A Generalization of the Winkler Extension and its Application for Ontology Mapping

Maurice Hermans Frederik C. Schadd

Maastricht University, P.O. Box 616, 6200 MD Maastricht, The Netherlands

Abstract

Mapping ontologies is a crucial process when facilitating system interoperability and information exchange. Ontology Mapping systems commonly utilize string metrics in the mapping process to compare concept names. String metrics can be extended using the Winkler method, which increases the similarity value of two strings if these have a common prefix. A common occurrence for two corresponding ontology concepts is that the name of the first concept is a non-prefix sub-string of the name of the second concept. The Winkler extension does not allocate a higher similarity value to these pairs of strings, however intuitively this indicates that the two names have a similar meaning. This paper proposes a generalization of the Winkler extension, such that pairs of names with large common non-prefix sub-strings receive a higher similarity value as well. The proposed metric is evaluated on a record-matching dataset and a dataset from the Ontology Alignment Evaluation Initiative. The experiments reveal that metrics applying our proposed generalization outperform the same metrics when applying the Winkler extension.

1 Introduction

Ontologies commonly form the basis of modern knowledge systems. These ontologies are created by domain experts to suit the needs of the specific knowledge system. Hence, it is likely that two ontologies describing the same domain, but originating from different knowledge systems, will contain differences such as heterogeneous concept names, structure or granularity. Facilitating information exchange between knowledge systems which are based on heterogeneous ontologies is a challenging, but crucial task. In order to exchange information between two ontologies, a mapping is required which identifies the correspondences between the ontology concepts. The task of matching is a critical operation in many fields, such as semantic web, schema/ontology integration, data warehouses, e-commerce etc. While contemporary knowledge systems are commonly based on ontologies, the problem of mapping conceptualizations of knowledge domains originates from the field of databases, which utilize schemas to encode meta data.

The task of matching takes as input two ontologies, each consisting of a set of concepts and determines as output the relationship. There are multiple relationships possible e.g. equivalence, subsumption but in this article we only deal with equivalence. To match two concepts there are numerous characteristics to consider which, when all added together, will determine whether they match or not. One such characteristic is the name of a concept which is exploited by string-based approaches. The task of matching entity names has been explored by a number of communities, including statistics, databases, and artificial intelligence. A matching system uses several similarity measures which exploit different ontology characteristics in order to produce an alignment between ontologies. One of these characteristics are the names of the concepts in an ontology, which are exploited with syntactic similarities, more specifically string similarities, which are the focus of this paper. This paper will also make an extension to already existing techniques by taking into account the longest common substring when comparing two strings. All techniques discussed in this paper will be evaluated using the datasets by Cohen et al. [1] and the conference dataset originating from the 2010 Ontology Alignment Evaluation Initiative (OAEI) [4].

The rest of this paper is structured as follows. Section 2 will provide the reader with the necessary background information of this domain. Section 3 will detail the proposed extension of contemporary methods in this field. In section 4 the experiments performed with the results obtained will be presented. Section 5 will discuss the results obtained in chapter 4 and also propose future research. Section 6 will report the conclusions of the research performed.

2 Background information

2.1 Schemas and ontologies

The use of schemas originates from the field of databases, they are used to encode meta data, which is very useful to retrieve relevant data from a database. Later ontologies were developed which add more expressive ways to encode the meta data. Both methods are widely used in knowledge systems. There are some important differences and commonalities between schemas and ontologies as described by Shvaiko et al. [13], of which the keypoints are:

1. Database schemas often do not provide explicit semantics for their data. Semantics is usually specified explicitly at design-time, and frequently is not becoming a part of a database specification, therefore it is not available [11]. Ontologies are logical systems that themselves obey some formal semantics, e.g., we can interpret ontology definitions as a set of logical axioms.

2. Ontologies and schemas are similar in the sense that (i) they both provide a vocabulary of terms that describes a domain of interest and (ii) they both constrain the meaning of terms used in the vocabulary [6, 15].

3. Schemas and ontologies are found in such environments as the Semantic Web, and quite often in practice, it is the case that we need to match them.

Ontology mapping frameworks provide knowledge systems with the capacity to exchange information with other knowledge systems which use different ontologies. But before a framework can map ontologies, the system needs to ensure the interoperability of representations through transformations. There are several levels at which interoperability can be accounted for as described by Euzenat et al. [3].

1. Encoding: being able to segment the representation in characters.

2. Lexical: being able to segment the representation in words (or symbols).

3. Syntactic: being able to structure the representation in structured sentences (or formulas or assertions).

4. Semantic: being able to construct the propositional meaning of the representation.

5. Semiotic: being able to construct the pragmatic meaning of the representation (or its meaning in context).

2.2 Matching techniques categorization

Ontology mapping frameworks exploit multiple ontology characteristics during the matching process [13]. Matching techniques can compare two ontology concepts by utilizing information which describe the concepts themselves, or by investigating other related concepts, thus also exploiting the structure of an ontology. Techniques which utilize the structure of the ontology can be categorized as follows:

1. Graph-based techniques are graph algorithms which consider the input as labelled graphs. The considered ontologies are viewed as graph like structures containing terms and their inter-relationships. The comparison of a pair of nodes within the graph is usually based on their position within the graphs. The intuition behind is that, if two nodes are similar their adjacent nodes might also be similar.

2. Taxonomy-based techniques are also graph algorithms which consider only the specialization relation. The intuition behind this is that *is-a* links connect already similar terms, therefore their neighbouring nodes may also be somehow similar.

3. Repository of structures stores schemas/ontologies and their fragments together with pairwise similarities between them. When new structures are to be matched, they are first checked for similarity to the structures which are already available in the repository. The goal is to identify structures which are sufficiently similar to be worth matching in more detail, or reusing already existing alignments.

4. Model-based algorithms handle the input based on its semantic interpretation (e.g., model-theoretic semantics). Thus, they are well grounded deductive methods.

Matching techniques which do not use the structure of the ontologies can use different types of information about the concepts themselves. The techniques which use these different types of information can be divided into several categories:

1. String-based techniques are often used in order to match names and name descriptions of schema/ontology concepts. These techniques consider strings as sequences of letters in an alphabet. They are typically based on the following intuition: two concepts can be similar if their names are similar. Section 2.3 will go into further details about the string matching techniques.

2. Language-based techniques consider names as words in a natural language. They are based on Natural Language Processing techniques exploiting morphological properties of the input words.

3. Constraint-based techniques are algorithms which deal with the internal constraints being applied to the definitions of entities, such as types, cardinality of attributes, and keys.

4. Linguistic resources such as common knowledge or domain specific thesauri are used in order to match words based on linguistic relations between them (e.g., synonyms, hyponyms) [12]. In this case names of schema/ontology entities are considered as words of a natural language.

5. Alignment reuse techniques represent an alternative way of exploiting external resources, which contain in this case alignments of previously matched schemas/ontologies.

6. Upper level formal ontologies can be also used as external sources of common knowledge. The key characteristic of these ontologies is that they are logic-based systems, and therefore, matching techniques exploiting them can be based on the analysis of interpretations.

The listed techniques all have strengths and weaknesses with regard to the different heterogeneities which can exist between two ontology concepts. For instance a technique which uses linguistic resources can easily detect synonymous concepts but will be unable to handle concepts whose names contain spelling errors. Thus a combination of different techniques will be required to cope with all types of heterogeneities.

2.3 String similarities

The focus of this paper lies on the use of string similarities when applied to ontology mapping. Typically, these are applied to the names of concepts in order to produce a similarity matrix of correspondences. These can then be combined with similarity matrices stemming from difference measures, such that a alignment can be extracted.

String distance functions map a pair of strings s and t to a real number r where smaller values indicate a higher similarity between s and t. Similarity functions are analogues except that higher values of r indicate a higher similarity. To avoid confusion to the reader the value r is the one defined by similarity functions. The algorithms used to determine string similarities can be split up in multiple categories depending on their underlying logic to compare strings. First there are algorithms which look at the number of edit operations needed to transform one string into another for example the *Levenshtein* similarity [10]. Then there are algorithms which look at the number of matching characters in both strings for example the *Jaro* similarity [8]. Commonly, the *Winkler* extension [16], which increases the similarity of pairs of strings that have a common prefix, is applied to the *Jaro* similarity. Another category of algorithms are token based, strings are split up into tokens, like the *Jaccard* similarity [7]. There are also algorithms which combine multiple similarities to assign scores to pairs of strings, these are called hybrid similarity functions.

2.3.1 Levenshtein

One important subclass of distance functions are *Edit-distance functions*, which use the number of edit operations required to convert string s to string t. The most considered operations are character insertion, deletion, and substitution. Each of these operations will have a cost assigned to them. The costs assigned to an operation can be static or trained. We will consider the *Levenshtein* distance [10] which assigns a unit cost to each of the edit operations. Given strings s and t, the cost of an operation c_i, where i identifies the type of performed operation, and the quantity x_i which indicates how often an operation of type i needs to be

performed to convert s into t, the Levenshtein distance, which utilizes the three above mentioned operation, can be computed as follows:

$$Levenshtein(s,t) = \sum_{i=0}^{3} c_i \cdot x_i \tag{1}$$

2.3.2 Jaro

The Jaro algorithm [8] is not based on edit operations but determines its similarity by looking at the number of matching characters between two strings and their relative position. Given two strings $s = a_1, a_2...a_K$ and $t = b_1, b_2...b_L$, define a character a_i in s to be common with t when there is a $b_j = a_i$ in t such that $i - H \leq j \leq i + H$, where $H = \frac{\min|s|,|t|}{2}$. Let $s' = a'_1, a'_2...a'_{K'}$ be the characters in s which are common with t (in the same order they appear in s) and let $t' = b'_1, b'_2...b'_{L'}$ be analogous; now define a *transposition for* s', t' to be a position i such that $a'_i \neq b'_i$. Let $T_{s',t'}$ be half the number of transpositions for s' and t'. The Jaro similarity is defined as:

$$Jaro(s,t) = \frac{1}{3} \cdot \left(\frac{\mid s' \mid}{\mid s \mid} + \frac{\mid t' \mid}{\mid t \mid} + \frac{\mid s' \mid - T_{s',t'}}{\mid s' \mid} \right) \tag{2}$$

2.3.3 Jaro-Winkler

A very well known extension to the Jaro algorithm is the Winkler extension [16]. This extension uses the length of the of the longest common prefix of s and t to assign more favourable ratings to pairs of strings which contain identical prefixes. This extension can be used in combination with any similarity but it is most commonly applied to the Jaro similarity. Let P be the length of longest common prefix, then define $P' = \max(P, 4)$ then the Jaro-Winkler similarity is defined as:

$$\begin{aligned} Jaro\text{-}Winkler(s,t) = \\ Jaro(s,t) + \frac{P'}{10} \cdot (1 - Jaro(s,t)) \end{aligned} \tag{3}$$

2.3.4 Jaccard

This algorithm is a token-based distance measure, which can be applied to strings which have been preprocessed into tokens, called tokenization. Tokenization is the process of demarcating and possibly classifying sections of a string of input characters. The strings to be compared are considered to be multisets of words (or tokens). The *Jaccard similarity* [7] between two word sets S and T which is defined as:

$$Jaccard(S,T) = \frac{\mid S \cap T \mid}{\mid S \cup T \mid} \tag{4}$$

2.3.5 SoftTFIDF

Some background information is required in order to fully detail the *SoftTFIDF* similarity. The *TFIDF* [9] weighting scheme for document vectors, to which the *cosine* similarity is commonly applied, is a measure that is widely used in the information retrieval community for document retrieval. This measure depends, like the Jaccard similarity, on common elements between the two sets of tokens, but here the elements are weighted. The weights assigned to tokens w are larger when those tokens are rare in the collection of strings from which s and t are drawn. The similarity can then be defined as:

$$TFIDF(S,T) = \sum_{w \in S \cap T} V(w,S) \cdot V(w,T)$$

where $V'(w,S)$ is defined as the TF-IDF weight of the token w in the token vector of S and the function $V(w,S) = V'(w,S) / \sqrt[2]{\sum_{w'} V'(w',S)})$ is defined as the TF-IDF weight of token w related by the magnitude of the token vector of S. The SoftTFIDF algorithm, proposed by Cohen et al.[2], extends the notion of $S \cap T$ such that it includes tokens which are similar according to a secondary similarity function. Since it utilizes a secondary similarity function denoted as sim' the SoftTFIDF can be categorized as a *hybrid similarity function*. Let $CLOSE(\theta, S, T)$ be the set of words $w \in S$ such that there is some $v \in T$ such

that $dist'(w,v) > \theta$, and for $w \in CLOSE(\theta,S,T)$ and let $D(w,T) = \max_{v \in T} dist'(w,v)$. Then the SoftTFIDF similarity is defined as:

$$SoftTFIDF(S,T) = \sum_{w \in CLOSE(\theta,S,T)} V(w,S) \cdot V(w,T) \cdot D(w,T) \tag{5}$$

3 Proposed extension

The proposed extension is mainly focused on ontology mapping but will also be benchmarked on other datasets containing real world data. This extension came to mind when studying the datasets in the field of ontologies, since concepts defined there are very likely to have high similarity because of the intuition when naming the concepts. To clarify this see the figure below, which is a small part of two ontologies in the OAEI dataset.

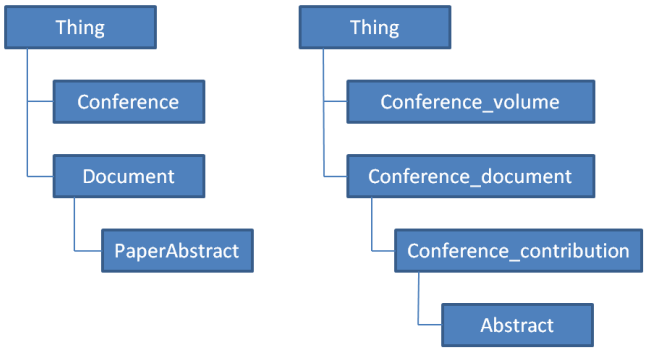

Figure 1: Two partial ontologies from the OAEI-conference dataset

These two example ontologies are part of the matching task using the OAEI dataset. A human inspecting these two example ontologies would quickly realize that the *Conference* and *Conference_volume* denote the same meaning as well as the *Document* concept is the same as the *Conference_document* concept. Likewise, the concepts *PaperAbstract* and *Abstract* also denote the same meaning. The *Winkler* extension to an algorithm only takes into account prefixes when comparing two strings, thus the corresponding concepts *PaperAbstract-Abstract* and *Document-Conference_document* do not receive an increase of their similarity value when applying this extension. Intuitively, given that these pairs of names share a substring of considerable size, in this example at the suffix position, one would want allocate these pairs a higher similarity value opposed to their edit-distance based similarity. Hence, an extension is desired which also increases the similarity of strings if these share a non-prefix substring.

The extension researched in this paper utilizes the measure of the longest common substring between two strings, referred to as the LCS-Extension (Longest Common Substring). On its own this measure can be utilized as a string similarity as well, as evidenced by the reasearch of Stoilos et al. [14] performed on a benchmark dataset, making it a suitable candidate for combination with an edit distance. Whereas the *Winkler* extension is limited to the length of a common substring that is also a prefix of both strings, the proposed extension utilizes the length of the longest common substring regardless of its position in any of the two input strings. Let sim denote the similarity used as basis for the extension, $LCS(s,t)$ be the length of the longest common substring of s and t and S a scaling factor such that $0 \leq S \leq 1$, the proposed LCS extension can then be defined as follows:

$$LCS\text{-}Extension(s,t) = sim(s,t) + \frac{LCS(s,t)}{\min(s,t)} \cdot S \cdot (1 - sim(s,t)) \tag{6}$$

The Winkler extension utilizes the length of the common prefix up to the length of 4 characters for the similarity adjustment. However, intuitively one could argue that the longer a common substring of two arbitrary strings is, the more likely it is that the meanings of these two strings correspond with each other. Hence, the proposed extension does not impose a limit on the computed substring length, but contrasts this length with the longest possible substring length, being the total length of the smaller of the two input strings. The proposed extension will be evaluated using different similarities as a basis.

4 Experiments

To compare the proposed extension with the other algorithms discussed in section 2.3, two datasets are used. The first dataset, stemming from the OAEI 2010 competition [4], contains a series of matching tasks between ontologies describing the conference domain, where the string metrics are applied to the names of the ontology concepts. The second dataset is a record-matching dataset, stemming from the research by Cohen et al. [2]. It contains a series of record matching tasks describing various domain, such as the names of animals and businesses.

4.1 Blocking method

When evaluating a similarity measure it is preferred to compute all pairwise similarities between two ontologies. This can result in large lists which are not computationally feasible. It is desired to pre-process the data, using so called blocking methods. For this research the same blocking method has been applied as in the evaluation by Cohen et al. [2] An example illustrating the intuition behind blocking; in statistical record linkage, it is common to group records by some variable which is known a priori to be usually the same for matching pairs. For example when matching records containing address information it is common to only consider pairs which have the same zip code.

The data used in this paper does not contain individuals for each concept, so there is little prior information available for pre-processing purposes. However the data is already partitioned into two mutually exclusive lists which reduces the number of pairs to be considered. To block this data, knowledge-free approaches are needed to reduce the number of considered pairs. The *blocking task* of two sets A and B selects all pairs of strings $(s, t) \in A \times B$ such that s and t share some substring v which appears in at most a fraction f of all names. This method is called the token blocker. Another method for blocking the data is using n-grams to only consider strings which share an n-gram. For the moderate-size test sets considered here, we used $f = 1$. On the datasets which have been used in this research, the token blocker finds between 93.3% and 100.00% of the correct pairs for the different matching tasks, with an average of 98.9%.

4.2 Evaluation

The algorithms will all be evaluated using *precision* and *recall* values. These values are defined, in terms of true positives, false positives and false negatives of a retrieved list with regard to a reference list, as follows:

$$Precision = \frac{tp}{tp + fp} \quad (7) \qquad Recall = \frac{tp}{tp + fn} \quad (8)$$

Precision and recall are set-based measures, stemming from the field of information retrieval [5]. These evaluate the quality of an unordered set of retrieved ontology concepts according to their correctness and completeness. The investigated metrics will be evaluated using interpolated precision values at recall levels of 0.0, 0.1, ..., 0.9, 1.0, which are obtained by analysing the ranked list of retrieved correspondences. The rule to obtain the precision value at recall level i is to use the maximum precision obtained from the concept for any actual recall level greater than or equal to i. Note that the non-interpolated precision is not defined for recall values of 0, as opposed to the interpolated precision at recall level $i = 0$.

Before any of the similarities are evaluated on the datasets, these are blocked using the token blocker. All pairwise combinations of concepts are evaluated using the blocking method, after which the tested metrics are applied on the remaining pairs of concept names. The interpolated precision values for each mapping task are combined using the average interpolated precision.

4.3 Comparison with the Winkler extension

This experiment will compare the *Winkler* extension with the proposed *LCS* extension. This will show whether the intuition behind the proposed extension leads to a better performance than the more specific *Winkler* extension. Preliminary experiments revealed that a scaling factor of $S = 0.8$ produced the highest performance for the *LCS* extension, with significant sub-par performances only observable at low values of S and $S = 1$ The extensions were compared using both the *Jaro* and *Levenshtein* metric as base similarity.

The first comparison, seen in Figure 2, has been performed on the conference dataset. In the recall interval from 0 to 0.4 there is a minimal difference in the performance of the tested metrics, neither showing

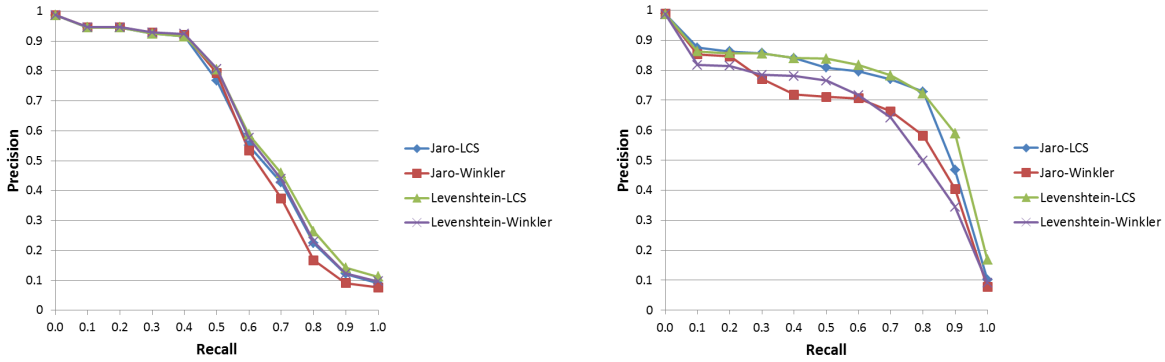

Figure 2: Comparison of the Winkler extension against the LCS extension when applied to two different base similarities on the Conference dataset (left) and Cohen dataset (right).

an advantage. The recall interval of 0.4 and 0.6 displays more pronounced differences, of which the most notable is that the *Jaro-LCS* metric performed slightly worse than the remaining metrics.

From a recall values of 0.6 and higher it appears that the proposed extension displays a superior performance with regard to the Winkler extension applied to the same base similarity.

When comparing the metrics on the Cohen data set, see Figure 2, the proposed *LCS* extension outperforms both *Winkler extension* based metrics by a substantial percentage. The *Levenshtein-Winkler* metric performs worse at a recall of 0.1 whereas the *Jaro-Winkler* performs almost similar up until a recall of 0.2. At the remaining recall values the *LCS extension* outperforms the *Winkler extension* by a significant margin, peaking at recall values of 0.8 and 0.9 with an increase of precision of at least 0.1.

4.4 Comparison with other measures

In this experiment, the best performing configuration of our proposed extension is compared to other established methods from this field. The *LCS extension* will be applied to the *Levenshtein* similarity, due to its superior performance as seen in sub-section 4.3.

The performed evaluation on the conference data set, see figure 3, reveals that the token based *Jaccard similarity* displays the worst performance of the tested metrics. The hybrid *SoftTFIDF* metric is performing slightly worse than the edit based distances on lower recall values, but displays a superior performance on higher recall values. The edit-based distances all display a similar performance curve, with some of them performing strictly better, of which the *Levenshtein-LCS* performs best considering all the recall values.

The comparison shown in figure 3 is obtained by comparing all the algorithms on the Cohen dataset. It is evident that on this data set token-based distance functions outperform the majority of the edit-based

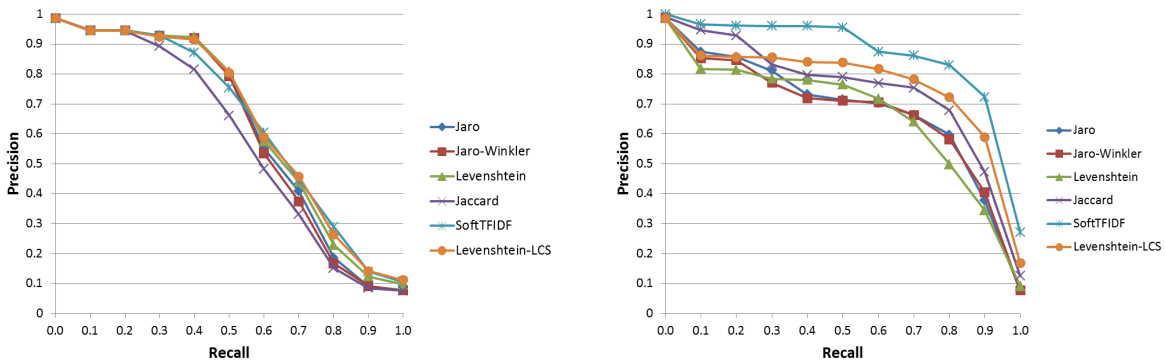

Figure 3: Comparison of all tested similarity measures on the Conference dataset (left) and Cohen dataset (right)

distance functions, especially at lower recall values. Since the *SoftTFIDF* combines a token-based approach with an edit-distance based approach, by using an edit-distance metric as secondary distance function, it outperforms all tested metrics by a significant margin. The *Jaccard similarity* outperforms all tested edit-distance based metrics for recall values up to 0.3. However, for recall values of 0.3 and higher, the proposed extension applied to the *Levenshtein* metric significantly outperforms the tested edit-distance based metrics as well as the *Jaccard* metric.

5 Conclusion

In this paper, we proposed a generalization of the Winkler extension using the measure of the longest common sub-string. We used the Jaro and Levenshtein similarity as base in order to compare our generalization with the Winkler extension. The experiments show that our extension outperforms the Winkler extension for either base similarity on both datasets, the differences being more pronounced when evaluating the record-matching dataset. Contrasting the proposed extension with contemporary metrics revealed that it outperforms all tested metrics, except for the hybrid *SoftTFIDF* metric.

The proposed extension has been applied to edit-distance based functions in the performed experiments. Future research could investigate the potential improvements of the extension when being incorporated into a hybrid distance function. Also, it is possible that a performance gain can be achieved by analysing the input strings for stop-words, such that concept names for which the common substring is a stop word do not receive an increased similarity value.

References

[1] W. W. Cohen. Data integration using similarity joins and a word-based information representation language. *ACM Trans. Inf. Syst.*, 18(3):288–321, July 2000.

[2] W. W. Cohen, P. Ravikumar, and S. Fienberg. A comparison of string distance metrics for name-matching tasks. 2003.

[3] J. Euzenat. Towards a principled approach to semantic interoperability. In Asuncin Gmez Prez, Michael Gruninger, Heiner Stuckenschmidt, and Michael Uschold, editors, *Proc. IJCAI 2001 workshop on ontology and information sharing, Seattle (WA US)*, pages 19–25, 2001.

[4] J. Euzenat, A. Ferrara, C. Meilicke, A. Nikolov, J. Pane, F. Scharffe, P. Shvaiko, H. Stuckenschmidt, O. Svab-Zamazal, V. Svatek, and C. Trojahn. Results of the ontology alignment evaluation initiative 2010. 2010.

[5] F. Giunchiglia, M. Yatskevich, P. Avesani, and P. Shvaiko. A large dataset for the evaluation of ontology matching. *Knowl. Eng. Rev.*, 24:137–157, June 2009.

[6] N. Guarino. The role of ontologies for the semantic web (and beyond). Technical report, Institute for Cognitive Sciences and Technology (ISTCCNR), 2004.

[7] P. Jaccard. Nouvelles recherches sur la distribution florale. *Bulletin de la Sociète Vaudense des Sciences Naturelles*, 44:223–270, 1908.

[8] M. A. Jaro. 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.

[9] K. Sparck Jones. Document retrieval systems. chapter A statistical interpretation of term specificity and its application in retrieval, pages 132–142. Taylor Graham Publishing, London, UK, 1988.

[10] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. Technical Report 8, 1966.

[11] N. F. Noy and M. Klein. Ontology evolution: Not the same as schema evolution. *Knowledge and Information Systems*, 6:428–440, 2004.

[12] F. C. Schadd and N. Roos. Improving ontology matchers utilizing linguistic ontologies: an information retrieval approach. In *Proceedings of the 23rd Belgian-Dutch Conference on Artificial Intelligence (BNAIC 2011)*, 2011.

[13] P. Shvaiko and J. Euzenat. A survey of schema-based matching approaches. In Stefano Spaccapietra, editor, *Journal on Data Semantics IV*, volume 3730 of *Lecture Notes in Computer Science*, pages 146–171. Springer Berlin / Heidelberg, 2005.

[14] G. Stoilos, G. Stamou, and S. Kollias. A string metric for ontology alignment. In *Proceedings of ISWC*, pages 624–637, 2005.

[15] M. Uschold and M. Gruninger. Ontologies and semantics for seamless connectivity. *SIGMOD Rec.*, 33(4):58–64, December 2004.

[16] W. E. Winkler. String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage. Technical report, 1990.