TOWARDS A SEMANTIC CATALOG OF SIMILARITY MEASURES

Alfredo Sánchez-Alberca, Rafael Lahoz-Beltra and Juan Castellanos-Peñuela

Abstract: *The tasks that encloses the Semantic Web often requires semantic comparisons between classes, properties, relations and instances, that mainly consist in semantic similarity. As a consequence, the number of similarity functions used in this field has greatly increased in the last years. However, the selection of the best similarity functions to use in each case is a difficult task usually guided by heuristics and intuition. To overcome this problem, in this paper we propose a taxonomy of similarity functions as a first step to a semantic catalog of similarity functions. This taxonomy is induced by another taxonomy of data types. Our taxonomy allows selecting the similarity functions applicable to a pair of entities based on the data type used to represent them. It also eases the composition of functions according the composition of data types, and supports the inheritance of applicable functions from the more abstract to the more specific data types. As a consequence, it facilitates the application of similarity functions in an automated way. Finally, we also populate this taxonomy with some of the most popular similarity functions used in Semantic Web.*

Keywords: *Semantic Web, semantic comparison, similarity measures*

ACM Classification Keywords*: 1.5.4 Pattern recognition applications. I.2.4 Knowledge Representation Formalisms and Methods.*

Introduction

Similarities play a key role in the process of semantic comparison of entities, i.e. classes, properties, relations and instances, in the Semantic Web. Each entity is described by a set of attributes of different types. In the case of ontologies any concept or class has an identifier (IRI), a name (string), a description in natural language (string) and, most likely, attributes whose range is a basic data type e.g. Boolean, integer, float, date or string, a composed data type e.g. set, list, etc., or other classes. However, entities are also described by their relationships with other entities. For example, a concept can be part of other concept, i.e. meronymy/holonymy relationship , or a specialization, i.e. hypernymy/hyponymy relationship, or other “ad hoc” relation. In addition, in populated ontologies or databases, concepts have also extensional definitions through their instances. So, to compare two entities we need to decide which properties should be taken into account as well as choose the best similarity functions to use for comparing them.

To illustrate the problem with an example, we are developping an ontology of spanish wines, as Spain is one of the majors producers of wine. Our intend is to compare wines, but if we want to compare two wines, first we need to establish which are the main wine characteristics to compare (grape variety, alcohol content, ph, colour, bouquet, flavor, price, etc.), and then how to compare them. For instance, if we are interested in tasting what similarity function should we use? Which functions could we really use? Furthermore, once we have measured the similarity between these characteristics, how will this result be used to calculate the similarity between the wines?

Reviewing the literature on similarity functions used for comparing entities in ontologies [Euzenat and Shvaiko, 2007; Cohen et al., 2003; Koudas et al., 2006; Cross and Sudkamp, 2002], we will note that there are a lot of similarity functions and their formalization sometimes is unclear for non-mathematicians. By and large, there are no rules that point out the similarity functions to use in each case and consequently the choice of an appropriate similarity function is not an easy task, being guided many times by heuristics or intuition. Even, in many occasions the choice is strongly conditioned by the understanding of the similarity function more than by its adequacy.

In order to overcome these problems, in this work we propose a new taxonomy of similarity functions that could help to guide the semantic comparison of entities. This classification is built according to the data type of the ontology or schema entities compared. The taxonomy defines a hierarchy of similarity functions based on the `is_a` relation for data types. In this way, similarity functions for a general data type could be applied to all its more specific descendant data types. It also eases the composition of functions according the composition of data types. This could be very helpful in order to automate the calculation and aggregation of multiple similarity measures from different similarity functions, especially when it is not clear which similarity function performs better.

At the same time, we have populated the taxonomy with the most used similarity functions in the Semantic Web. Based on this catalog, we also have developed a Java library of similarity functions called SIMEON (Similarity Measures for Ontologies).

The structure of the paper is as follows: First we review the notion of similarity and distance; next, we introduce the structure of the taxonomy of similarity functions; then we present some examples of similarity functions for simple and compound data types respectively, showing their classification in the taxonomy and illustrating their use; and, finally we draw the conclusions and point out future work.

Similarity functions

The similarity notion has been studied deeply in psychology. In this field, several theories has been proposed for measuring similarities, for example, see [Tversky, 1977] and [Goldstone, 1999]. This section deals with the definitions of similarity and distance functions proposed in the abovementioned theories. These definitions have been widely adopted by the artificial intelligence community.

Definition 1 (Similarity function)**.** Given a set of objects or entities E, a *similarity function* is a function $\sigma : E \times E \longrightarrow \mathbb{R}^+$ that maps every pair of entities of E with a real number that expresses the grade of likeness between the entities and that satisfies the following properties:

$$\text{Non negativity: } \forall x, y \in E, \sigma(x,y) \geq 0, \quad (1)$$

$$\text{Maximality: } \forall x, y \in E, \sigma(x,x) \geq \sigma(x,y). \quad (2)$$

Some authors add to these properties *symmetry* but, as showed in [Tversky, 1977], it does not hold in some contexts.

Definition 2 (Symmetric similarity function)**.** A *similarity function* $\sigma : E \times E \longrightarrow \mathbb{R}^+$ is *symmetric* if satisfies

$$\text{Symmetry: } \forall x, y \in E, \sigma(x,y) = \sigma(y,x). \quad (3)$$

In the same way, it is possible to define a function to measure dissimilarity.

Definition 3 (Dissimilarity function)**.** Given a set of objects or entities E, a *dissimilarity function* is a function $\delta : E \times E \longrightarrow \mathbb{R}^+$ that maps every pair of entities of E with a real number that expresses the grade of unlikeness between the entities and that satisfies the following properties:

$$\text{Non negativity: } \forall x, y \in E, \delta(x,y) \geq 0, \quad (4)$$

$$\text{Minimality: } \forall x \in E, \delta(x,x) = 0. \quad (5)$$

There are more constraining notions of dissimilarity, such as distances or metrics.

Definition 4 (Distance)**.** Given a set of entities E, a *distance* is a dissimilarity function that satisfies

$$\text{Definiteness: } \forall x, y \in E, \delta(x,y) = 0 \text{ iff } x = y, \quad (6)$$

$$\text{Symmetry: } \forall x, y \in E, \delta(x,y) = \delta(y,x), \quad (7)$$

$$\text{Triangular inequality: } \forall x, y, z \in E, \delta(x,y) + \delta(y,z) \geq \delta(x,z). \quad (8)$$

In order to facilitate comparison, the functions of similarity and dissimilarity should be normalised.

Definition 5 (Normalised (dis)similarity function). A (dis)similarity function is *normalised* if their range is the real unit interval $[0, 1]$.

Very often, similarity functions are built from dissimilarity functions or distances.

Definition 6 (Correspondence between similarity and dissimilarity functions). A similarity function σ and a dissimilarity function δ, are correspondent if

$$\sigma(x, y) = \pi(\delta(x, y)),$$

for some isomorphism $\pi : R \longrightarrow [0, 1]$ that inverts the order (decreasing monotony) and such that $\pi(0) = 1$.

Some popular isomorphisms π are

a) $\pi(x) = 1 - x$, when δ is normalised.

b) $\pi(x) = 1 - \frac{x}{\max \delta}$, when δ has a maximal value.

c) $\pi(x) = 1 - \frac{x}{1 + x} = \frac{1}{1 + x}$, when δ is not bounded.

Taxonomy of similarity functions

We have found in literature many similarity functions that have been used in the Semantic Web, particularly for ontology or schema matching. In [Ehrig et al., 2005] there is a classification of similarity functions for ontology matching, based on the source of information taken for assessing the similarity measure, either from the ontologies being matched (with their attributes, relations and instances), or from other related ontologies (through previous mappings). But we have no found any classification based on data types of the entities to compare with the similarity functions. Hence, we are interested in building a taxonomy of similarity functions according to this classification.

Data types define the semantic of data. According to ISO/IEC 11404 for general purpose data types [iso, 2007] a data type is, from the conceptual point of view, a set of different values characterized by a set of properties and operations. This definition includes both the intensional part of the type, which provides the properties that their values satisfy, and the extensional one, which gives the explicit set of values that conform the data type. The taxonomy established in this work is based in the intensional part of the specification, that is, in the properties of values of data types that allow or disallow the use of different similarity functions.

According to the data type complexity of their domain, we distinguish the following similarity functions:

1. **Similarity functions for simple data types**. These are similarity functions whose domain is a pair of basic data types, such as numeric types (e.g. integers, real numbers, etc.), characters, Booleans, etc. Basic similarity functions well known fall into this category, which is dealt with in the next section.

2. **Similarity functions for compound data types**. These are similarity functions whose domain is a pair of data types composed of elements of other data types (usually a collection of elements). The functions for these data types are the most interesting part of the taxonomy because most similarity functions used in ontology or schema matching fall in this category. The classification under this category is explained after the section corresponding to simple data types.

Figure 1 shows the taxonomy built according to this criterion. This taxonomy defines an `is_a` hierarchy of similarity functions based on the data types of their domain[1]. Hence, any similarity function defined for a data type could be calculated for any of its descendant data types in the hierarchy.

[1]We refer here to the mathematical sense of the word domain, that is, the set of elements where a function is defined.

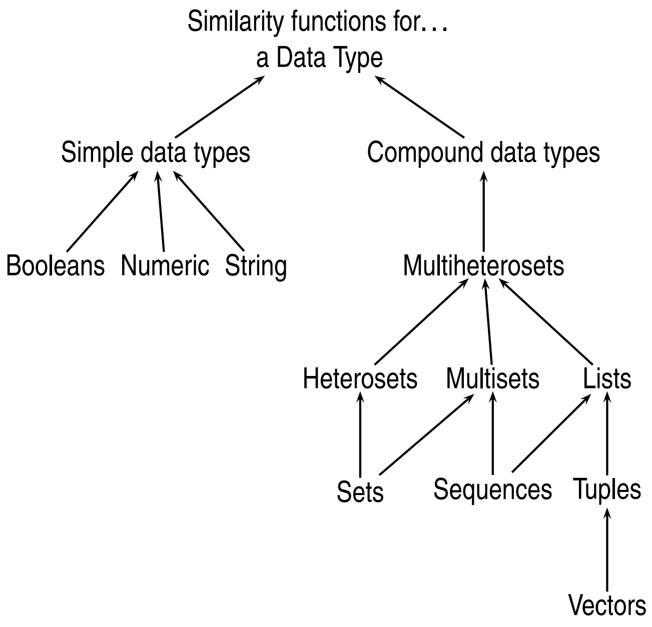


Figure 1: Similarity functions taxonomy based on domain data types. The hierarchy is organized according to the `is a` relation. We have omitted the "Similarity functions for" prefix in each node to narrow the graph.

At the root of the hierarchy we define the most trivial similarity function, the identity similarity, that is valid for all data types.

Definition 7 (Identity similarity). Given a universe E of elements of any data type, the *identity similarity function* is a similarity function $\sigma_{id} : E \times E \longrightarrow [0,1]$ defined as

$$\sigma_{id}(s,t) = \begin{cases} 1, & \text{iff } s = t; \\ 0, & \text{iff } s \neq t. \end{cases}$$

The next sections present the main categories of this taxonomy and guive some examples of common similarity functions that populate them.

Similarity functions for simple data types

Simple data types are data types whose elements cannot be decomposed in simpler ones. To identify similarity functions for this part of the taxonomy, we have to take into account that the comparison between elements of a data type depends on its semantics and its properties. For example, a number is represented by a symbol, but its associated semantics is a magnitude expressed by a quantity, hence it is possible to use similarity functions that compare quantities. The words of a language do not represent quantities, but they have a meaning in the language that permits their comparison by means of synonymy, hyponymy or meronymy relations.

In this section we review the most important similarity functions for Boolean, numeric and string data types.

Similarity functions for Booleans

A *boolean* is a data type that only has two possible values, usually represented as 0 and 1 (or `False` and `True` respectively).

This is the most simple data type, and the only similarity function for them is the *Boolean identity function*, which coincides with the XNOR operation for bits.

XNOR similarity is the basis to compare bits sequences in computer communication, for example. Moreover, this similarity could be used for comparing boolean attributes of ontology or schema entities.

Similarity functions for numeric data types

A *number* is a data type that represents a quantity. There are many numeric data types, but the most common are Natural $\mathbb{N}$, Integer $\mathbb{Z}$, Rational $\mathbb{Q}$ and Real $\mathbb{R}$. The fact that a number has associated a quantity permits defining an order relation between the numbers that could be used to compare them by means of their difference.

The most important distance for two numbers x and y is the *difference* that is defined as $\delta(x, y) = |x - y|$. This distance (and the corresponding similarity function) is not normalized, but usually it is normalized dividing it by the maximal distance in the set of numbers to compare.

Similarity functions for strings, words or terms

A *string* is a sequence of symbols or characters in an alphabet. In natural languages, strings forms words, terms, and texts. *Words* are the units of a language [Wilson, 2008], and a *term* is a sequence of few words in a language, which represents a concept. Words and terms could be seen as a sequence of elements of type character, and from that point of view, they could be managed as a compound data type, but here, words are considered indivisible in elements belonging to simpler data types.

Natural languages have the capacity of expressing the same or similar concepts with different words or terminological variations. According to [Maynard and Ananiadou, 1999] the main kinds of word variations are

Morphological: It is a variation in the form of a word based on the same linguistic root. There are three types: inflection, derivation and a combination of both.

Syntactic: It is a variation in the grammatical structure of a word. There are three types: insertion, permutation and coordination.

Morphosyntactic: It is a combination of both variations, morphological and syntactic.

Semantic: It is a variation in the whole form of a word, providing a synonym, hypernym or hyponym.

Techniques for detecting morphological variations are based on stemming algorithms that transform a word in its linguistic root, eliminating prefixes or suffixes of number, gender, and tense. One of the algorithms most used for the English language is Porter algorithm [Porter, 1980].

Techniques for detecting syntactic variations are based on rewriting rules, such as the rule that inverts the order of two consecutive words in a term. These rules are detected by analysing the repetition of patterns in a huge linguistic corpus.

On the other hand, semantic terminological variations could be assessed with the help of dictionaries or thesauri, e.g. WordNet [Miller, 1995]. Some similarities functions use the synsets (sets of synonyms) of thesauri and others use their taxonomies. The taxonomy of a thesaurus defines a conceptual space by means of hyponymy or meronymy relations ($\sqsubseteq$) and it is possible to exploit the semantic association between concepts through these relations in order to establish their similarity. The most famous similarity functions based on thesauri are the *information content similarity function* and the *Resnik similarity function* [Resnik, 1995].

The other option to measure the semantic similarity between concepts in a taxonomy is by means of the distance between the corresponding nodes. The logic behind is that two concepts are more similar as shorter is the path between them. The easiest way to calculate this distance is to count the number of arcs, that is, the length of the shortest path between nodes. This idea was proposed in [Rada et al., 1989] and later applied to WordNet in

[Lee et al., 1993] and [Leacock and Chodorow, 1998]. The most important similarity functions that apply this are the shortest path similarity and the ancestor similarity [Maedche and Zacharias, 2002].

Finally, there is also the possibility of combining both alternatives, information content based and path distance based, as shown in [Hahn and Chater, 1997].

Similarity functions for compound data types

Compound data types are constructed from elements of other data types following some structure. For comparing compound data types it is necessary to consider not only the similarity between the components data types, but the information included in the structure itself. It is important to emphasize the properties of every type of structure in order to define similarity functions.

Considering the main structural properties, we present a novel classification of aggregated data types according to four fundamental properties: homogeneity, size, order, and element uniqueness.

Homogeneity refers to the condition that all elements are, or are not, of the same base data type.

Size refers to the fact that the number of elements that conforms the collection is fixed or variable. If it is fixed, the comparison is always carried out between entities with the same size. If it is variable, a comparison between entities with different sizes is possible.

Order refers to fact that elements in the collection must follow an established order. This order has nothing to do with the underlying or natural order of data type elements, but with the structure of the collection itself.

Uniqueness refers to the condition that element repetitions are, or are not, allowed.

If we make a grid over the above properties we find the data types shown in Table 1. In that table some cells of the grid are empty, but this fact does not mean that a data type of such characteristics does not exist. For example, the upper left corner cell corresponds to what in combinatorial are *variations*, which are ordered collections of data with fixed size of the same type that do not repeat, or if we relax the order restriction we have *combinations*. However, these data types are rarely used in schemas or ontologies and will not be dealt with in this article.

Table 1: Classification of the collection data types according to properties of homogeneity, size, order and element uniqueness.

		Fixed size		Variable size	
		With uniqueness	Without uniqueness	With uniqueness	Without uniqueness
Homogeneous	With order		*Vector*		*Sequence*
	Without order			*Set*	*Multiset*
Heterogeneous	With order		*Tuple*		*List*
	Without order			*Heteroset*	*Multiheteroset*

The above four properties establish, in fact, constrains that give structure to data types. For example, a data type with order is more structured than a data type without order, and a data type with fixed size is more structured than a data type with no restriction about size. Thereby, the more structured a data type is, the more information we have to calculate semantic similarities.

In addition, these properties induce an `is a` hierarchy over the data types of table 1 and over the similarity functions applied to the entities of these data types, as shown in Figure 1. In this hierarchy the similarity functions corresponding to the most structured data types are in the lowest levels. This hierarchy facilitates the categorization and the application of similarity functions for these data types because similarity functions for a data type could be applied to all its descendant data types. For example, similarity functions for a data type without uniqueness restriction as multisets could also be applied to data types with uniqueness restriction as sets, because a set `is a` particular case of multiset.

Hence, in the rest of this section, we proceed to present the categories of compound similarity functions, giving some examples common similarity functions that populate them.

Similarity functions for multiheterosets

A *multiheteroset* is a variable-sized unordered collection of elements of different types where repetitions of elements are allowed. Multiheterosets are the less structured data types because they have no restriction. As a consequence, there are few ways of comparing them, especially when data types of multiheteroset elements are incomparable. The most intuitive function results from the comparison between the intersection and the union of the multiheterosets, but taking into account element repetitions, and it is known as *Jaccard similarity function* [Jaccard, 1901].

It is possible to assign different weights to commonalities and discrepancies, as for example does *Dice's similarity function*, that assigns double weight to coincidences [Dice, 1945].

Similarity functions for heterosets

A *heteroset* is a variable-sized unordered collection of elements of different types where repetitions of elements are not allowed.

Heterosets are a particular case of multiheterosets; therefore, every similarity function of the previous section could be used for heterosets.

In addition to the previous similarity functions, here is possible to follow a more general approach that comes from Tversky's contrast model [Tversky, 1977], where similarities between two objects A and B depends on the features common to A and B, features in A but not in B and features in B but not in A. Some well-known similarity functions derived from this model are the *Manhattan*, *Simpson* and *Euclidean similarity functions* [Holliday et al., 2002].

Similarity functions for multisets

A *multiset* is a variable-sized unordered collection of elements of the same type where repetitions of elements are allowed.

As multisets are a particular case of multiheterosets, all similarity functions for multiheterosets could be used with multisets.

From a statistical point of view, multisets are samples with an associated frequency distribution. Hence, it is possible to compare distributions to asses a similarity measure. A well-known test for comparing distributions is *Chi-square test* [Manning and Schütze, 1999].

Another different approach is to assign a weight to each element, which depends not only on its frequency in the multiset, but also on its frequency in other multisets in the same application domain. A well-known function that uses this idea is TF$\times$IDF (*Term Frequency* $\times$ *Inverse Document Frequency*); this function is usually applied in the vector space model for comparing documents (viewed as multisets of words) in information retrieval [Salton et al., 1975].

Similarity functions for sets

A *set* is a variable-sized unordered collection of different elements of the same type where repetitions of elements are not allowed.

Sets are particular cases of heterosets and multisets, therefore all similarity functions for these data types could be used for sets. These similarity functions only take into account whether sets have or not common elements; however, when elements are different they do not take into account their differences, because many times are incomparables. However, as elements of sets are of the same type, they can be easily compared. Therefore, if there exists a similarity function or distance for the elements of the set, it is possible to calculate the similarity between two sets as a function of the similarity between their elements. There are several alternatives: the most optimistic is to take the two elements more similar, and the most pessimistic is to consider the two elements less similar. The problem of these measures is that only one pair of elements is considered. Most of the time it is much more reasonable to consider similarities between all possible pairs (average similarity) However, this last function is not really a similarity function because it does not satisfy the maximality condition (2). According to [Valtchev and Euzenat, 1997], to satisfy maximality condition it is necessary to match elements and to measure the similarities between the elements matched. This optimal matching must satisfy the following:

a) Every element of A must match at most one element of B and vice versa.
b) The number of elements matched must be maximal.
c) The matching must make the similarity maximum.

The search of such optimal matching is an optimization problem that has been addressed in [Ahuja et al., 1993].

Similarity functions for lists

A *list* is a variable-sized ordered collection of elements of different types where repetitions of elements are allowed.

The only constrain for the lists is order. If we relax this constrain, the lists could be seen as multiheterosets; then we can use similarity functions for multiheterosets to compare lists.

Exploiting the order restriction provides different strategies for comparing lists. The easiest strategy is to count matching elements at the same list position as does the *Hamming's similarity function*.

It is also possible to consider matching elements in a fixed-size window, as does the *Jaro's similarity function*, that also takes into account transposition of matching elements in the window [Jaro, 1989]. Another variation of this similarity is *Jaro-Winkler's similarity function*, that give higher similarity values to sequences with longer common prefixes [Winkler, 1999].

Although Jaro's and Jaro-Winkler's similarity functions are both valid for lists, they came up in the field of record linkage and are mainly used with strings, which are a particular case of sequences.

Another natural way of comparing lists is through their sub-lists, especially when order is so important as to not permit transpositions. The comparison is done by measuring the length of the biggest common sub-lists. This similarity function is useful to identify identical whole parts in both lists, something very important when these parts define some properties of the lists. Variations of this similarity only take into account prefix or suffix sub-lists.

Similarity functions for sequences

A *sequence* is a variable-sized ordered collection of elements of the same type where repetitions are allowed.

As we have seen, words or terms are a special class of symbol sequences in an alphabet. Therefore, within the context of schema and ontology comparison, the measures presented in this section are used often with labels of entities.

Sequences are a particular case of lists where elements belong to a unique domain; hence, every similarity function for lists could be used for sequences.

On the other hand, it is possible to consider sequences as multisets that relax the order condition and thus to apply the similarity functions for multisets to compare sequences.

To compare sequences, like in the case of lists, the most trivial strategy is to compare elements in the same position. If we only count matches we have Hamming's similarity function; however, this function only takes into account coincidences between elements in the same positions. Therefore, when elements do not match, the function does not measure discrepancies between elements. Now that elements are comparable because they are of the same type, it is possible to extend Hamming's similarity when dealing with similarities between elements in the same position.

A different strategy is to compare all the proper sub-sequences of a fixed length of a sequence. These subsequences are known as *n-grams* [Kondrak, 2005]. It is possible to compare the multisets of n-grams of two sequences using any of the similarity functions for multisets as, for example, Jaccard's similarity function. When we compare terms, this function gives good results for abbreviations.

Another typical strategy is to measure the number of elemental operations that transforms a sequence into another [Hahn and Chater, 1997]. A well-known distance that makes use of this strategy is the edit distance of Levenshtein [Levenshtein, 1966], which only considers three possible operations: insertion, deletion and substitution of an element.

Similarity functions for tuples

A *tuple* is a fixed-size ordered collection of elements of different types where repetitions of elements are allowed.

Tuples are often used in schemas and ontologies to record objects with attributes of different types.

Tuples could be seen as lists that relax the fixed-size restriction; in this case it is possible to use the similarity functions for lists to compare tuples.

Given that tuples have fixed size and are ordered, elements in the same position are in correspondence; hence, it is possible to compare elements in the same position and to combine the similarities obtained with an aggregation function. The most well-known family of similarity functions for tuples is based on averages. For example, if we consider the identity similarity function to compare elements that are in the same position in the tuples, the resulting average similarity function is Hamming's similarity function. When elements of tuples have different importance it is better to use a *weighted average*.

Similarity functions for vectors

A *vector* is a fixed-size ordered collection of elements of the same type where repetitions of elements are allowed.

Vectors are a particular case of tuples and, consequently, it is possible to apply the similarity functions for tuples to compare vectors.

In vectors, as in tuples, there is a correspondence between elements in the same position; besides similarity functions for vectors often combine similarities between elements though using the same similarity function for each position in the vector. There are different ways of combining elements similarities, and one of the most well-known is the family of *Minkowski's distances*. This family offers many well-known distances as, for instance, the *Manhattan distance* or the *Euclidean distance*.

Conclusions and future work

Up to date many similarity functions have been published, but they are not presented and organized in a way that the selection of one or several of them to do semantic comparisons is an easy task. To solve this problem, we have presented a taxonomy of similarity functions. The structure of the taxonomy is induced by a taxonomy of data types, in a way that functions applicable to compare entities of a particular data type are applicable to every data type that is a subclass of this. The compound data types are classified according to their homogeneity, size variation, order and uniqueness. It should be added that the composition of data types allows the composition of similarity functions. That is, if two entities e_1 and e_2 are, both of them, represented by a data type t composed by the data types $t_1, t2, \ldots, t_n$, then the function used to measure the similarity between e_1 and e_2 can be obtained by composing similarity functions that correspond to $t_1, t_2, \ldots, t_n$. For example, if the input can be represented by a vector of numbers, the similarities between numbers can be combined through the similarity functions for vectors. The function that can be used in the composition depends on where t is inside the taxonomy.

With this taxonomy, the steps to select a similarity function are as follows:

1. To identify the data type that can be used to represent the entities to be compared.
2. To obtain the applicable functions according to the taxonomy.
3. To select the functions to be used according to the criteria of the experts in the issue, the previous use of the functions, etc. At this step, if it is not clear which similarity function to use, it is possible to apply all the similarity functions that populate that category and to decide after analysing the results.

Inside every category of the taxonomy, we have shown some of the most-used similarity functions though it must be said that many others have not been included here for size restrictions.

On the basis of this taxonomy we have built a Java library called SIMEON (Similarity Measures for Ontologies)[2] where the most popular similarity functions has been implemented.

In the near future we plan to populate the taxonomy with new similarity functions and to experiment applying them to an wine ontology, which we have recently build in other project, in order to discover which combinations of similarity functions perform better to compare wines.

Acknowledgements

This paper is published with partial support by the project ITHEA XXI of the ITHEA ISS (www.ithea.org) and the ADUIS (www.aduis.com.ua).

Bibliography

[iso, 2007] Iso/iec 11404:2007. information technology – general-purpose datatypes (gpd). http://www.iso.org/iso/catalogue_detail.htm?csnumber=39479.2007.

[Ahuja et al., 1993] Ahuja, R., Magnanti, T., and Orlin, J. *Network Flows: Theory, Algorithms and Applications*. Prentice Hall, New Jersey, 1993.

[Cohen et al., 2003] Cohen, W. W., Ravikumar, P., and Fienberg, S. E. A comparison of string distance metrics for name-matching tasks. In Knoblock, C. and Kambhampati, S., editors, *Proceedings of IJCAI-03 Workshop on Information Integration*, pages 73–78, Acapulco, Mexico, 2003.

[Cross and Sudkamp, 2002] Cross, V. and Sudkamp, T. A. *Similarity and compatibility in fuzzy set theory : assessment and applications*, volume 93 of *Studies in Fuzziness and Soft Computing*. Physica-Verlag, Heidelberg; New York, 2002.

[Dice, 1945] Dice, L. R. Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302, 1945.

[Ehrig et al., 2005] Ehrig, M., Haase, P., Stojanovic, N., and Hefke, M. Similarity for ontologies — a comprehensive framework. In Bartman, D., Rajola, F., Kallinikos, J., Avison, D., Winter, R., Ein-Dor, P., Becker, J., Bodendorf, F., and Weinhardt, C., editors, *Proceedings of the 13th European Conference on Information Systems (ECIS 2005)*, Regensburg, Germany, 2005.

[2] https://github.com/asalber/simeon

[Euzenat and Shvaiko, 2007] Euzenat, J. and Shvaiko, P. *Ontology Matching*. Springer, Berlin, Heidelberg, 2007.

[Goldstone, 1999] Goldstone, R. Similarity. In Wilson, R. and Keil, F., editors, *MIT encyclopedia of the cognitive sciences*, pages 763–765, Cambridge, Massachusetts. MIT Press, 1999.

[Hahn and Chater, 1997] Hahn, U. and Chater, N. Concepts and similarity. In Lamberts, L. and Shanks, D., editors, *Knowledge, Concepts and Categories*, Cambridge, Massachusetts. Physhology Press/MIT Press, 1997.

[Holliday et al., 2002] Holliday, J. D., Hu, C. Y., and Willett, P. Grouping of coefficients for the calculation of inter-molecular similarity and dissimilarity using 2d fragment bit-strings. *Combinatorial Chemistry and High-Throughput Screening*, 5(2):155–166, 2002.

[Jaccard, 1901] Jaccard, P. Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bulletin del la Société Vaudoise des Sciences Naturelles*, 37:547–579, 1901.

[Jaro, 1989] Jaro, M. A. Advances in record-linkage methodology as applied to matching the 1985 census of Tampa, Florida. *Journal of the American Statistical Association*, 84(406):414–420, 1989.

[Kondrak, 2005] Kondrak, G. N-gram similarity and distance. In Consens, M. P. and Navarro, G., editors, *12th International Conference String Processing and Information Retrieval (SPIRE)*, volume 3772 of *Lecture Notes in Computer Science, Berlin, Germany*, pages 115–126, Buenos Aires, Argentina. Springer, 2005.

[Koudas et al., 2006] Koudas, N., Sarawagi, S., and Srivastava, D. Record linkage: similarity measures and algorithms. In Chaudhuri, S., Hristidis, V., and Polyzotis, N., editors, *ACM SIGMOD International Conference on Management of Data*, pages 802–803, Chicago, Illinois. ACM, 2006.

[Leacock and Chodorow, 1998] Leacock, C. and Chodorow, M. Combining local context and wordnet similarity for word sense identification. In Fellfaum, C., editor, *MIT Press*, pages 265–283, Cambridge, Massachusetts, 1998.

[Lee et al., 1993] Lee, J. H., Kim, M. H., and Lee, Y. J. Information retrieval based on conceptual distance in is-a hierarchies. *Journal of Documentation*, 49(2):188–207, 1993.

[Levenshtein, 1966] Levenshtein, V. Binary codes capable of correcting deletions, insertions, and reversals. *Cybernetics and Control Theory*, 10(8):707–710. Original in *Doklady Akademii Nauk SSSR* 163(4): 845–848, 1965.

[Maedche and Zacharias, 2002] Maedche, A. and Zacharias, V. Clustering ontology-based metadata in the semantic web. In Elomaa, T., Mannila, H., and Toivonen, H. T. T., editors, *13th European Conference on Machine Learning (ECML'02) 6th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'02)*, Helsinki, Finland, 2002.

[Manning and Schütze, 1999] Manning, C. D. and Schütze, H. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, Massachusetts, 1999.

[Maynard and Ananiadou, 1999] Maynard, D. and Ananiadou, S. Term extraction using a similarity-based approach. In Bourigault, D., Jacquemin, C., and Lhomme, M.-C., editors, *Recent Advances in Computational Terminology*, pages 261–278. John Benjamins, Amsterdam (NL), 1999.

[Miller, 1995] Miller, A. Wordnet: A lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[Porter, 1980] Porter, M. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[Rada et al., 1989] Rada, R., Mili, H., Bicknell, E., and Blettner, M. Development and application of a metric on semantic nets. *IEEE Transactions on Systems, Man and Cybernetics*, 19(1):17–30, 1989.

[Resnik, 1995] Resnik, P. Using information content to evaluate semantic similarity in a taxonomy. In Perrault, C. R., editor, *Proceedings of the 14th IJCAI*, pages 448–453, Montréal (Canada), 1995.

[Salton et al., 1975] Salton, G., Wong, A., and Yang, C.-S. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[Tversky, 1977] Tversky, A. Features of similarity. *Psychological Review*, 84(4):327–352, 1977.

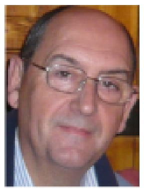
[Valtchev and Euzenat, 1997] Valtchev, P. and Euzenat, J. Dissimilarity measure for collections of objects and values. In Liu, X., Cohen, P. R., and Berthold, M. R., editors, *Advances in Intelligent Data Analysis, Reasoning about Data, Second International Symposium, IDA-97*, volume 1280 of *Lecture Notes in Computer Science, Berlin, Germany*, pages 259–272, London, United Kingdom. Springer, 1997.

[Wilson, 2008] Wilson, B. The natural language processing dictionary. `http://www.cse.unsw.edu.au/~billw/nlpdict.html`, 2008.

[Winkler, 1999] Winkler, W. E. The state of record linkage and current research problems. Technical Report Statistical Research Report Series RR99/04, U.S. Bureau of the Census, Washington, D.C., 1999.

Authors' Information

Alfredo Sánchez-Alberca *- San Pablo CEU University of Madrid, Urbanización Montepríncipe, s.n. 28668 Boadilla del Monte, Spain; e-mail: asalber@ceu.es*
Major Fields of Scientific Research: Artificial intelligente, Machine learing, Knowledge modelling and representation, Ontologies and Semantic Web



Juan Castellanos-Peñuela *- Head of Natural Computing Group, Faculty of Computer Sciences, Polytechnic University of Madrid, Campus de Montegancedo, s.n., 28668 Boadilla del Monte, Spain; email: jcastellanos@fi.upm.es*
Major Fields of Scientific Research: Natural computing, Formal language and Automata theory.

Rafael Lahoz-Beltra *- Faculty of Biological Sciences, Complutense University of Madrid, 28040 Madrid, Spain; e-mail: lahozraf@ucm.es*
Major Fields of Scientific Research: Evolutionary computation, Embryo development modelling and Design of bioinspired algorithms.