likewise its **niKlas** syntax is work in progress that is aimed on a wide range of users. At its current version the system is still not enough user-friendly to achieve the stated goals. The main idea of hiding complex semantic technologies in the background of the system and providing users with easy-to-use graphical query builders is still to be implemented.

A semantic wiki on ŧaOPīs consists basically of an extensible set of wiki pages describing some particular content. Pages are considered to be objects having their corresponding classes, attributes as well as relations to other objects. Relations to other object are implemented through hyperlinks.

The system has been in everyday use since June 2007 for various tasks like e-learning, OpenSource project management, community management, alumni, collaborative ontology development as well as knowledge management. In order to query ŧaOPīs 's various knowledge bases we constructed a simple agent querying projects on one ore more ŧaOPīs instances. The following listing presents a simple Python script that downloads a given projects knowledge base.

```
# -*- coding: utf-8 -*-
import urllib
import re
import sys

if len( sys.argv ) > 1:
  url = sys.argv[ 1 ]
  proorg_re = re.compile( r'proorg=(.*)' )
```

[1] Please refer to [10, 16, 17] for an introduction to ŧaOPīs architecture

```
  proorg = proorg_re.findall( url )
  proorg = proorg[ 0 ]

  kb = urllib.urlopen( url )
  lines = kb.readlines()

  kb_f = open( proorg + '.flr', 'w' )

  for i in lines:
    kb_f.write( i )
  kb_f.close()
  kb.close()
  print proorg
else:
  raise Exception,'No url supplied!'
```

Using this script the following predicate was implemented in $\mathcal{F}$LORA-*2* , loading any knowledge base from an URL.

```
loadKB( ?url ) :-
  str_cat( 'python kb.py ', ?url, ?cmd )
  @_prolog( string ),
  shell_to_list( ?cmd, [ [ ?kb ] ], ?_ )
  @_prolog( shell ),
  _add(?kb).
```

Where `kb.py` is the filename of the above Python script. Agents were then defined as:

```
?- _add( loadKB ).

load_project :-
  loadKB('http://address1/?proorg=Jupiter'),
  loadKB('http://address2/?proorg=Saturn'),
  loadKB('http://address3/?proorg=Neptun').

?- load_projects.

/* Agent definition ... */
```

Whereby Jupiter, Saturn and Neptune would be the projects of interest for the agent.

3 String Similarity Metrics

String similarity metrics allow us to measure the distance between two strings of characters. The closer two strings are the more likely it is that these strings are equal (or at least that they were intended to be equal but aren't due to, for example, a spelling error).

3.1 Jaro distance

The Jaro distance metric tries to take care of typical spelling deviations [6, 7, 2]. It is defined as follows: for two strings s and t, let s' be the characters in s that are "in common with" t, and let t' be the characters in t that are "in common with" s. One could say that a character a in s is "in common with" t if the same character a appears in about the place in t. Let $T_{s',t'}$ measure the number of transpositions of characters in s' relative to t'. The Jaro similarity metric for s and t is

$$J(s,t) = \frac{1}{3} \times \left(\frac{|s'|}{|s|} + \frac{|t'|}{|t|} + \frac{|s'| - T_{s',t'}}{2|s'|} \right)$$

3.2 Jaro-Winkler distance

The Jaro-Winkler distance is an extension of the Jaro distance metric, from the work of Winkler in 1999 [21, 2]. This extension uses a modified weighting mechanism of poorly matching string pairs s, t that have a common prefix. The output score is then adjusted as:

$$JW(s,t) = J(s,t) + (l * p * (1.0 - J(s,t)))$$

Where l is the length of common prefix at the start of the string, p is a constant scaling factor for how much the score is adjusted upwards for having common prefix's. This adjustment gives better ratings to strings that match from the beginning for a set prefix length, as the (normal) Jaro distance.

3.3 Dice's coefficient

Dice coefficient [2] is a term based similarity measure. It is defined as twice the number of characters common to compared strings divided by the total number of characters in both tested strings. The Coefficient result of 1 indicates identical vectors as where a 0 denotes orthogonal vectors.

$$D(s,t) = \frac{2 * c(s,t)}{n(s) + n(t)}$$

Where $c(s,t)$ is the number of common characters in strings s and t, and $n(s)$ and likewise $n(t)$ are the numbers of characters in the respective string.

3.4 Levenshtein distance

The Levenshtein distance [9, 2] is a distance function that counts the number of needed edit operations to equate the strings. The distance is given as

the minimum edit distance which transforms string s into string t. Levenshtein defines four edit operations with corresponding costs as follows:

- Copy character from s over to t (cost 0) $D(i-1, j-1) + d(s_i, t_j)$
- Delete a character in s (cost 1) $D(i, j-1) + 1$
- Insert a character in t (cost 1) $D(i, j) = \min D(i-1, j) + 1$
- Substitute one character for another (cost 1) $D(i-1, j-1) + d(s_i, t_j)$

Whereby $d(i, j)$ is a function and $d(c, d) = 0$ if $c = d$, or else $d(c, d) = 1$.

3.5 Damerau–Levenshtein distance

The Damerau–Levenshtein distance is an extension to the previous that includes an additional edit operation: transposition of two characters (cost 1) [3].

3.6 Hamming distance

The Hamming distance [5, 2] is defined as the number of bits which differ between two binary strings. It is the number of bits which need to be changed to turn one string into the other. For example the bit strings 11011010 and 11001100 have a hamming distance of 3 bits, (as three bits are different). This simple hamming distance function can be extended into a vector space approach where the characters within a string are compared, counting the number of characters in the same positions.

3.7 Overlap coefficient

The overlap coefficient [2] is a set based similarity measure. If a set X is a subset of Y or the converse then the similarity coefficient is a full match, which is similar to previously mentioned Dice coefficient. The overlap coefficient is defined as:

$$O(s, t) = \frac{|s \cap t|}{min|s|, |t|}$$

3.8 Soundex

Soundex is a phonetic indexing scheme, often used in genealogy [2]. This approach is used to match individuals names and as such has not been provably applied to a more general context.

Soundex allows phonetic misspellings to be evaluated easily, for instance the names Mark, Marko, Marco and Markus are often genealogically the same person. This is a term or word based evaluation where each term is given a Soundex code. Each soundex code consists of a letter and three numbers between 0 and 6, for example "Cubrilo" is "C164". The letter is always the first letter of the surname. The numbers hash together the rest of the name. This approach is very promising for disambiguation of transliterated/misspelled names, i.e non english names represented as ASCII are frequently misrepresented. The digits are based on the consonants as in the following listing, (this can differ from implementation to implementation):

1. B,P,F,V
2. C,S,K,G,J,Q,X,Z
3. D,T
4. L
5. M,N
6. R

The vowels are excluded. If two or more adjacent letters, not being not separated by a vowel, have the same numeric value, only one is used, which also applies if the first (which is used for the letter in the code), and the second letter in name have the same value. The second letter would not be used to generate a digit. If there are not three digits after the consonants are converted, zeros are added to the code. For example the name Vijaya has no consonants after the V and J, so the soundex code would be V200.

3.9 NYSIIS

In 1970 the New York State Identification and Intelligence project headed by Robert L. Taft published the paper "Name Search Techniques" [19] in which he compared Soundex with a new phonetic routine (NYSIIS - New York State Identification

and Intelligence Algorithm). NYSIIS was designed through rigorous empirical analysis. The algorithm is as follows:

1. remove all 'S' and 'Z' chars from the end of the surname
2. trans-code initial strings
   $MAC \Rightarrow MC$ $PF \Rightarrow F$
3. Trans-code trailing strings as follows,
   $IX \Rightarrow IC$ $EX \Rightarrow EC$ $YE, EE, IE \Rightarrow Y$
   $NT, ND \Rightarrow D$
4. trans-code 'EV' to 'EF' if not at start of name
5. use first character of name as first character of key
6. remove any 'W' that follows a vowel
7. replace all vowels with 'A'
8. trans-code 'GHT' to 'GT'
9. trans-code 'DG' to 'G'
10. trans-code 'PH' to 'F'
11. if not first character, eliminate all 'H' preceded or followed by a vowel
12. change 'KN' to 'N', else 'K' to 'C'
13. if not first character, change 'M' to 'N'
14. if not first character, change 'Q' to 'G'
15. trans-code 'SH' to 'S'
16. trans-code 'SCH' to 'S'
17. trans-code 'YW' to 'Y'
18. if not first or last character, change 'Y' to 'A'
19. trans-code 'WR' to 'R'
20. if not first character, change 'Z' to 'S'
21. trans-code terminal 'AY' to 'Y'
22. remove trailing vowels
23. collapse all strings of repeated characters
24. if first char of original surname was a vowel, append it to the code

3.10 Needleman-Wunsch distance

This approach is known by various names, Needleman-Wunsch, Needleman-Wunsch-Sellers, Sellers and the Improving Sellers algorithm [12, 2]. It is similar to the Levenshtein distance, except that it adds a variable cost adjustment to the cost of a gap to the distance metric. A gap can be either an insertion or a deletion. We can view the Levenshtein distance as the Needleman-Wunch distance with $G = 1$.

- Substitution/Copy: $D(i-1, j-1) + d(s_i, t_j)$
- Insertion: $D(i, j) = minD(i-1, j) + G$
- Deletion: $D(i, j-1) + G$

Where G is the "gap cost" and $d(c, d)$ is again an arbitrary distance function on characters (e.g. related to typographic frequencies, amino acid substitutability, etc).

4 Methodology

Having the various similarity metrics defined we wanted to find out how do such metrics behave when implemented in intelligent agents reasoning about some given domains. In order to obtain comparable data we decided to use existing meta data from the ŧaOPīs system. For over 3 years, at the time of writing this article, student from the Faculty of Organization and Informatics have been using ŧaOPīs in various e-learning environments. One such environment includes a Knowledge management (KM) course, where teams of students collaborated on various KM topics with the goal to formalize their newly learned knowledge. The result of this collaboration is a set of KM related semantic wiki projects, each including plenty of various meta data [10]. Since the projects were more or less related to similar topics, we choose 10 that had most provided meta data and were well structured. The projects knowledge bases comprise a total of 32792 meta data statements defined over 5003 objects.

Two intelligent agents were implemented amalgamating the knowledge bases of the chosen projects and trying to obtain results using the described similarity metrics. The first agent was a simpler one trying to infer if there were wiki pages with

similar titles. The second agent was more sophisticated, and tried to infer additional attributes of some object based on the assumption that two objects define the same concept.

The similarity metrics were implemented as a Python script, the similarity predicate was implemented in XSB Prolog and the two agents were implemented in $\mathcal{F}$LORA-*2* . Due to a combinatoric explosion of strings that had to be compared we made use of XSB's and likewise $\mathcal{F}$LORA-*2* 's tabling facilities and implemented three simple heuristics before applying the similarity algorithms:

1. If two strings are equal they are similar as well.
2. For a string to be considered at all it has to be at least 3 characters long.
3. For two strings to be compared their length shouldn't differ for more then 3 characters.

Additionally we made use of $\mathcal{F}$LORA-*2* 's class optimization facilities in order to keep computing reasonable. The first agent was implemented as follows:

```
test_1( ?x, ?y, ?metric ) :-
  ?_o1[ title->?x ]@amalgam,
  ?_o2[ title->?y ]@amalgam,
  ?_o1 \== ?_o2,
  similar( ?x, ?y, ?metric )@amalgam,
  insert{
    sim_title( ?metric, ?x, ?y )@amalgam
  }.
```

Whereby the variables $?x$ and $?y$ are bound to two similar titles of different objects bound to the variables $?\_o1$ and $?\_o2$, the variable $?metric$ has to be bound to one of the similarity metrics (jaro, jaro-winkler, dice, levenshtein, damerau-levenshtein, hamming, overlap, soundex, NYSIIS, needleman-wunsch), $similar/3$ is the similarity predicate (e.g. it succeeds if $?x$ and $?y$ are similar according to metric $?metric$, and *amalgam* is the dynamic module created from the chosen project's knowledge bases. Since the computation of results takes a certain amount of time we used persistent modules in order to store computed results (in this case in the form of $sim\_title/3$ atoms) in a database for later analysis.

The second agent's implementation is similar but more complex:

```
test_3( ?x, ?y, ?at1, ?at2, ?metric ) :-
  ?_o1:?x@amalgam,
  ?_o2:?y@amalgam,
  ?_o1 \== ?_o2,
  similar( ?x, ?y, ?metric )@amalgam,
  ?at1 = collectset{
    ?_at1 |
      ?_o1[ ?_at1->?_ ]@amalgam
  },
  ?at2 = collectset{
    ?_at2 |
      ?_o2[ ?_at2->?_ ]@amalgam
  },
  insert{
    inf_att( ?metric, ?x, ?y, ?at1, ?at2 )
  }.
```

Again variables $?x$ and $?y$ are bound to two similar strings defining two class definitions of two different objects $?\_o1$ and $?\_o2$, but now the agent tries to reason as follows: *if some object defines a concept that is similar to a concept some other object defines, then these two objects share common attributes.* These attributes are bound to the logic variables $?at1$ and $?at2$, and stored together with the class definition strings in $inf\_att/4$ atoms.

5 Results & Discussion

After running the two defined agents obtained results were evaluated as follows: for each similarity metric the total number of results was counted. Each result was inspected manually if it was semantically valid or not. Table 1 summarizes the results for agent 1:

Table 1: The number of valid, invalid and total results for agent 1

Metric	Valid	Invalid	Total
Jaro	24	23	47
Jaro-Winkler	25	56	81
Dice	5	0	5
Levenshtein	27	12	39
Damerau-Levenshtein	27	12	39
Hamming	17	4	21
Overlap	5	0	5
Soundex	28	79	107
NYSIIS	19	0	19
Needleman-Wunsch	51	538	589

As one can see, only the Dice's coefficient, the Overlap coefficient and the NYSIIS had a 100% accuracy. Hereby we need to mention that the 5

results, obtained by both using Dice's and Overlap coefficients, were actually equal strings. Thus the same results would be obtained if not using any string similarity at all. On the other hand the Needleman-Wunsch alignment score yielded by far the most results, but unfortunately most of them were wrong. Thus in the first agent NYSIIS takes the lead.

Table 2: The number of valid, invalid and total inferred attributes for agent 2

Metric	Valid	Invalid	Total
Jaro	584	38	622
Jaro-Winkler	611	164	775
Dice	440	0	440
Levenshtein	697	0	697
Damerau-Levenshtein	755	0	755
Hamming	548	0	548
Overlap	440	0	440
Soundex	695	161	856
NYSIIS	562	0	562
Needleman-Wunsch	787	207	994

Table 2 shows the results of the second agent. This time Dice's coefficient, Levenshtein distance, Damerau-Levenshtein distance, Hamming distance, Overlap coefficient as well as NYSIIS had a 100 % accuracy. Again Dice's and Overlap coefficient's results were obtained due to solely equal strings. The other accurate string similarity metrics had both equal and non-equal strings from which conclusions were drawn. Thus in this case the Damerau-Levenshtein distance obtained the best results for agent 2.

The obtained results indicate that the NYSIIS metric shows surprisingly good results for Croatian words and short phrases. Even if the Levenshtein, Damerau-Levenshtein and Hamming distance did have an 100 % accuracy in the second agent, we find the results of the first agent more conclusive. The first agent was looking for similarities in a by far greater string set then the second, which is due to the fact that more objects have a title attribute (which is predefined) than they are classified into some class.

6 Conclusion & Future Research

In this paper we analyzed ten similarity metrics using two different agents: one trying to find objects with equal or similar titles (thus indicating that one and the same object is described on two different pages), and a second trying to suggest attributes for a some given class based on the similarity of class names.

After analyzing the different similarity metrics we conclude that the NYSIIS metric shows the best results in dealing with syntactic inconsistencies (emerged mostly due to inadequate spelling) on the given data set of ŧaOPīs . This conclusion can provide us with the means to implement various suggestion mechanisms. Thus future research will include the implementation of title, attribute and tag suggestion mechanisms based on the NYSIIS metric.

References

[1] Berners-Lee, T., Hendler, J., and Lassila, O. The semantic web. *Scientific American Magazine* (May 2001).

[2] Chapman, S. String similarity metrics for information integration. Available at `http://www.dcs.shef.ac.uk/~sam/stringmetrics.html`, accessed 1^{nd} April 2010.

[3] Damerau, F. J. A technique for computer detection and correction of spelling errors. *Communications of the ACM 7*, 3 (March 1964), 171–176.

[4] Dringus, L. P., and Ellis, T. Using data mining as a strategy for assessing asynchronous discussion forums. *Computers & Education 45* (2005), 141–160.

[5] Hamming, R. W. Error detecting and error correcting codes. *Bell System Technical Journal 26*, 2 (1950), 147–160.

[6] Jaro, M. A. Advances in record linking methodology as applied to the 1985 census of Tampa Florida. *Journal of the American Statistical Society 64* (1989), 1183–1210.

[7] Jaro, M. A. Probabilistic linkage of large public health data file. *Statistics in Medicine 14* (1995), 491–498.

[8] Kifer, M., Lausen, G., and Wu, J. Logical foundations of object-oriented and frame-based languages. *Journal of the Association for Computing Machinery 42* (May 1995), 741–843.

[9] Levenshtein, V. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady 10* (1966), 707–710.

[10] Maleković, M., and Schatten, M. Leadership in team based knowledge management - an autopoietic information system's perspective. In *19th Central European Conference on Information and Intelligent Systems – CECIIS2008 Conference Proceedings* (September 2008), B. Aurer and M. Bača, Eds., Faculty of Organization and Informatics, pp. 47–52.

[11] Mehler, A. Structural similarities of complex networks: a computational model by example of wiki graphs. *Applied Artificial Intelligence 22* (2008), 619–683.

[12] Needleman, S. B., and Wunsch, C. D. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology 48*, 3 (1970), 443–453.

[13] Schatten, M. *Programming Languages for Autopoiesis Facilitating Semantic Wiki Systems.* PhD thesis, University of Zagreb, Faculty of Organization and Informatics, Varaždin, Croatia, February 2010.

[14] Schatten, M., Bača, M., and Ivanković, M. Public interfaces as the result of social systems structural coupling. In *Proceedings of the 1st International Conference on Information Society and Information Technologies ISIT 2009* (October 2009), M. Mertik and N. Damij, Eds., Faculty of information studies in Novo mesto.

[15] Schatten, M., Maleković, M., and Rabuzin, K. Inconsistencies in semantic social web applications. In *Proceedings of the 20th Central European Conference on Information and Intelligent Systems* (2009), B. Aurer, M. Bača, and K. Rabuzin, Eds., Faculty of Organization and Informatics.

[16] Schatten, M., Čubrilo, M., and Ševa, J. A semantic wiki system based on f-logic. In *19th Central European Conference on Information and Intelligent Systems – CECIIS2008 Conference Proceedings* (2008), B. Aurer and M. Bača, Eds., Faculty of Organization and Informatics, pp. 57–61.

[17] Schatten, M., and Žugaj, M. Organizing a fishnet structure. In *29th International Conference Information Technology Interfaces Proceedings* (Cavtat-Dubrovnik, Croatia, June 25 — 28 2007), pp. 81–86.

[18] Spangler, W. S., Kreulen, J. T., and Newswanger, J. F. Machines in the conversation: Detecting themes and trends in informal communication streams. *IBM Systems Journal 45*, 4 (2006), 785–799.

[19] Taft, R. L. *Name Search Techniques.* New York State Identification and Intelligence System, Albany, New York, 1970.

[20] Žugaj, M., and Schatten, M. Informacijski sustav za upravljanje znanjem u hipertekst organizaciji. *Ekonomski vjesnik 21*, 1-2 (2008), 19–30.

[21] Winkler, W. E. The state of record linkage and current research problems. *Statistics of Income Division, Internal Revenue Service Publication R99*, 04, 1999.

[22] Yang, G., Kifer, M., and Zhao, C. Flora-2: A rule-based knowledge representation and inference infrastructure for the semantic web. In *Second International Conference on Ontologies, Databases and Applications of Semantics (ODBASE) Proceedings* (Catania, Sicily, Italy, November 2003).