A Comparative study on Syntax Matching Algorithms in Semantic Web

V.P.SUMATHI[1], Dr.K.KOUSALYA[2], R.KALAISELVI[3]
[1] Department of Computer Science and Engineering
Kumaraguru College of Technology, Coimbatore, Tamilnadu, INDIA
[2] Department of Computer Science and Engineering
Kongu Engineering College, Perundurai, Tamilnadu, INDIA
[3] Department of Computer Science and Engineering
Sri Shakthi Institute of Engineering and Technology, Coimbatore, Tamilnadu, INDIA
[1]sumathi.vp.cse@kct.ac.in [2]kouse@kongu.ac.in [3]kalaiselvi.rangaraj@gmail.com

Abstract: - In semantic web, extraction of meaningful information involves many tedious processes due to similarity between the information, context of the words used, structure similarity and relationship between the words. Ontology helps to understand the context of heterogeneous information available in the web. The domain specific ontologies can be merged to extract integrated information from various semantic websites. Different algorithms are in practice to find similarity between class names that are exist in different ontologies. The class names which are syntactically and semantically equal are identified and merged to produce global ontology that can be used for information retrieval. In this paper we have implemented and compared various syntax matching algorithms and identified the best syntax matching algorithm appropriate for semantic web environment. Performances of identified algorithms are analyzed and evaluated with respect to precision, recall and F-measure.

Keywords:- Syntax matching algorithm, Semantic Web, Information theory, Heterogeneity, Similarity measures.

1 Introduction

Semantic Web is a mesh of information linked up in such a way that easily processable by machines, on a global scale. Semantic web drives the evolution of the current web by enabling users to find, share, and combine information more easily. Semantic web is constructed with the help of ontology. The prevalent definition presented by Gruber: Ontology is a formal explicit specification of a shared conceptualization. Extracting information from semantic webs is a tedious process due to their heterogeneity. The ontologies of different websites are merged to extract information. One of the important steps in merging two domain specific ontologies is to identifying the input ontologies which are belonging to same domain or different domain. Ontologies belonging to two different domains cannot be merged together as such. It is required to identify every classes, sub classes, object properties, data properties and relationship between the classes described in ontologies. It is also necessary to find the similarity between both ontologies which in turn identify the similarity between object, class, data properties, object properties and relationships between the classes. Various similarity measure algorithms can be employed to identify similarity between the class names. Currently, in semantic web similarity measure is classified as syntax similarity measure and semantic similarity measure. Syntax similarity measure reflects the relation between the patterns of the two strings where as semantic similarity measure is based on the meaning of class name and context, which can be obtained with the help of pre-constructed libraries. For measuring syntax similarity, around 67 similarity measure algorithms are available for information retrieval [8]. From the list, some similarity measures are more appropriate for identification of string similarity.

This paper concentrates only on syntax matching algorithms and a study on seven syntax similarity measures is carried out. Similarity measures are used for identifying syntax similarity between two class names, object names, instances and relationships between names. In this paper name refers to either class name, object name available in the ontologies. But few number of research works are employed more than one similarity measures for finding similarity between class names. Similarity identification is a tedious and time consuming job. Moreover, if the result justification is done using only syntax matching algorithms, it may not be precise. This is due to different ontologies describing different data with same class names or different class name describing same data. Both cases occur in semantic web. It is necessary to go for semantic matching in addition to syntax matching measures. Most of the previous works on ontology merging did not address the way to find efficient syntax matching algorithms. In this paper, the term used as name refers to either class name or object name available in the ontologies.

The main contributions of this paper are

1. Implements seven different syntax matching algorithms namely Hamming distance, Levenshtein distance, Dameran – Levenshtein distance, Jaro Winkler distance, Optimal String Alignment Algorithm, N-Gram string matching algorithms and Soundex.
2. Compared the performance of all seven algorithms with respect to precision, recall and F-measure.
3. Helps the researcher to understand and choose the best syntax similarity measure.

This paper is organized as follows. Section 2 presents the related works and information about choice of various syntax matching algorithms used by many researchers. Section 3 explains various syntax matching algorithms with appropriate examples and information theory. Section 4 provides discussion of results obtained through experiments. Section 5 includes conclusion and recommendations for researchers to choose best syntax similarity measures based on their domains.

2 Related Work

Similarity measures are used in variety of fields like comparison of strings, symbolic word, patterns, images, DNA sequences and codes. Identifying the similarity between the pair is one of the problematic tasks. All similarity measures cannot be used to identify the similarity in all cases due to their own restrictions. Some measures are used only for numeric comparison and some other for string comparison. Only countable number of techniques is used for both numeric and string. There are various types of models for identifying similarities. In semantic web, widely used model is probabilistic model where the calculated similarity value lies between zero and one. Information theory of similarity is playing a vital role behind the similarity measures. The definition of similarity can be discussed in the form of intuitions and assumptions [8]. The edit distance measures is widely used to identify the similarity in various fields like biological sequences, schema mapping, text retrieval, document clustering, ontology mapping, ontology alignment, ontology merging and so on. In this work, only strings similarity has been analysed. After identifying the similarity between class names, the equivalence classes are merged to get the global ontology which is used to extract information from various semantic webs.

Identifying the syntax similarity is the initial step in ontology merging for performing merging operation on two domain specific ontologies. In this context the syntax similarities are measured by using variety of similarity algorithms. Mostly edit distance techniques are used which is based on three operations insertion, deletion and substitution. These operations are performed while transforming a class name from one to another belongs to same domain. The sub polynomial approximation algorithm is used to improve the performance of edit distance algorithm. Due to this approach edit distance algorithm runs in near-linear time [1]. In the field of string comparison, to improve the performance of edit distance algorithm first step is to compress the strings, and then to compute the edit distance between the compressed strings [7]. SEDIL software was released to learn edit distance calculations [17]. In string matching, two different input strings represent the same context are considered as equal. Based on the requirement names may be given elaborately or simple manner. Hamming distance algorithm is used to identify the similarity between the names are equal in length. In query processing context the field names were compared by using Hamming distance technique [2]. The parallel string matching concept was used to reduce the computation time of Hamming distance algorithm [28].

The word comparison was done by using Levenshtein distance in dictionary searching scenario [24]. In ontology matching techniques, both Levenshtein distance and soundex were used in companied manner [16]. In recent research papers, Dameran Levenshtein distance was used to identify the similarities. This is the improved version of Levenshtein distance. The character comparison was done by considering transposition of the character in both names. Most of the ontology merging papers used Jaro Winkler similarity measures. By using

Jaro Winkler approximation the similarity values calculated lies between zero to one. This technique comes under probabilistic model based similarity measures. In semantic web all types of operations like ontology mapping, alignment and merging were performed by using Jaro Winkler algorithm for similarity measures [4,10,11,13,19,20,27,32].
In schema matching the class names and object names were compared by using N-gram similarity measures [3,6]. Most of the research related to medicine field use the N-gram string matching algorithm [15,25,30,33]. Ontology alignment, string matching, automatic spelling correction, automatic key phrase extraction, indexing techniques and semantic layer constructions were used the N-gram similarity measures for calculating similarity between names as discussed in [12,18,21,23,30,31]. In natural language processing the phonetic matching approaches for Indian languages was done by using N-gram and Soundex similarity measures. These two techniques can be used to perform spell checking and correction for given two input class names [26]. Some of the ontology merging techniques which are working based on N-grams similarities measures combined with Dice coefficient similarity measures for calculating similarity values [9]. Most of the phonetic sound based researches used Soundex similarity measuring technique to correct the spelling error occur [5,14,16,26]. The natural language processing field involves variety of functions like translation of words between languages, the pronunciation difference between continents. These were successfully handled by Soundex technique [22]. The above said reference article provides most widely used techniques in various fields. Most of the researcher may not be aware of all these techniques at the initial stage of their research work. This paper provides idea about all these algorithms with detailed explanations including examples in forthcoming section.

3 Syntax Matching Algorithms

Let m and n are the input names. The similarity between m and n is measured by the ratio between the amount of information needed to provide the commonality of m and n and the information needed to fully describe m and n. The similarity is calculated using the formula in equation (1). In this equation I(x) is the amount of information contained in a proposition x. The function f is explained in equation (2). Commonality p is calculated by taking negative logarithm of the probability of the statement shown in equation (3) and q provides description which is calculated by taking logarithm of the probability of the description on m and n. The difference between m and n is calculated by subtracting commonality values from the description is given in equation (4). If both names are identical then p value is greater than zero therefore f(p,p) = 1. If there is no common character occurrence between the names then p becomes zero, f(0,q) = 0.
In this paper, various syntax matching algorithms are implemented and their performance differences are analysed. Edit distance is a way to measure similarity between two names by counting the minimum number of operations required to transform one name into the other. The possible operations are insertion, deletion and substitution. Edit distance provides basic for most of the syntax similarity algorithms. The syntax similarity algorithms suitable for string comparison are (i)Hamming distance, (ii)Levenshtein distance, (iii)Dameran – Levenshtein distance, (iv)Jaro Winkler distance, (v)Optimal String Alignment Algorithm, (vi)N-Gram string matching algorithms and (vii)Soundex. These are widely used in semantic web environment.
Hamming distance algorithm is used if both string names are equal in length; otherwise this method is not applicable to calculate similarity. Transpositions of characters are not taken into account and this method never detects human typing errors. In Levenshtein distance similarity measure the number of insertion, deletion and substitution operations are counted when strings are not equal in length. Here also transpositions of character occurrences are not considered. In Dameran-Levenshtein distance measure transpositions are considered while transforming from one string to another.

$$Sim(m,n) = f(I(common(m,n), I(discription(m,n)) \qquad (1)$$

$$f(p,q) = \{(p,q) \mid p >= 0, q > 0, p <= q \} \qquad (2)$$

$$p = -\log P(common(m,n) \qquad (3)$$

$$Difference(m,n) = I(discription(m,n) - I(common(m,n) \qquad (4)$$

The transpositions of character occurrence should be within the window size. The window size is calculated by considering length difference between two given input names. In most of the research papers related to ontology mapping and merging, the Jaro Winkler distance similarity measure is used. In this algorithm, characters of first string are compared to the characters of second string at unaligned position. The matching is performed only within the match range calculated between two strings. In optimal string alignment algorithm dynamic programming technique is used to measure the similarity. In N-gram similarity the given input string is divided into number of substrings of length N. The similar substrings of length N are counted. In Soundex algorithm spellings mistake occurs due to pronunciation are eradicated. So the strings given by various continents are compared easily and similarities are identified. These algorithms are explained in the following sections.

3.1 Edit Distance

Edit distance is a way to measure similarity between two strings by counting the minimum number of operations required to transform one string into the other. The edit distance involves three different operations, insertion(I), deletion(D) and substitution(S). Consider two strings of length i, j where string 1 = s1[1…..i] , string 2 = s2[1..j]. The numbers of operations performed while transforming string1 to string2 are counted.

I : if i < j by one then an insertion operation will take place
D : if i > j by one then a deletion operation will take place
S : if i = j and s1[i] != s2[j] then substitution operation will take place

Various combinations of operations are involved while transforming string s1 to s2. This can be represented as combination of I, D and S. Total number of operations can be calculated by using the formula N(I) + N(D) + N(S) where N means number of insertions, deletions and substitutions operations and these values various from 1 to L where L is max(length(s1),length(s2)). The number of occurrence of I and D are equal to difference between lengths of two strings. The maximum number of occurrence of S depends on the minimum length of the string. From the table1 the combination of operation possible while transforming from one class name to another can be easily identified.

Table 1 Operations Vs String lengths

Operations Performed	**Comparison of string length**		
	$L_1 = L_2$	**$L_1 > L_2$**	**$L_1 < L_2$**
Insertion	Not possible	Not possible	Possible
Deletion	Not possible	Possible	Not possible
Substitution	Possible	Possible	Possible

L_1 – String1 length and L_2 – String2 length

3.2 Hamming Distance

The Hamming Distance (HD) is calculated by counting the number of substitutions take place while transforming the names. While comparing each and every character position, no substitution is needed if both characters are same, otherwise the character in name1 is substituted by character in name2 and it is counted as one substitution. This type of measurement is suitable only when both names are in equal length. Strings of unequal length will leads to high cost of substitution. The difference between two names are equal to the number of substitutions taken place while converting from one class name to another class names

Differences (name1, name2) = Length of the string - Number of characters similar

Consider the names “toned” and “roses” where second and fourth character are same and the character position 1,3,5 are different. So the numbers of substitutions required is three while transforming “toned” into “roses”.

Difference(“toned”,”roses”) =
I(description(“toned”,”roses”)) –
I(common(“toned”,”roses”))

The difference is equal to 3. The similarity is the ratio between commonality and description which is 2/5 equal to 0.4. Let us consider the names people and people, the differences between the names are zero. The similarity value between the names people and people is 6/6 equal to 1.

$$x \in name1\, and\ y \in name2\, at\ position\ t$$

$$HD(x,y) = \sum \begin{cases} \exists x, \exists y, t & if\ x = y \quad 0 \\ otherwise & \quad\quad\quad 1 \end{cases} \tag{5}$$

3.3 Levenshtein Distance

This algorithm identifies the similarity between strings by performing minimum number of insertions, deletion and substitution operations while transforming from name1 to name2. This algorithm can measure the similarity value for unequal length class names.

Let L_1, L_2 be the length of the name1, name2. While comparing the lengths I, D and S operations are possible to take place. The length differences between the names are used to calculate the number of insertion and deletion operations performed while transforming from name1 to name2. If $L_1 < L_2$, insertions and substitution will be performed. For $L_1 > L_2$ deletion and substitution will be performed.

Case $L_1<L_2$:

No. of insertion operations $N(I) = L_2 – L_1$

Case $L_1>L_2$:

No. of deletion operations $N(D) = L_1 – L_2$

Number of substitutions S is calculated using equation (6). The Levenshtein distance between the given two class names are calculated using the equation (7). For example consider the names colour and color. The number of operations needed to transform from one to another is calculated. Lev(colour, color) is ((2*5)-(2*4))/2 + (6-5) = 2.

3.4 Dameran –Levenshtein Distance

This technique is similar to Levenshtein distance, where transpositions of character occurrence should be considered within the window size. Window size is calculated by considering length difference between two given names. The original motivation is to measure distance between human misspelled names and original names to improve performance of information retrieval applications. Let L1 and L2 be the length of name1 and name2. The Dameran-Levenshtein distance between the class names are calculated using the equation (8).

c	o	l	o	u	r
c	o	l	o	r	

Consider colour and color where window size is 1. The nth character in name1 is compared to nth and n+1th character position of the name2. Due to window size one, the letter ‘r’ of name2 is compared with both ‘u’ and ‘r’ in name1. Instead of substituting ‘u’ by ‘r’ the deletion of ‘u’ take place at position 5 in name1. Hence the number of operations needed to transform from one to another is calculated using equation (8). Dam-Lev (colour, color) is (10-(2*5))/2 + (6-5) = 1.

3.5 Jaro Winkler Distance

For calculating string similarity it is necessary to calculate two important values. The first one is match range and second one is number of transpositions. The match range means the number of character positions are considered for a single character in name1 to find the matches in name2. The match range is calculated using the equation (9). The character matches are checked within the match range. For each character encountered in the first string, it is matched to the first unaligned character in the second string which is an exact match. If no such occurrence within the match range the character is not matched. The numbers of transpositions are calculated by counting the number of character which are not matched in the exact positions but matched within the match range. For similarity measure consider only half of the transpositions.

$x \in name1\, and\ y \in name2\ at\ position\ \ t$

$$S = Min(l1, l2) - \sum \begin{cases} \exists x, \exists y, t \quad if\ \ x = y & 0 \\ otherwise & 1 \end{cases} \qquad (6)$$

$$Lev(name1, name2) = \frac{2 * Min(L1, L2) - 2 * \sum \begin{cases} \exists x, \exists y, t \quad if\ \ x = y & 0 \\ otherwise & 1 \end{cases}}{2} + Abs(L1 - L2) \qquad (7)$$

$x \in name1\ and\ \ y \in\ name2\ at\ \ position\ t\ \ which\ \ \text{var}\,ious\ \ from 1 to\ window size$

$$Dam - Lev(name1, name2) = \frac{2 * Min(L1 - L2) - 2 * \sum \begin{cases} \exists x, \exists y, t \quad if\ x = y & 0 \\ otherwise & 1 \end{cases}}{2} + Abs(L1 - L2) \qquad (8)$$

$$Matchrange = \frac{Max(name1.length(), name2.length())}{2} - 1 \qquad (9)$$

The Jaro Distance between the given two names are calculated by subtracting Jaroproximity from 1.
JaroDistance (name1.name2) = 1 - Jaroproximity(name1.name2)
Jaroproximity between the given two names are calculated using the equation (10). Winkler modifications consider the prefix substring matches of given two names which boost the similarity scores between name1 and name2 that matches character by character in at the starting index. The prefix size is calculated by considering the number of character exactly matched by original index from the starting index 0.
Jarowinkler proximity = Jaroproximity + 0.1*prefix size * (1.0 – Jaroproximity)

j	o	n	e	s		
1	2	3	-	4		
j	o	h	n	s	o	n

For example consider the names jones and johnson. The match range value for the given two names is 2.
Matchrange = Max(5,7)/2 -1 =7/2 - 1 = 3- 1 = 2
So, the nth character in one name will be compared to nth and n+1th character of other name. If the character matches in the same position index or within match range then number of matched character is counted as one. The transpositions are calculated by considering the nth character is equal to any one positions like n+1, n+2, n+3 …. (n + match range) character in other name then transposition count is incremented by one.
Matchrange = 2
Number of matches = 4
Half of the transpositions = ½ = 0
Prefix size = 2 (only first two characters are exactly matched)
Jeroproximity (jones,Johnson)
= 1/3 *(4/5+4/7+ (4-0)/4) = 0.790

Jarowinkler proximity = 0.790+0.1*2*(1-0.790)
= 0.832

3.6 Optimal string alignment algorithm

In this algorithm dynamic programming techniques is used to find the similarity between given class names. Dynamic programming is a technique to find solution to any big problem by combining solutions of the similar sub problems exists in that problem. It is the tabular computation of D(n,m). Using bottom up approach compute D(i,j) for small values of i and j where i, j value varies from 1 to length of the input names. Compute the large D(i,j) based on previously computed smaller values of D(i,j). Computation of D(i,j) as given in equation (11).
Consider the example name1 = execution and name2 = intention. In table 2, name1 occupy column wise cell and name2 occupy row wise cell positions. Initially all cells are occupied by index value of characters in names. Each matrix element D(i,j) is calculated using the equation (11). If insertion or deletion operation is possible, the cell value is computed by selecting minimum value of left, down and diagonal cell values plus 1.

$$Jaroproximity(name1, name2) = \frac{\begin{array}{c} percentage\ of\ name1\ matched + percentage\ of\ name2\ matched + \\ Percentage\ of\ matches\ that\ were\ not\ transposed \end{array}}{3} \quad (10)$$

$$where$$

$$percentage\ of\ name1\ matched = \frac{matches(name1, name2)}{name1.length()}$$

$$percentage\ of\ name2\ matched = \frac{matches(name1, name2)}{name2.length()}$$

$$percentage\ of\ matches\ that\ were\ not\ transposed = \frac{matches(name1, name2) - tranpositions(name1, name2)}{matches(name1, name2)}$$

$$D(i, j) = Min\begin{bmatrix} D(i-1, j)+1\ for\ deletion \\ D(i+j-1)+1\ for\ inserion \\ D(i-1, j-1)+\begin{Bmatrix} 2 & if & x(i) \mathrel{!=} y(j) \\ 0 & if & x(i) = y(j) \end{Bmatrix} \quad for\ \ substitution \end{bmatrix} \quad (11)$$

Optimal string alignment algorithm

Input : string1 length(n) , string2 length(m)
Output : D(n,m) cell provides similarity value

```
{   initialize D(i,0) = i  D(j,0) = j
// compute recurrence relation
for each i = 1..n
for each j=1..m
compute D(i,j) using equation (11)
// Terminate after the computation of D(n.m)
// adding back trace to find minimum edit distance
D(n,m) provides similarity value
Back trace from D(n,m) to D(0,0)
```

$$Back\,trace\,of\;D(i,j)=\begin{cases}\leftarrow & insertion\;\;operation\,performed\\ \downarrow & Deletion\;\;operation\,performed\\ diagonal\;\;arrow & substituion\,operation\,performed\end{cases}$$

$$where\;n>i>0,\;\;m>j>0$$

}

Table 2 Optimal string alignment algorithm output matrix

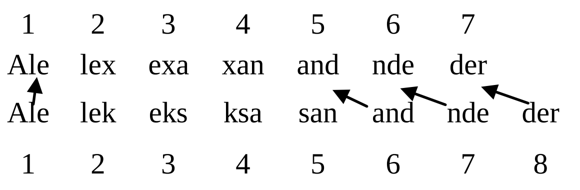
n	9	↓8	↙←↓9	↙←↓10	↙←↓11	↙←↓12	↓11	↓10	↓9	↙ **8**
o	8	↓7	↙←↓8	↙←↓9	↙←↓10	↙←↓11	↓10	↓9	↙ **8**	←9
i	7	↓6	↙←↓7	↙←↓8	↙←↓9	↙←↓10	↓9	↙ **8**	←9	←10
t	6	↓5	↙←↓6	↙←↓7	↙←↓8	↙←↓9	↙ **8**	← 9	←10	←↓11
n	5	↓4	↙←↓5	↙←↓6	↙←↓7	↙←↓**8**	↙←↓9	↙←↓10	↙←↓11	↙ ↓10
e	4	↙ 3	←4	↙ ←**5**	←**6**	←7	←↓8	↙←↓9	↙←↓10	↓9
t	3	↙←↓ 4	↙←↓**5**	↙←↓6	↙←↓7	↙←↓8	↙ 7	←↓8	↙←↓9	↓8
n	2	↙←↓ **3**	↙←↓4	↙←↓5	↙←↓6	↙←↓7	↙←↓8	↓7	↙←↓8	7
i	**1**	↙←↓ 2	↙←↓3	↙←↓4	↙←↓5	↙←↓6	↙←↓7	↙ 6	←7	↙ ←8
#	**0** #	1 e	2 x	3 e	4 c	5 u	6 t	7 i	8 o	9 n

If substitution operation means 2 is added with minimum value otherwise zero is added if two characters are equal. These cell value computation continued until D(n,m) computation is finished. To derive edit distance minimum try to backtrack the matrix. After the computation of D(n,m), it is traced back from upper right corner to lower leftmost corner ie from D(n,m) to D(0,0). The value available in D(n,m) gives the similarity measure of the given two names. As per the given output matrix the similarity value of the given two input names is 8. While performing the backtrack function the type of operations is performed can be identified using three different arrows shown in output matrix.

3.7 N-gram string matching algorithm

N–gram means all substrings are of length N. In N-gram model it is necessary to find the appropriate value for N. The N value should be stable one. For larger strings the value of N is three which is more appropriate than any other value. It is also called trigrams. All N-gram set for a particular name is called as sample space. The first step of this algorithm is to find all possible substrings of length N of the given two input class names. The order of the character in the substrings is corresponding to main string. It is used to find the match between given two names. For string similarity calculation all substrings are derived from both names. The numbers of substring similar are counted. The edit distance algorithm is used to find the similar substring of given two main string. The similarity between two names is calculated using the equation (12). Probability is a way of assigning every event a value between zero and one. The probability that one of the events will occur is given by the sum of

the probability of the individual events. The N-gram algorithm is working based on probability model.
According to the information theory, the commonality and description between two names are calculated. I(Common (name1,name2)) is equal to number of similar N-grams between two names. I(Description (name1,name2)) is equal to total number of N-grams available in both names. Sim(name1,name2) is calculated using equation (13). The optimized value of N is three which is trigrams. For the similarity computation of trigrams the names are divided into three letter words. Two arrays of trigrams are compared. Numbers of equal trigrams are counted. Using Dice's coefficients the similarity between two names is calculated as in equation (14). Computed similarity value is more if the similarity between the names is higher. Consider the example name1 = Alexander and name2 = Aleksander. The corresponding trigrams are Ale, lex, exa, xan, and, nde, der and Ale, lek, eks, ksa, san, and, nde ,der.



The similarity between the given two names $Sim_{N\text{-gram}}$ is equal to 1/(1+15-(2*4) = 0.125. Only four trigrams are common between the given two input names. Both names are described using fifteen trigrams. Using Dice co-efficient formula the similarity is calculated. Sim_{Dice} is equal to (2*4)/(7+8) = 0.533.

3.8 Soundex Algorithm

In this algorithm, converting name to code is an important task. Where it considers the first character of the name as it is and the remaining character encoded to a three digit number. For this encoding the prerequisite is to fix some numerical values for each and every character. Both capital and small characters are having the same numeric value. Similar sound characters are sharing common value. The numeric values of all the letters given in table 3.

Table 3 Characters equivalent numeric value

Characters	Value
b,f,p,v	1
c,g,j,k,q,s,x,z	2
d,t	3
l	4
m,n	5
r	6

The Soundex algorithm consists of four steps to generating code for the given name. The algorithm has two operation namely encoding and matching operations. In encoding, the name is converted into corresponding code. Matching operation identify exact match between the code derived from names. The codes are in equal length so Hamming distance algorithm is used to identify the similarity between the codes. Consider two set of inputs. The set1 consists of Robert, Rubert and its corresponding code is R163 for both names. The set 2 consists of week, weak and its code is W200.

Soundex code generation procedure

1. Retain the first letter of the string.
2. Remove all occurrences of the following letters, unless it is the first letter a,e,h,i,o,u,w,y.
3. Assign numbers to the remaining letters specified in the table 3.
4. If two or more letters with the same number were adjacent in the original name or adjacent except for any intervening h and w then omit all but the first.

Using Hamming distance similarity measure it is claimed that these two are similar names. This algorithm neglect the spelling difference produced by the vowels. In set2 similarity between names is one. But the meaning of both words is not related to any particular concept. This result is conveying wrong similarity value. For using this type of measures, it is been tried to understand weather it is suitable for chosen semantic web environment are not.

$$N-gram(name1,name2)=\frac{1}{1+|N-gram(name1)|+|N-gram(name2)| + 2*|N-gram(name1)\cap N-gram(name2)|} \qquad (12)$$

$$Sim(name1,name2)=\frac{I(common(name1,name2)}{I(description(name1,name2)} \qquad (13)$$

$$Sim_{Dice} = \frac{2*|trigram(name1)\cap trigram(name2)|}{|trigram(name1)|+|trigram(name2)|} \qquad (14)$$

4 Result and Discussions

Experiment was conducted on two set of class names in domain specific person ontologies. All seven algorithms were implemented and compared the properties like class name, object name, attribute name and type of relationship between class and subclass available in two input ontologies. The precision, recall and F-measure are computed for all algorithms. It was found that the similarities between the class names are not equal in all algorithms. Also it was identified that the results produced by some algorithms depend on number of operations were performed while transforming from one name to another. For some algorithms output was fraction, lies between 0 and 1. For comparison purpose all algorithm outputs were normalized between 0 and 1.

In this experiment the initial comparison was done between class names available in person ontologies belong to family domain. The ontologies were taken for comparison is shown in figure 1 and 2. Matched class names were identified by performing many to many comparison then object property and data property of all matched classes were considered for computing similarity between them. Protégé tool was used to view the person ontologies. The class properties, object properties and data properties are indentified and extracted from ontologies. The class names in first person ontology were compared to the class names available in second person ontology. Fifty two names were taken for comparison. The precision, recall and F-measure were calculated using the equation (15), (16) and (17).

$$Precision = \frac{Number\ of\ Accurate\ results}{Total\ Number\ of\ answers\ retrieving\ by\ the\ system} \quad (15)$$

$$Recall = \frac{Number\ of\ Accurate\ results}{Total\ Number\ of\ Accurate\ results\ from\ raw\ data} \quad (16)$$

$$F-Measure = \frac{2*Precision*Recall}{Presition+Recall} \quad (17)$$

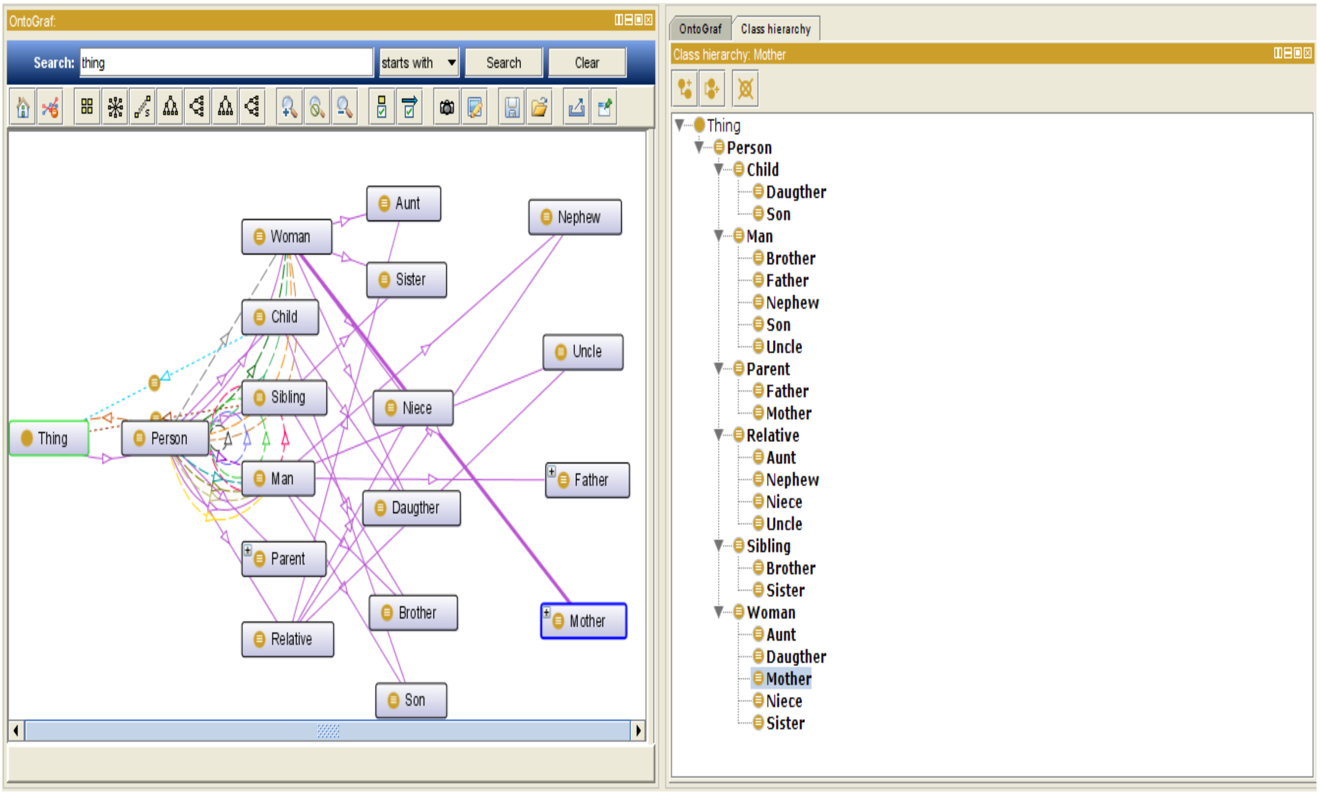


Fig.1 person.owl

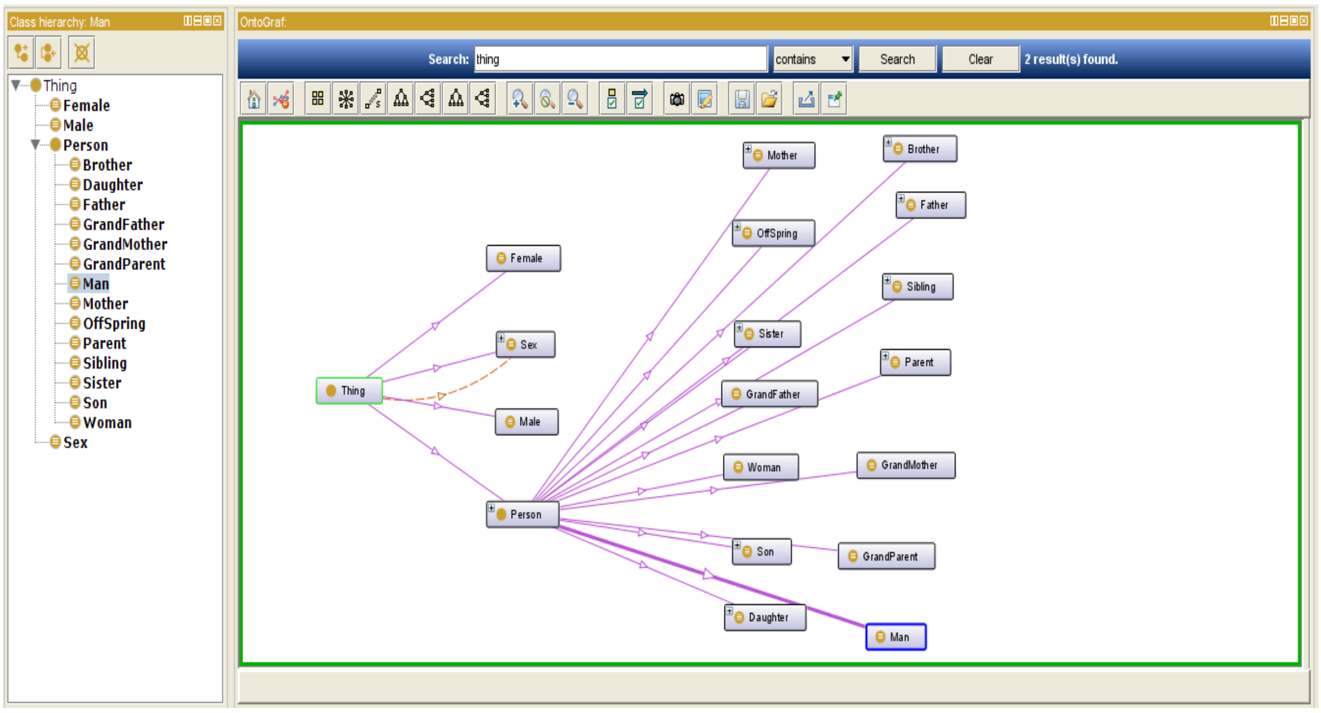


Fig. 2 person1.owl

Table 4 Performance comparison of syntax similarity algorithms

TH	Hamming Distance			Levenshtein distance			Dameran – Levenshtein Distance			JaroWinkler distance			N-gram			N-gram Dice's coefficient		
	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
0.7	1	0.12	0.21	1	0.53	0.69	0.8	0.71	0.75	0.81	0.88	0.84	1	0.12	0.21	1	0.38	0.55
0.8	1	0.12	0.21	1	0.32	0.48	0.7	0.38	0.5	0.93	0.76	0.84	1	0.12	0.21	1	0.35	0.52
0.9	1	0.12	0.21	1	0.26	0.41	1	0.26	0.41	1	0.41	0.58	1	0.12	0.21	1	0.18	0.31

Table 5 Performance of Soundex algorithm

TH	Soundex Algorithm								
	Hamming Distance			Dameran – Levenshtein Distance			JaroWinkler distance		
	P	R	F	P	R	F	P	R	F
0.7	0.9	0.76	0.84	0.8	0.91	0.86	0.8	0.94	0.86
0.8	0.9	0.47	0.63	0.9	0.76	0.84	0.9	0.76	0.82
0.9	0.9	0.47	0.63	0.9	0.47	0.62	0.94	0.5	0.65

P-Precision R-Recall F- F-measure TH-Threshold value

Threshold values were set to 0.7, 0.8, and 0.9. By varying the threshold values, the performance of the various algorithms were analyzed. The similarity values were calculated between each and every pair of words. The obtained values were whole numbers which indicate numbers of operations performed while transforming from one class name to another class name. Fixing suitable threshold value for all algorithms results in removal of unmatched pairs from the input class names.

The precision, recall and F-measures obtained using different algorithms is given in the table 4. Dameran–Levenshtein distance and Jaro Winkler distance algorithms are having high recall rate and it is more suitable for ontology classes. The remaining four algorithms are having low recall rate. Class names vary based on pronunciations can be easily identified using N-gram and Soundex algorithms. Class names vary with sound then soundex algorithm in combination with Dameran–Levenshtein distance algorithm and Jaro Winkler Distance algorithm shows good result. Results are shown in table 5.

5 Conclusion and Recommendations

The retrieval of information from a particular domain is done by merging domain specific ontologies. The first step of merging process involves the identification of similarity between class names of ontologies. From this experiment, some algorithms like hamming, Levenshtein and N-gram hold high precision but low recall rate. These algorithms are identified the class names which are exactly matched. But in real environment, some class names have different spelling. Due to the pronunciation of the class names, the spelling gets variation. These class names are not identified by these algorithms. The algorithms like Dameran – Levenshtein Distance and Jaro Winkler produced acceptable precision and recall rate. At 0.7 threshold value 80% of the class names were identified. While merging the ontologies from different region with different pronunciation the combination of algorithms like soundex with either Dameran – Levenshtein Distance or Jaro Winkler algorithms produces better result than any other algorithms. In semantic web environment, precision and recall are more important than execution time, because once if the merging was done without the human intervention, then the success ratio of automatic ontology merging will be very high. The researchers can try to understand the nature of class names in domain specific ontologies and then choose the appropriate algorithm for syntax matching in semantic web environment. The syntax matching is not enough for identification of similarity. Some contribution on identification of semantic matching is also necessary. The combination of both syntax and semantic will produce better results than any one techniques.

References:

[1] Alexandr Andoni and Krzysztof Onak, Approximating Edit distance in Near-Linear Time, *SIAM Journal of computing*, no. 6 2012, pp. 1635-1648.

[2] Alex X. Liu, Ke Shen,Eric Torng, Large Scale Hamming Distance Query Processing, *ICDE 27th International Conference,* IEEE, 2011, pp. 553-564.

[3] Antonis, Koukourikos, Giannis Stoitsis, Pythagoras Karampiperis, Data-Driven Schema Matching in Agricultural Learning Object Repositories, *Metadata and Semantics Research Communications in Computer and Information Science*, Volume 343, 2012, pp. 301-312.

[4] Babak Bagheri, Hariri, Hassan Sayyadi, Hassan Abolhassan, Combining Ontology Alignment Metrics Using the Data Mining Techniques, *ECAI International Workshop on Context and Ontologies*, 2006, pp.65-67.

[5] Blerim Rexha, Valon Raca, Agni Dika, Enhancement of string matching queries on Albanian Names for Kosovo Civil Registry, *Recent Advances in Computers, Communications, Applied Social Science and Mathematics*, ISBN: 978-1-61804-030-5, 2011.

[6] Daniel Rinser, Dustin Lange, Felix Naumann, Cross-lingual entity matching and infobox alignment in Wikipedia**,** *Information Systems*, Volume 38, Issue 6, 2013, pp. 887–907.

[7] Danny, Hermelin, Gad M. Landau, Shir Landau, Oren Weimann, A Unified Algorithm For Accelerating Edit-Distance Computation via Text-Compression, *Symposium on Theoretical Aspects of Computer Science,* 2009 (Freiburg), pp. 529–540.

[8] Dekang, Lin, An Information-Theoretic Definition of Similarity, *ICML*, Volume 98, 1998.

[9] Fabiana Freire, de Araujo, Fernanda Ligia R.Lopes, Bernadette Farias Loscio, MeMO: A Clustering-based Approach for Merging Multiple Ontologies, *Workshops on Database and Expert Systems Applications(DEXA)*, IEEE, 2010, pp. 176-180.

[10]Feiyu Lin, State of the Art: Automatic Ontology Matching, *Research Report*, ISSN 1404-0018, 2007.

[11]Feiyu Lin, Kurt Sandkuhl, Shoukun Xu, Context-based Ontology Matching: Concept

and Application Cases, *10th International Conference on Computer and Information Technology(CIT),* IEEE, 2010, pp. 1292-1298.

[12]Fuqi Song, Gregory Zacharewicz, David Chen, An ontology-driven framework towards building enterprise semantic information layer, *Advanced Engineering Informatics*, Science Direct, Volume 27, Issue 1, January 2013, pp. 38–50.

[13]Giorgos, Stoilos, Giorgos Stamou, Stefanos Kollias, Y. Gil et al., A String Metric for Ontology Alignment, *The Semantic Web-ISWC 2005,* Springer-Verlag Berlin Heidelberg, LNCS 3729, 2005, pp. 624–637.

[14]Hettiarachchi, Gayan Prasad, Attygalle, Dilhari, SPARCL: An Improved Approach for Matching Sinhalese Words and Names in Record Clustering and Linkage, *Global Humanitarian Technology Conference (GHTC)*, IEEE, 2012, pp. 423–428.

[15]Hong.N, Fang.A, Wu.S, Zheng.J, Qian.Q, An Integrated Biomedical Ontology Mapping Strategy Based on Multiple Mapping Methods, *Web Information Systems Engineering – WISE 2013 Workshops Lecture Notes in Computer Science*, Springer Berlin Heidelberg, Volume 8182, 2014, pp. 373-386.

[16]Kamel Hussein, Shafa anri, Jalal Omer Atoum, A Framework for Improving the Performance of Ontology Matching Techniques in Semantic Web, *International Journal of Advanced Computer Science and Applications*, Volume 3, No. 1, 2012, pp. 8-14.

[17]Laurent Boyer, Yann Esposito et al. SEDIL: Software for Edit Distance Learning, *Proceedings of the 19th European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer Berlin Heidelberg, 2008, pp. 672-677.

[18]Mahesh, Lalwani, Nitesh Bagmar, Saurin Parikh, Efficient Algorithm for Auto Correction Using n-gram Indexing, *International Journal of Computer & Communication Technology,* 2009, pp. 23-27.

[19]Maurice, Hermans, Frederik C. Schadd, A Generalization of the Winkler Extension and its Application for Ontology Mapping, *Proceedings of The 24th Benelux Conference on Artificial Intelligence(BNAIC)*, 2012.

[20]Mohammed Maree, Mohammed Belkhatir, A Coupled Statistical Semantic Framework for Merging Heterogeneous Domain Specific Ontologies, *International Conference on Tools with Artificial Intelligence(ICTAI),* IEEE Computer Society, Volume 2, 2010, pp. 159-166.

[21]Nirmala Pudota, Antonina Dattolo, Andrea Baruzzo, Felice Ferrara, Carlo Tasso, Automatic keyphrase extraction and ontology mining for content-based tag recommendation, *International Journal of Intelligent Systems, Special Issue: New Trends for Ontology-Based Knowledge Discovery*, Volume 25, Issue 12, 2010, pp. 1158–1186.

[22]Nowak.G, Grabowski.S, Draus.C, Zarebski.D, Bieniecki.W, Designing a computer-assisted translation system for multi-lingual catalogue and advertising brochure translations, *Proceedings of VIth International Conference on Perspective Technologies and Methods in MEMS Design (MEMSTECH),* IEEE, 2010, pp. 75–180.

[23]Richard C., Angell, Georege E, Freund, Peter Will, Automatic Spelling Correction Using a Trigram Similarity Measure, *Journal of Information Processing & Management*, Volume 19, Issue 4, 1983, pp. 255-261.

[24]Rishin, Haldar, Debajyoti Mukhopadhyay, Levenshtein Distance Technique in Dictionary Lookup Methods: An Improved Approach, *arXiv preprint arXiv: 1101.1232,* 2011.

[25]Samer Mahmoud, Wohoush, Mahmoud Hasan Saheb, Indexing for Large DNA Database Sequences, *International Journal of Biometrics and Bioinformatics (IJBB),* Volume 5, Issue 4, 2011, pp. 202.

[26]Sandeep, Chaware, Srikantha Rao, Analysis of Phonetic Matching Approaches for Indic Languages, *International Journal of Advanced Research in Computer and Communication Engineering,* Volume 1, Issue 2, April 2012.

[27]Siham Amrouch, Siham Mostefai, Syntactico-semantic algorithm for automatic ontology merging, *International Conference on Information Technology and e-Services(ICITeS)*, IEEE, 2012, pp. 1-5.

[28]Szymon Grabowski, Kimmo Fredriksson, Bit-parallel string matching under Hamming distance in O(n) worst case time, *Journal of*

Information Processing Letters, volume 105, Issue 5, 2008, pp. 182-187.

[29]Thierry Lecroq, Fast Exact String Matching Algorithms, *Journal of Information Processing Letters*, Volume 102, Issue 6, 2007, pp. 229-235.

[30]Vadivu, G., S. Waheeta Hopper, Ontology Mapping of Indian Medicinal Plants with Standardized Medical Terms, *Journal of Computer Science* 8, no.9, 2012, pp. 1576-1584.

[31]Vassilis Spiliopoulos, George A. Vouros, Vangelis Karkaletsis, On the discovery of subsumption relations for the alignment of ontologies, *Web Semantics: Science, Services and Agents on the World Wide Web*, Volume 8, Issue 1, March 2010, pp. 69–88.

[32]Wad Hassan Gomaa, AlyAly Fanmy, Arabic Short Answer Scoring with Effective Feedback for Students, *International Journal of Computer Applications*, Volume 8, Jan 2014, pp. 35-41.

[33]Yoshimasa T Saruoka, John Mc Naught, Ananiadou.S, Learning string similarity measures for gene/protein name dictionary look-up using logistic regression, *Journal of Bioinformatics*, Volume 23, Issue 20, 2007, pp. 2768-2774.