Similarity Recognition in the Web of Data

Alfio Ferrara
Dipartimento di Informatica
Università degli Studi di Milano
Via Comelico 39
20135 - Milano, Italy
alfio.ferrara@unimi.it

Lorenzo Genta
Dipartimento di Informatica
Università degli Studi di Milano
Via Comelico 39
20135 - Milano, Italy
lorenzo.genta@unimi.it

Stefano Montanelli
Dipartimento di Informatica
Università degli Studi di Milano
Via Comelico 39
20135 - Milano, Italy
stefano.montanelli@unimi.it

ABSTRACT

In the web of data, similarity recognition is the basis for a variety of resource-consuming activities and applications, including data recommendation, data aggregation, and data analysis. In this paper, we propose HMatch4, a novel instance matching algorithm for similarity recognition, which has been developed on the ground of our experience with HMatch3 [3].

1. INTRODUCTION

In the web of data, the capability to recognize the degree of similarity between different descriptions of web resources is getting more and more crucial for a number of purposes [2, 5]. Focused techniques specifically conceived for the web of data are required to address the peculiar aspects of similarity evaluation in such a context. Two main issues need to be considered.

Similarity is not identity. In the literature, traditional approaches/tools for similarity recognition are based on the idea of comparing resources by analyzing their features. In these solutions, the goal is to recognize the *identity* between two resources, namely the capability to detect when the two descriptions refer to the same real object. From this perspective, identity is seen as a special case of similarity characterized by a high similarity value (i.e., high number of shared features). Low similarity values are usually interpreted as a *non-identity* result, meaning that the two considered resource descriptions refer to different real objects. Consider the following example:

tiger woods	**arnold schwarzenegger**
profession: golfer	profession: bodybuilder
nationality: united states	profession: politician
	nationality: united states

In this case, the degree of similarity between tiger woods and arnold schwarzenegger is quite low, because they actually have only one feature in common (i.e., nationality). According to traditional instance matching approaches, the similarity value between the two resources is discarded, concluding that they do not represent the same person. Such a behavior is correct, however, matching techniques for the web of data have to be capable of capturing and preserving the correct degree of similarity both when the considered resources are identical or very similar and when they are only quite similar or even completely different. This can be useful in many application scenarios, like for example in web data classification where the goal can be to aggregate resources based on the similarity over one (or few) specific features (see for example [4]).

The scale as a key issue. In the web of data, matching usually involves very large data collections, potentially composed by hundred of thousands of web resources described by millions of features. In this scenario, the intrinsic limitation of existing instance matching approaches to similarity evaluation is due to the cost of directly comparing all the pairs of resource items in order to compute their degree of similarity within a considered dataset. In a classical *comparison-based* matching approach, this means that a matching operation is required to calculate the similarity degree for each possible pair of items. For a dataset of n items, this means that $O(n^2)$ matching operations are required in the worst case to compare each item against all the other items in the dataset. Optimization strategies have been proposed in the literature to reduce the number of comparison operations and to increase the performance of the overall matching process [5]. However, to the best of our knowledge, the currently available instance matching tools are still affected by severe limitations in terms of scalability, which usually means that in real systems the similarity recognition task need to be executed offline in case of very large collections of data.

In this paper, we propose HMatch4, a novel instance matching algorithm for similarity recognition, which has been developed on the ground of our experience with HMatch3 [3]. HMatch4 is natively conceived for working in the web of data, where the above matching issues are properly considered and addressed. In the following, we first describe the HMatch4 techniques and related algorithm (Section 2 and 3). Then, we provide the results of a preliminary evaluation obtained by comparing HMatch4 against HMatch3 and other popular tools for instance matching, namely LogMap [7] and SLINT+ [8] (Section 4). Related work and concluding re-

marks are finally discussed (Section 5 and 6).

2. THE HMATCH4 PROCESS

We first describe our model for representation of the web-of-data resources, and then we present the HMatch4 matching process.

2.1 Modeling web resource items

In HMatch4, we rely on the use of an internal data model called *web resource item* (*wri*) for representation of the resources to match. A wri element is featured by a set of feature-value pairs and it is defined as follows:

$$wri = \{(f, v)_1 \dots (f, v)_k\}$$

where each pair $(f, v) \in wri$ represents a feature-name f and the corresponding feature-value v.

Wrapping resources to the wri model. The wri model has been conceived to support matching of different kinds of web-of-data resources, like for example social data (e.g., Facebook, Twitter resources) and linked data (e.g., Freebase, DBpedia resources). The idea of modeling resources as sets of feature-value pairs is motivated by the need to deal with a number of different native formats that are commonly employed for description of web-of-data resources. Appropriate wrapping operations are required to transform the native web resource representation into a wri-based representation. In general, this wrapping step is straightforward. A feature-value pair $(f, v) \in wri$ is created for each property and corresponding value within the native description of the considered resource. For instance, in case of a social data resource like a tweet, a feature-value pair is defined in the wri representation for each tweet field (e.g., id, text, user, lang, place, created_at, entities). A similar approach is enforced to generate a wri representation when a linked data resource URI extracted from a repository $\mathcal{R}$ is considered. In particular, a feature-value pair is created for each property name and corresponding property value that is directly connected with URI in the RDF specification extracted from the repository $\mathcal{R}$. In case that URI has a property name p associated with multiple property values $v_1 \dots v_m$, a feature-value pair (p, v_j) is created in the wri description for each value v_j with $j \in [1, m]$.

Example. As an example, we show the wri description for a linked data resource featuring the famous athlete muhammad ali. Such a description is extracted from the Freebase repository by only considering the properties profession, type, and nationality.

muhammad_ali
(profession, athlete),
(profession, professional boxer),
(type, olympic athlete),
(nationality, United States of America).

2.2 Matching process

HMatch4 works on a dataset $\mathcal{D}$ of wri elements to match and it produces a similarity matrix as a result.

Spirit of HMatch4. The idea is to measure the similarity degree between two items by calculating their number of common feature-value pairs in the wri representations. Given wri_1 and wri_2, this can be determined by calculating the set of pairs (f, v) that belong to both the considered items (i.e., $wri_1 \cap wri_2$). As a difference with classical comparison-based approaches, HMatch4 proposes a sort of *index-based* matching approach where the similarity degree of two items is the result of an indexing operation and a "pair-by-pair" comparison between wri elements is not required. The key idea of HMatch4 is to consider each single wri in the dataset and to index all the possible subsets of feature-value pairs belonging to wri that can be relevant for detecting a similarity with other wri elements. Two items wri_1 and wri_2 are similar if they share the same entry in the index, meaning that they have a common subset of feature-value pairs in their wri representations (i.e., the feature value pairs of the index entry). The similarity degree is assessed by measuring the size (i.e., cardinality) of the shared subset of feature-value pairs.

Matching process. The matching process of HMatch4 is articulated in three main steps, namely *configuration*, *execution*, and *assessment* (see Figure 1). The **configuration step** defines the setup of the matching execution and it specifies the requirements to be satisfied by two items wri_1 and wri_2 for being considered as similar. We call *feature-set* F the set of all the features involved in the specification of wri elements within the considered dataset $\mathcal{D}$, namely:

$$F = \{\bigcup_{i=1}^{n} fs(wri_i)\}$$

where $n = |\mathcal{D}|$ is the number of items within the dataset $\mathcal{D}$ and $fs(wri) = \{f_j \mid (f_j, v) \in wri\}$ is the set of features characterizing the feature-value pairs of wri. Then, we calculate the power set $\mathcal{P}(F)$ containing all the possible subsets of features over F. Then, we define the set $\mathcal{F} \subset F$ as follows:

$$\mathcal{F} = \{rfs \mid rfs \in F \wedge \frac{\mid rfs \mid}{\mid F \mid} \geq th_s\}$$

where $th_s \in (0, 1]$ is a similarity threshold and it determines the minimum similarity degree that is required to consider two items as matching items. A set $rfs \in \mathcal{F}$ is called *relevant feature-set* and it represents a combination of features to be considered for similarity recognition. The rationale of our matching process is that two items wri_1 and wri_2 are similar iff they share a set of feature-value pairs where the features coincides with a set $rfs \in \mathcal{F}$.

The **execution step** creates an index structure $\mathcal{I}$ containing entries for the combinations of feature-value pairs within wri descriptions that are relevant for similarity recognition. An index entry $ie \in \mathcal{I}$ has the form $ie = \langle rfs, fvl, wp \rangle$, where $rfs \in \mathcal{F}$ is a relevant feature-set, fvl is a list of feature-value pairs, and wp is a set of wri elements belonging to $\mathcal{D}$. Given an index entry ie and the associated relevant feature-set $rfs = \{f_1, \dots, f_s\}$, the corresponding list of feature-value pairs fvl has the form $fvl = (f_1, v_1) \mid \dots \mid (f_s, v_s)$, and wp contains the wri elements that provide fvl in their wri representation. For each item $wri \in \mathcal{D}$, we create an entry in the index $\mathcal{I}$ for those combinations of feature-value pairs of wri that are based on a relevant feature-set belonging to $\mathcal{F}$. Details about the HMatch4 algorithm for creating the index $\mathcal{I}$ are presented in Section 3.

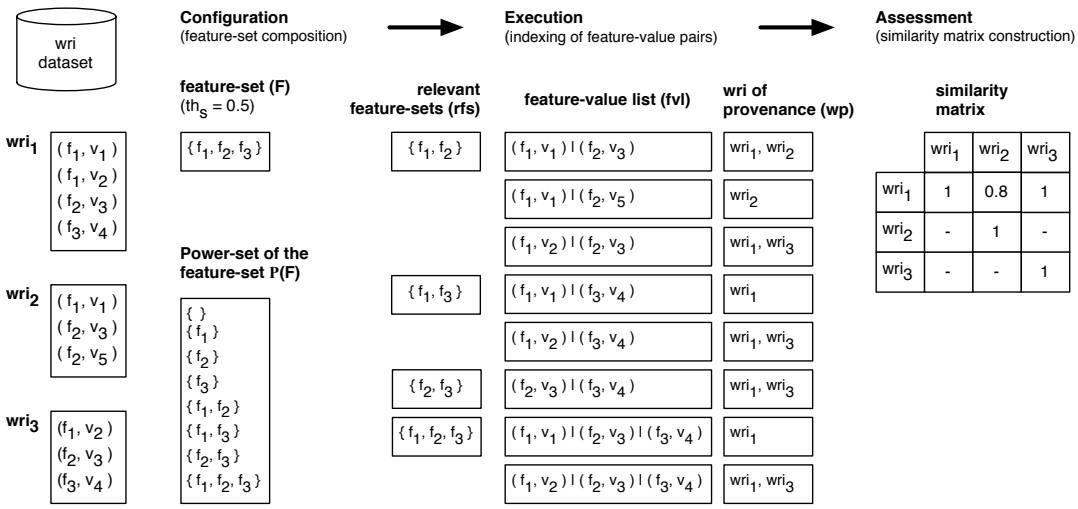

Figure 1: The similarity recognition process of HMatch4

The **assessment step** generates the similarity matrix $\mathcal{M}$ for the wri items of the dataset $\mathcal{D}$. Given two items wri_1 and wri_2, their similarity value $sim(wri_1, wri_2)$ is determined by querying the index $\mathcal{I}$ and by extracting the index entry shared by wri_1 and wri_2 (if it exists). Given an index entry ie where $wri_1, wri_2 \in wp$, the similarity value $sim(wri_1, wri_2)$ is calculated through the Dice's coefficient formula:

$$sim(wri_1, wri_2) = \frac{2 \cdot \mid rfs \mid}{\mid fs(wri_1) \mid + \mid fs(wri_2) \mid}$$

where $fs(wri)$ is the set of features characterizing the feature-value pairs of wri and rfs is the relevant feature-set associated with the index entry ie, meaning that the items wri_1 and wri_2 share feature-value pairs for all the features in rfs. According to this definition, the similarity values computed by HMatch4 are symmetric (i.e., $sim(wri_1, wri_2) = sim(wri_2, wri_1)$) and thus the resulting matrix $\mathcal{M}$ is upper triangular. It is possible that two items wri_1 and wri_2 have not a shared entry in the index structure. In HMatch4, this means that $sim(wri_1, wri_2) = 0$. It is also possible that two or more index entries are shared by two items wri_1 and wri_2, meaning that the two items have more relevant feature-sets in common. In this case, the entry with $max(\mid rfs \mid)$ is selected for calculating $sim(wri_1, wri_2)$.

Example. We consider the wri about muhammad ali previously introduced and the following wri descriptions about michael_jordan and george foreman.

michael_jordan
(profession, athlete),
(type, olympic athlete),
(type, celebrity).

george_ foreman
(profession, professional boxer),
(type, olympic athlete),
(nationality, United States of America).

According to these descriptions, the feature-set $F = \{$ profession, type, nationality $\}$ is generated. By setting a similarity threshold $th_s = 0.5$, we obtain the relevant feature-sets $\mathcal{F} = \{\{$profession, type$\}, \{$profession, nationality$\}, \{$type, nationality$\}, \{$profession, type, nationality$\}\}$. The resulting index structure is shown in Figure 2. Thus, the similarity matrix generated in the assessment step is the following:

	m. ali	m. jordan	g. foreman
m. ali	1.0	0.8	1.0
m. jordan	-	1.0	0.0
g. foreman	-	-	1.0

For calculation of $sim($m. ali,m. jordan$)$, we consider the first entry in the index structure of Figure 2 that is shared by m. ali and m. jordan (see column wp). Thus, we obtain: $sim(m.ali, m.jordan) = (2 \cdot 2)/(3 + 2) = 0.8$. Moreover, we note that $sim(m.jordan, g.foreman) = 0.0$. This is due to the fact that m. jordan and g. foreman only share the feature-value pair type, olympic athlete that does not coincide with a relevant feature-set, meaning that the similarity between m. jordan and g. foreman is not sufficient for being recognized in the current HMatch4 configuration ($th_s = 0.5$).

3. THE HMATCH4 ALGORITHM

In this section, we present the HMatch4 algorithm used in the execution step for creation of the index structure $\mathcal{I}$. The algorithm is shown in Figure 3. The algorithm is implemented by the function INDEXING$(\mathcal{D}, \mathcal{F})$, which takes the dataset $\mathcal{D}$ and the relevant feature-set $\mathcal{F}$ as input. The function initializes the index $\mathcal{I}$ as a map where keys are numeric values and values are sets of wri (line 2). Then, it takes into account all the wri in $\mathcal{D}$. For each wri and for each relevant relevant feature-set rfs in $\mathcal{F}$, we first initialize an empty set of values V (line 5). As an example, let us take into account the wri_1 of Figure 1, which is composed by the feature-value pairs $\{(f_1, v_1), (f_1, v_2), (f_2, v_3), (f_3, v_4)\}$, and the relevant feature-set $\{f_1, f_2\}$. For each feature f_i in rfs, we insert into V all the feature-values pairs having f_i as feature (lines 7-8). In our example, the set V at the end of this process is composed by the elements $V = \{ \{(f_1, v_1), (f_1, v_2)\},$

Relevant feature-set rfs	Feature-value list fvl	wri wp
{profession, type}	(profession, athlete) (type, olympic athlete)	m ali, m. jordan
	(profession, athlete) (type, celebrity)	m. jordan
	(profession, professional boxer) | (type, olympic athlete)	m. ali, g. foreman
{profession, nationality}	(profession, athlete) | (nationality, United States of America)	m ali
	(profession, professional boxer) | (nationality, United States of America)	m. ali, , g. foreman
{type, nationality}	(type, olympic athlete) | (nationality, United States of America)	m.ali, g. foreman
{profession, type, nationality}	(profession, athlete) | (type, olympic athlete) | (nationality, United States of America)	m ali
	(profession, professional boxer) | (type, olympic athlete) | (nationality, United States of America)	m ali, g. foreman

Figure 2: Index structure for the wri descriptions about muhammad ali**,** micheal jordan**, and** george foreman

```
1: function INDEXING(D, F)
2:     I ← a key-value map of the form <number: set>
3:     for all wri ∈ D do
4:         for all rfs ∈ F do
5:             V ← {}
6:             for all f ∈ rfs do
7:                 F ← {(f_i, v_i) | (f_i, v_i) ∈ wri ∧ f_i = f}
8:                 add F to V
9:             end for
10:            X ← v_1 × v_2 × ... × v_|V| | v_i ∈ V
11:            for all x ∈ X do
12:                h ← hash(x)
13:                if h ∈ keys(I) then
14:                    add wri to I[h]
15:                else
16:                    I[h] ← {wri}
17:                end if
18:            end for
19:        end for
20:    end for
21:    return I
22: end function
```

Figure 3: The HMatch4 **execution algorithm**

$\{(f_2, v_3)\}\}$. Now, we process each element in the cartesian product X of the sets in V, which are $\{(f_1, v_1), (f_2, v_3)\}$ and $\{(f_1, v_2), (f_2, v_3)\}$ (lines 10-11). The idea behind this step is that features having more than one value, such as f_1 in the example, are considered separately and in combination with all the other feature values. For each element x of X, we insert a new entry in the index $\mathcal{I}$. In particular, we obtain a numerical index key as the hash-value of x. We note that, in this paper, we assume to have just a simple hashing function capable of providing a unique value for each combination of feature-value pairs. The development of a more powerful hashing function is one of the goals of our future work. According to this procedure (lines 12-16), the index $\mathcal{I}$ will contain one entry for each combination of values in the relevant feature-set rfs. In our example, given all the feature-sets of Figure 1, we generate 7 entries, 3 for $\{f_1, f_2\}$, 2 for $\{f_1, f_3\}$, 1 for $\{f_2, f_3\}$, and 2 for $\{f_1, f_2, f_3\}$. In such a way, when a subsequent wri_j is processed, if it has one or more combinations of feature-value pairs that are equal to those of wri, it will be inserted in the same set of wri in the index, denoting the fact that there is a similarity between wri_j and wri.

4. EXPERIMENTAL RESULTS

The goal of our experimentation is to evaluate HMatch4 in terms of i) the quality of similarity recognition measured by precision and recall; ii) the scalability of HMatch4 when matching a growing number of web resources. Considering the quality assessment, we performed a comparison against a ground-truth set of mappings produced by human users. The scalability evaluation is performed on both time and space consumption in comparison with the matching tools LogMap and SLINT+ [7, 8]. These tools have been chosen for their known efficiency and for the availability of a working prototype. The scalability tests are performed on an automatically produced dataset based on multiple replications of a base dataset. Both the quality and scalability tests have been performed by comparing HMatch4 and HMatch3 [3], our previous version of matching tool.

4.1 Experiment setup

In this section, we discuss quality assessment and scalability evaluation.

Quality assessment. The ground truth has been produced by exploiting a novel crowdsourcing approach called *Liquid Crowd*. A crowdsourcing approach consists in reducing a problem in a set of elementary units of work that are distributed to a (possibly) large number of human workers. Each worker participates giving the solution for one or more work units and receives a reward (e.g., money, personal satisfaction or other benefits) proportional to the completed amount of work. The main idea behind Liquid Crowd is to change the definition of *worker* from a single user to a *group* of users. A work unit is considered *accepted* only if the assigned group reaches a consensus on the produced answer (i.e., the qualified majority of users converge on the same answer). In our experimentation, the ground truth for quality assessment is built on 58 individuals from Freebase repository with a total number of 275 feature-value pairs. Thus, the *work units* have been structured as a blind evaluation of a pair of web resources. For instance, the users have to evaluate the similarity of the given resources only knowing their features and features-values without knowing their identifiers (i.e., the names of the resources): this is done to avoid that users exploit their personal knowledge in evaluating the similarity.
In order to complete a work unit, the user has to choose between 4 possible answers: equal (E), very similar (VS), quite similar(QS), unequal (U). Thus, the final mappings produced are in the form $m(wri_x, wri_y) = \{E|VS|QS|U\}$. As a result from the proposed set of work units, the human workers produced 1136 mappings (work units that reached the consensus) with the following distribution: $U = 653, QS = 317, VS = 151, E = 15$. On this dataset we produced 1653 work units. The crowdsourcing session was open for 7 days to any volunteer: 82 persons took part to the experiment with a result of 1136 completed tasks.

Scalability evaluation. For this test, we performed differ-

ent executions of the 4 matching tools by replicating K times the dataset used for quality assessment, for K between 1 and 1461. The number of resources of the performed tests is between 58 (426 RDF triples) and 85144 (888293 RDF triples) and each resource has a mean of 4 feature-value pairs. The time measurements has been performed by evaluating time between the execution of the considered tool and its termination, while the memory measurement has been done by polling the used memory and catching the greatest value during the execution of each tool.

4.2 Results

For quality assessment, we compare the results of HMatch4 and HMatch3 against the ground truth produced by the human users of our crowdsourcing system. The decision to exclude LogMap and SLINT+ arises from the fact that they are proposed to find different representations of the same resource and not to perform similarity recognition, which means that i) the produced values are a representation of *identity* or candidate identity and ii) each resource appears in the results only compared to its best match. In order to make the results produced by HMatch4 and HMatch3 comparable to the ground truth, we converted the continuous values of our tools to the four similarity classes produced by the crowdsourcing system. This has been done creating 4 intervals and 4 association rules mapping each interval to the corresponding similarity class: $[0, 0.17) \rightarrow U, [0.17, 0.5) \rightarrow QS, [0.5, 0.8) \rightarrow VS, [0.8, 1] \rightarrow E$.

The comparison between HMatch4 and HMatch3 is based on 3 measures: *precision*, *recall* and *F-measure* (Figure 4). Given the set of mappings in a specific class produced by the automatic tool as T and the set of mappings of the ground truth in the same class as G, the precision value is computed as $\frac{T \cap G}{T}$ and recall value is computed as $\frac{T \cap G}{G}$, while F-measure is the harmonic mean between precision and recall. The obtained results show that HMatch4 and HMatch3 are very similar in both precision and recall values. We note that both tools behave exactly as humans on inequality recognition, while they have different perceptions on the other similarity classes. This is probably due to human evaluation of similarity that tends to discriminate the importance of similarity based on the name of the feature. We also considered a different comparison approach by converting the classes of the crowdsourcing system to values in the interval $[0, 1]$, in order to perform an overall accuracy measure. This measure has been computed as the inverse of the mean distance of the results produced by our tools against the ground truth. The results of the crowdsourcing system have been converted as follows: $E \rightarrow 1, VS \rightarrow 0.66, QS \rightarrow 0.33, U \rightarrow 0$. The accuracy values obtained are 0.886 for HMatch3 and 0.890 for HMatch4. Also in this case the results are almost the same.

Finally, we present the results of the scalability evaluation. All the tests have been executed on a 4-core Intel(R) Xeon(R) processor (model E5-1620) with a frequency of 3.60GHz and 16 GB of total RAM memory. The results shown in Figure 5 (execution time) represent the trend of the 4 considered tools for the time consumption: the y-axis is the log-scale of the required time while the x-axis is the size of the dataset represented by the maximum number of possible comparisons (i.e., for a dataset of N resources the x-axis shows $N \cdot N$).

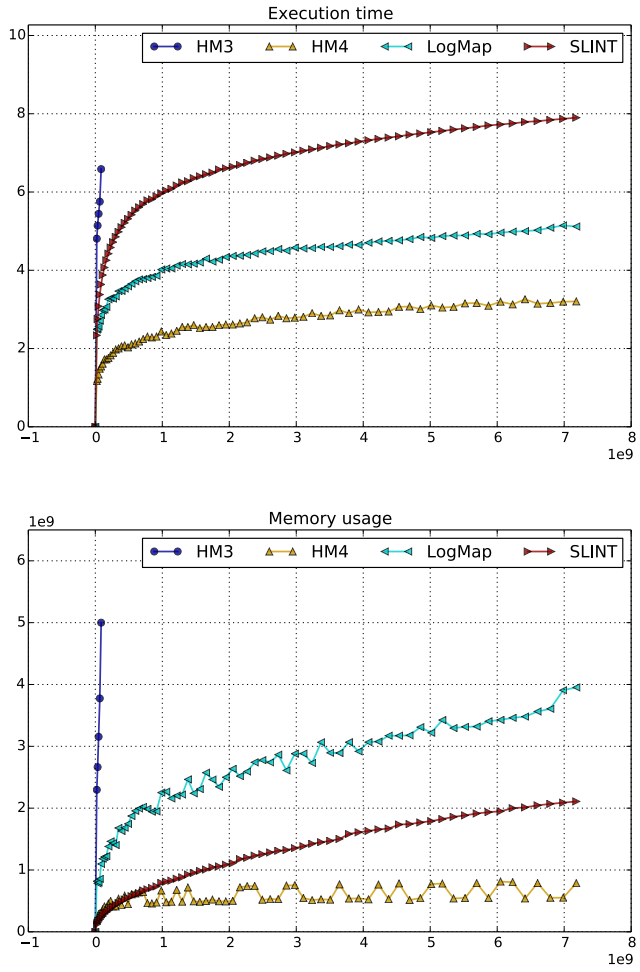

Figure 5: Execution time and memory usage for HMatch3, HMatch4, LogMap, and SLINT+

In Figure 5 (memory usage) the memory consumption of the 4 considered tools is shown. In this case, the y-axis shows the occupied memory in bytes while the x-axis represents the size of the dataset (intended again as the greatest number of possible comparisons). We note that the HMatch3 tool is executed just until the 161^{th} replication due to the excessive required time and memory. In all the executed test cases HMatch4 performed as the best tool between the considered counterparts. In the largest test case, we matched $N \cdot N$ resources with $N = 85144$ (401775 features, 888293 triples): SLINT+ required 2697 seconds, LogMap required 166 seconds and HMatch4 completed the comparison in 24 seconds.

5. RELATED WORK

In the recent years, a lot of research effort has been focused on data matching with a specific attention to instance matching in the framework of the Semantic Web. Most of the existing solutions have been conceived to deal with the so-called *identity-recognition* problem, where the target is to detect when different descriptions extracted from inde-

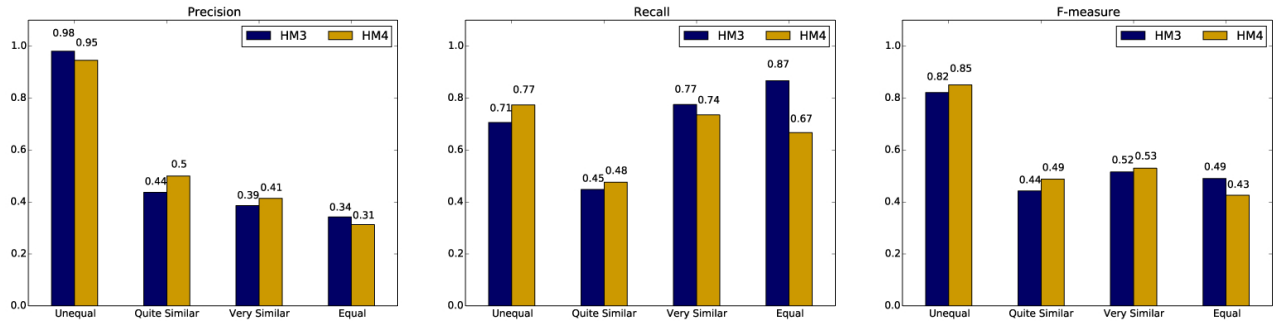

Figure 4: Precision, Recall and F-measure of HMatch3 and HMatch4

pendent web repositories refer to the same individual. On this research line, a reference survey of existing techniques and tools can be found in [5]. The creation of an instance-matching track within the context of the well-known OAEI initiative[1] is a further message that emphasizes the growing attention about the data matching issues. Examples of interesting tools that have been emerging from the OAEI competition are LogMap [7] and SLINT+ [8]. Yet, the focus of these proposed tools is the capability to recognize identities (i.e., *same-as* links) between pairs of web objects.

Approaches based on feature similarity are also relevant with respect to our HMatch4. In this kind of solutions, the objects to match are described through (numeric) feature vectors and the similarity degree is calculated in terms of distances in a n-dimensional space by relying on vector calculus operations. Vector-based matching techniques are usually employed in image similarity recognition and in nearest neighbor search where the items to compare are characterized by feature vectors with n numeric coordinates and the mapping within a n-dimensional space is straightforward [9]. For application of vector-based techniques to web-of-data matching, a transformation of (string-based) feature-value pairs into (numeric) feature vectors is required, but such a kind of transformation is not straightforward.

Finally, the possibility to use hashing solutions to index the object features to match is not completely new in the literature about data matching [6]. The capability to create an efficient data structure for storing similar values in neighbor positions of the index is promising and solutions in this direction are currently appearing [1]. We plan to integrate the use of hashing data structures into HMatch4. We will investigate this issue in the next-future research activities to further increase the performance of execution and assessment steps of the HMatch4 process (see Section 6).

Original contribution. With respect to all the above solutions, the peculiarity of HMatch4 is on the goal of the matching process rather than on the novelty of the proposed techniques. In HMatch4, the target is similarity recognition, based on evaluating the relevance of shared subset of feature-values in the different wri specifications. The use of an index structure is adopted to avoid a direct pair-by-pair comparison of all the items to match, with the aim at improving the overall performance of the matching process.

[1] `http://oaei.ontologymatching.org/`.

6. CONCLUDING REMARKS

In this paper, we presented HMatch4 and related techniques for similarity recognition. HMatch4 has been conceived for application in those contexts where the goal is to evaluate the similarity degree among description of different individuals like for example dimension-based data classification and web data summarization [4]. In future work, we plan to investigate the extension of HMatch4 to support approximate matching. The idea is to enforce hashing techniques that preserve "near" index positions when "near" feature values are recognized. The investigation of hashing techniques based on tree structures is also in the research agenda of HMatch4 for efficient indexing of feature-value pairs.

7. REFERENCES

[1] A. Andoni and P. Indyk. Near-optimal Hashing Algorithms for Approximate nearest Neighbor in high Dimensions. In *Proc. of the 47th IEEE FOCS*, 2006.

[2] D. Bianchini, C. Cappiello, V. De Antonellis, and B. Pernici. P2S: A Methodology to Enable Inter-organizational Process Design through Web Services. In *Proc. of the 21st Int. CAiSE*, 2009.

[3] S. Castano, A. Ferrara, S. Montanelli, and G. Varese. Ontology and Instance Matching. In *Knowledge-driven multimedia information extraction and ontology evolution*. Springer, 2011.

[4] A. Ferrara, L. Genta, and S. Montanelli. Linked Data Classification: a Feature-based Approach. In *Proc. of the 3rd EDBT LWDM Workshop*, 2013.

[5] Ferrara, A. and Nikolov, A. and Scharffe, F. Data Linking for the Semantic Web. *Int. Journal on Semantic Web and Information Systems*, 7(3), 2011.

[6] A. Gionis, P. Indyk, and R. Motwani. Similarity Search in High Dimensions via Hashing. In *Proc. of the 25th VLDB Conference*, 1999.

[7] E. Jiménez-Ruiz, B. C. Grau, Y. Zhou, and I. Horrocks. Large-scale Interactive Ontology Matching: Algorithms and Implementation. In *Proc. of the 20th ECAI*, 2012.

[8] K. Nguyen, R. Ichise, and B. Le. SLINT: A Schema-Independent Linked Data Interlinking System. In *Proc. of the 7th Int. Workshop on Ontology Matching*, 2012.

[9] Y. Tao, K. Yi, C. Sheng, and P. Kalnis. Quality and Efficiency in high Dimensional nearest Neighbor Search. In *Proc. of ACM SIGMOD*, 2009.