A Schema-Free Instance Matching Algorithm Based on Virtual Document Similarity

Siham Amrouch
LIM Laboratory,
Souk Ahras University,
Algeria
sihamamrouch@yahoo.fr

Sihem Mostefai
MISC Laboratory,
Constantine University,
Algeria
sihem.mostefai@univ-constantine2.dz

Abstract: *With the continuous development of semantic web, especially of the web of data, several knowledge bases expressed by ontologies are independently created and added to the Linked Open Data (LOD) cloud, on a daily basis. A major challenge for the LOD paradigm is to discover resources that refer to the same real-world object, in order to interlink web resources and hold large scale data integration and sharing. In this context, instance matching is a promising solution. It aims to link co-referent instances belonging to heterogeneous knowledge bases with owl: same as links. Several state-of-the-art existing approaches addressing this issue are based on the prior schema-level matching's, which does not avoid the limitation of heterogeneity at the property-level. In this paper, we propose a schema-free, scalable and efficient instance matching approach that is independent from matching results at the schema-level. We transform the instance matching problem to a document similarity problem and we solve it by a Clustering technique that uses an Ascendant Hierarchical Clustering algorithm to group similar instances in the same clusters. Furthermore, we design multiple validating patterns that use some structural information to validate obtained mappings and eliminate wrong ones. Experiments on instance matching track from Ontology Alignment Evaluation Initiative (OAEI) show that our approach gets prominent results compared to several participating systems in OAEI'2019, OAEI'2020 and OAEI'2021.*

Keywords: *Ontology, LOD, instance matching, ascendant hierarchical clustering, OAEI.*

Received April 9, 2022; accepted April 28, 2022
https://doi.org/10.34028/iajit/19/3A/3

1. Introduction

The ultimate goal of the semantic web is to transform the web of documents into a web of linked data that behaves much like a global searchable database. The web of data is constituted of machine-readable data [5] belonging to different datasets from different sources, interlinked with each other within the Linked Open Data (LOD) project. These datasets are structured and published as Resource Description Framework (RDF) triples. The links between them may represent different types of relationships. One particular type of link, known as owl: *Same as*[1] link, interconnects equivalent instances that refer to the same real-world object, and allows navigating the web of data (by humans and/or by machines) just like a global database, enabling efficient knowledge discovery and data integration.

As more and more datasets are added to the LOD cloud[2] (1301 datasets in May 2021, while there were only 12 at the beginning of the project in 2007), it is important to identify owl: *same as* links between the new datasets and the already existing ones to ensure their appropriate integration in the LOD cloud. To construct such links and answer the needs of LOD paradigm, instance matching, also mentioned as record linkage [21], entity linkage [16], data linkage [29], entity resolution [8, 4], object co-reference resolution [17], reference reconciliation [33], duplicate detection [10] and object identification (in the context of databases) [27], is of crucial importance. This requirement becomes apparent in a multitude of domains ranging from science (marine research, biology, astronomy, pharmacology, etc.,) to semantic publishing and cultural domains.

Instance matching [12, 20, 35] is the process of linking instances that refer to the same real-world object. It matches different descriptions of the same instance from various datasets. For example, Figure 1. Presents two different descriptions of the same person. The aim of instance matching process is to detect that they refer to the same real world object (the Argentinian footballer, Messi) and to link them by owl: same as link.

Several tools using various approaches for instance matching exist in the literature. Among others, we mention: Virtual Document Lexical Similarity (VLDS) [2], Via Multiple Indexes (VMI) [20], ObjectCoref [17], Logical and a Numerical Method for Data Reconciliation (LN2R) [33], etc. For more details about

[1]http://www.w3.org/TR/2004/REC-owl-semantics-20040210/#owl_sameAs/
[2]http://lod-cloud.net

instance matching approaches, readers can refer to [13, 26, 32, 34]. Classical approaches for instance matching depend on the quality of the property matching results. Property matching is the process that links semantically similar properties from various datasets. This process is not trivial in the LOD scenario, since each dataset is independently designed and structured in a specific ontology. For example, the data contained in the property "MariageDate" in one dataset can be contained in several other properties in another dataset, such as (dayOfMariage, monthOfMariage, yearOfMariage). Another example: the data contained in the property "Province" in a first dataset can be contained in the property "Description" in a second dataset. Therefore, both properties contain common data but none of the existing property matching approaches can detect this link. As a result, the common data will be ignored even if it may be worth considering for instance matching process.

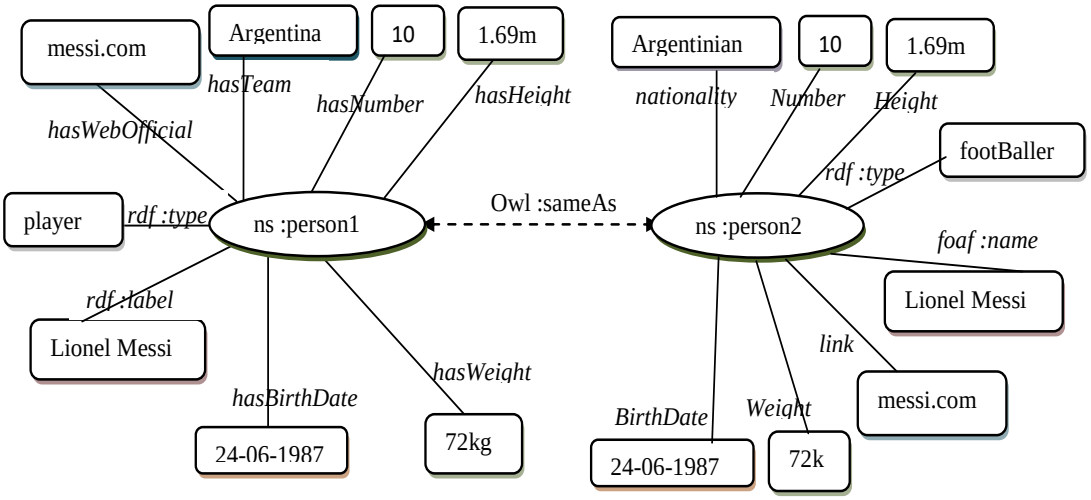

Figure 1. An example of two instances (from different datasets) referring to the same real-world object.

In this paper, we propose a schema-free instance matching approach that is independent from property matching results. To this end, each instance from both ontologies (datasets) is represented by a virtual document structured as a vector of simple and/or compound words extracted from the values of some specified properties of each instance. The order of these words is not important. Two instances from different ontologies can be judged as similar if their virtual documents (vectors) share some significant words. To scale out similar documents (representing similar instances), we employ a clustering approach based on Ascendant Hierarchical Clustering algorithm. Furthermore, we use some structural information to filter-out obtained results and eliminate wrong mappings to improve the efficiency of our approach. We evaluate our approach on datasets from instance matching track of the Ontology Alignment Evaluation Initiative (OAEI) benchmark. Performed experimentations show that our system gets highly competitive results compared to participating systems in OAEI'2019, OAEI'2020 and OAEI'2021.

This paper is an extended version of our previous work in instance matching [1]. In the current version, we add accurate definitions of basic concepts and problem statement. Next, we give a brief analysis of recent and related works from the literature, we extract the main and common steps of their architectures and we position our contribution among these works. Moreover, in the earlier version, we let the Ascendant Hierarchical Clustering algorithm built-up the whole dendrogram (that depicts the hierarchical clustering of documents), then we cut it according to some heuristic rules. In fact, hierarchical clustering of documents is unnecessary in our application, and it can increase time and space complexity needlessly. To overcome this limit, in the current paper, we use one of the most popular approaches to stop clusters' combination in the Hierarchical Clustering algorithm. This approach stops combining clusters when they reach a specified diameter. It reduces significantly the running time as well as the memory space used by our system. After that, we execute our system on a powerful server and we add time performance to the accuracy metrics presented in the earlier version [1]. We also compare our system to the OAEI'2021 participants in addition to OAEI'2019's and OAEI'2020's (previously performed in the earlier version). Furthermore, we add the main benefits and limits of using Ascendant Hierarchical Clustering algorithm to solve instance matching problem.

Our major contributions are summarized as follows:

- Representing each instance from both ontologies as a virtual document in the form of a vector of words from its property values.
- Proposing a schema-free instance matching approach that is independent from property matching results.
- Transforming the instance matching problem to a clustering issue and solving it by Ascendant Hierarchical Clustering algorithm.

- Designing multiple validating patterns that use some structural information to validate obtained mappings and eliminate wrong ones, to enhance the efficiency of the proposed approach.

The remainder of this paper is structured as follows: section two outlines some related works existing in the literature. In the third section, we give some definitions and problem statement. Section four presents an overview of our proposed approach. In section five, we present the main experimentations conducted and their evaluation, essentially the results obtained by our system on the instance matching track from the benchmark OAEI and their comparison with participating systems in OAEI'2019, OAEI'2020 and OAEI'2021. Finally, section six concludes the paper and discusses future work.

2. Related Work

Several approaches for instance matching issue have been proposed in the literature. Most of them share the following steps to scale out the mappings between instances:

1. Define a formal structure of instances (virtual documents, Rdf graphs …).
2. Design a formal structure for the instance matching process.
3. Specify one or more fields (properties) to be used in the matching process.
4. Use a given threshold to scale out most similar instances.

In the following, we present some of these approaches. Then, we position our approach against them to underline its main features. (We have chosen to present the systems that have participated in OAEI'2019, OAEI'2020 and OAEI'2021):

- *AML*: [11, 22] is a property-dependent ontology matching as well as instance matching system. It is based on Agreement Maker Lite [7] and incorporates external knowledge such as biomedical ontologies and wordNet. It uses lexical and structural matching algorithms to construct four matchers: lexical matcher, mediating matcher, word matcher and parametric string matcher. AML employs its own logical repair algorithm to repair mappings and filter-out inconsistent ones.
- *Lily*: [15] is a large scale ontology matching as well as instance matching system. Its matching process contains three main steps: The pre-processing step extracts and prepares necessary information for subsequent steps. The similarity computing step computes similarities between ontology elements and scales-out primary mappings that will be refined in the last step of post-processing. To this end, Lily uses ontology matching debugging strategy to verify and improve the alignment results and ontology matching tuning to enhance overall performance.
- *Log Map*: [18] is a logic-based ontology matching and instance matching system that implements the consistency and locality principles [19]. It supports user intervention during the matching process. Its matching process uses lexical indexation to scale-out primary candidate mappings. LogMap incorporates a logic-based module extraction to modularize input ontologies and reduce the problem size. Furthermore, it uses Horn propositional representation to encode the relevant modules together with a subset of the candidate mappings. Finally, a semantic index and a repair algorithm are used to detect and eliminate unsatisfiable mappings.
- *FTRLim*: [38] is a large scale instance matching system based on the FTRL (Follow The Regularized Leader) model [24]. It generates a set of indexes for instances to scale out the matching candidate pairs and calculates the similarities between them, based on specified attribute(s) and relationships. After that, specific instance pairs are automatically selected as training set (using hyper-parameters from a configuration file) and the FTRL model is trained. Furthermore, previously computed similarities are aggregated into a similarity score with the trained FTRL model. Finally, aligned instances according to the similarity scores and specified threshold are selected.
- *RE-Miner*: [25] is an instance matching system based on Referring Expressions (RE) that uses a given subset of class and property mappings. It starts by discovering REs (RE is a description that identifies an instance unambiguously) for all instances by instantiating the key of classes as well as non-key properties. These keys are obtained by using SAKey [36]. Two instances from different knowledge graphs are likely to be similar if they share some REs. If one instance from the source knowledge graph is linked to more than one instance from the target knowledge graph, a voting strategy is used to choose the most confident link.

From the aforementioned, some of these approaches are designed for specific domains, and most of them are based on schema-level matching (especially, property-level matching). However, our approach is independent of any application domain and it is not based on any schema-level matching. These characteristics have been adopted to make our system applicable to any domain and also to avoid the limitations of schema-level matching techniques.

3. Definitions and Problem Statement

In this section, we state the problem of instance matching process, targeted by this paper after providing the main definitions and basic terminology used in the field:

- Definition 1 (ontology)

Ontology is a formal, explicit specification of a shared conceptualization [14, 31]. Formally, ontology is a six-tuple O={C, P, R_c, R_p, A, I}, where:

C: Set of concepts.
P: Set of properties.
R_c*:* Set of *« is-a »* relationships between concepts.
R_p*:* Set of *« is-a »* relationships between properties.
A: Set of Axioms.
I: Set of instances of concepts (individuals).

The concepts, properties and instances are called ontology entities.

To integrate data from disparate ontologies, we must know the semantic mappings (correspondences) between their elements [3, 37]. This set of mappings is called ontology alignment. and the process that generates them is called ontology matching:

- Definition 2 (Ontology matching)

Ontology matching is the process that scales out the set of semantic mappings between ontology entities. Formally, ontology matching process is defined as a five-tuple *{Id,* E_s*,* E_t*, R, N}*, where:

Id: Unique identifier for the correspondence.
E_s*:* Source entity.
E_t*:* Target entity.
R: The relationship (matching type) between entity pairs *(*E_s*,* E_t*). R* can be of different types (equivalence, subsumption, inverse, overlapping, etc.).
N: A numeric value from [0, 1] describing the confidence value. The higher the confidence value is, the more reliable the matching result is.

The process of matching concepts and properties is called schema-level matching. and the process of matching instances is called instance matching:

- Definition 3 (instance)

An instance describes a real-world object through a set of (property, value) pairs.

Formally, an instance (or even a concept) can be expressed by Rdf triples, called statements, in the form of:<*subject, predicate, object*>. A subject can be a URI or a blank node. an object can be a URI, a blank node or a basic value:

- *URI*: Unified Resource Identifier is the unique identifier of an instance on the web (if two instances from different ontologies have the same URI, we claim directly that they refer to the same real-world object).
- *Blank node*: It represents an anonymous entity.
- *Basic value*: It represents a data type such as string, integer, literal, etc.
- *Predicate* or *property*: it models the relationship between the subject and the object. When the object is a URI, the predicate is called object property. When it is a basic value, it is called data type property. The instances related to an instance by an object property are called its neighbours. The property may be descriptive (such as: Rdfs: comment) or discriminative (such as e-mail property). In addition to URIs, the label of an instance is a highly discriminative property for instance matching. It is the human readable name of an instance that helps to identify the real-world object it corresponds to. Name and label values belong to the property Rdfs*:* label, or to some other common properties like foaf: name or a fragment of its URI [9].
- *Meta information*: of an instance describes its schema information (its class and properties).
- Formally: the triple <*S,* Rdf*:type, C*> indicates that the *URI* "*S"* is an instance of the class "*C"* (*S* can appear as a subject or as an object in another triple).

- Definition 4 (instance matching)

Instance matching aims to identify instances from different ontologies that refer to the same real-world object [35, 20, 12]. It can be seen as the process of building *owl:sameAs* links between co-referent instances. Formally, *owl:sameAs* links built by instance matching process are defined as a four-tuple {*Id,*I_s*,* I_t*, N*} where:

Id: Unique identifier for the correspondence.
$i_s \in I_s$, (source instances).
$i_t \in I_t$ (target instances).
$N \in$ [0, 1] is the confidence value, the higher the confidence value is the more similar the input instances are. The unique matching type in instance matching is the equivalence type.

4. Instance Matching Based on Ascedant Hierarchical Clustering Algorithm

In this section, we provide an overview on how our approach uses the clustering approach to group similar instances together in same clusters. We start by describing the Ascendant Hierarchical Clustering algorithm. After that, we detail the whole steps of the proposed approach.

4.1. Ascending Hierarchical Clustering

Clustering-based algorithms group similar instances (objects) in independent clusters (blocs). Numerous clustering-based algorithms exist in the literature, among which Ascending Hierarchical Clustering, K-means, DBSCAN, etc. For more clustering algorithms, the

reader can refer to [28, 23].

Ascending Hierarchical Clustering (AHC) [6] is one of the oldest and most widely used clustering algorithms. It uses a bottom-up strategy. Firstly, each instance forms a singleton cluster. After that, the matrix similarity between all instances is computed. Next, it iteratively, merges the closest two clusters according to some similarity measure until all instances are grouped in a single cluster. The obtained hierarchical grouping of clusters is called dendrogram (see Figure 2). It can be cut according to some specific heuristic rules.

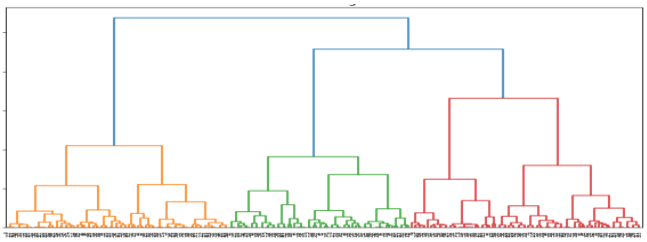

Figure 2. The dendrogram representing the hierarchical grouping of instances.

Ascendant Hierarchical Clustering Algorithm

1. Compute the matrix of similarity of each instance pair.
2. Repeat
3. Merge the closest two clusters
4. Update the similarity matrix to reflect the proximity between the new cluster and the original clusters
5. Until only one cluster remains

Figure 3. Pseudo code of the ascending hierarchical clustering algorithm.

The pseudo code of the Ascending Hierarchical Clustering algorithm is depicted in Figure 3.

4.2. Approach Overview

Our approach for instance matching implemented in python is illustrated by Figure 4. It is composed of four main steps: Pre-processing, virtual document aggregation, ascendant hierarchical clustering and filtering.

1. Pre-Processing: after loading source ontologies and extracting their instance data, the pre-processing step consists of representing each instance of both ontologies by a virtual document which has the form of a vector containing a set of simple and/or compound words extracted from the property values of each instance. In fact, we do not use all the information furnished by all the property values of instances. For instance, noisy information (such as random strings) is eliminated. Furthermore, the most discriminative properties are specified to be used in the construction of virtual documents, because discriminative properties are useful for instance matching process. For example, to identify real world objects, humans look first at their names (the labels of instances). In our system, we follow the same mechanism. To specify discriminative properties, we first look for the names or labels of instances. They may be represented via special properties such as Rdfs:label, foaf:name, the suffix of its URI or even, it may be extracted from the Description field of the instance. If textual fields are not specified in the datasets or they contain noisy information (random strings), we look for other types of discriminative properties which are very suitable for instance matching process, such as dates, numbers or links. Most discriminative properties and noisy information to be eliminated are specified by users in a configuration file. For example, basing on the specified discriminative properties, the instances depicted in Figure 1. will be represented by the virtual documents presented in Figure 5.

The output of pre-processing step is the sets of virtual documents of each instance from both source and target ontologies.

2. Virtual Documents: aggregation: to prepare input instances (objects) for the clustering algorithm AHC, all virtual documents representing all instances from both source and target ontologies are aggregated to a single dataset.
3. Ascendant Hierarchical Clustering: this is the core step of our system because it gives the main part of the mappings between input instances. The input of this module is the set of virtual documents representing all instances from both source and target ontologies. Initially, there are as many clusters as the number of virtual documents (instances) and each singleton cluster is initialized by only one virtual document. Next, the matrix of similarity between all instances is computed. Herein, the jaccard similarity is used to measure similarity between virtual documents. The normal execution of the next steps of the Ascendant Hierarchical Clustering algorithm consists of repeatedly merging the two closest clusters in a single one (see figure 3) and the resulting hierarchical grouping of clusters will be given through the dendrogram (such as the one depicted in figure 2). However, in our application, there is no need for the hierarchical grouping of clusters. In addition, it can uselessly increase the runtime and memory space complexity. To deal with this issue, we use an appropriate approach to stop combining the clusters in the Hierarchical Clustering algorithm. Four popular approaches based on different criteria for stopping the combination of clusters in the Hierarchical Clustering algorithm exist in the literature[4]:

    1. Pick several clusters upfront.
    2. Stop clusters combination when the resulting cluster has low cohesion.
    3. Stop clusters combination when the resulting cluster reaches a specified radius.

4. Stop clusters combination when the resulting cluster reaches a specified diametere.

   We chose to use the fourth approach, because it is the most suitable and applicable one for our application.

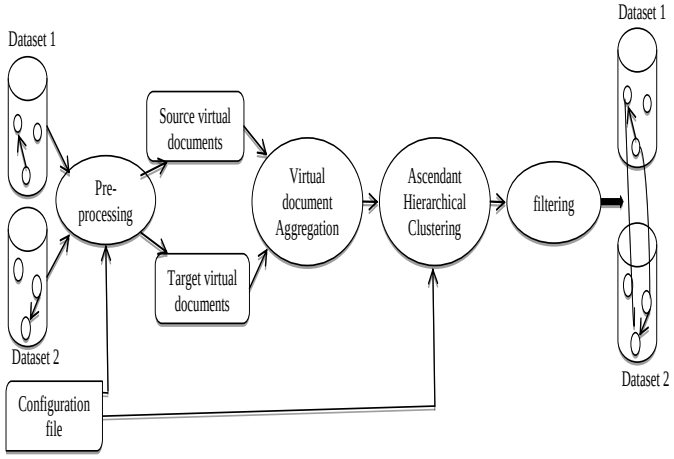

Figure 4. An overview of the proposed instance matching approach.

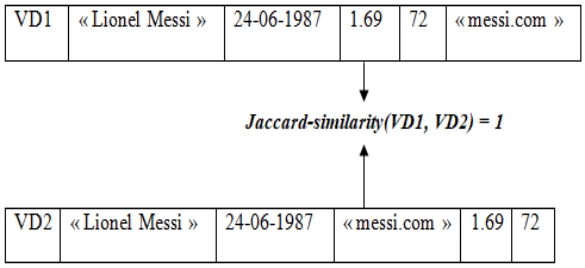

Figure 5. An example of virtual documents representing similar instances.

By applying this approach, we stop combining the clusters when achieving a specified diameter of clusters. The diameter of a cluster is defined as the maximum distance between any pair of instances in the cluster. The permitted diameter of clusters is specified in the configuration file. By applying this approach, we can avoid unnecessary hierarchical groupings and thus significantly reduce the running time as well as the memory space complexity. Finally, each obtained cluster of instances is interpreted as a mapping between those instances and the set of obtained mappings will be filtered in the next step.

4. Filtering: this step consists of filtering obtained mappings to eliminate wrong ones and improve the efficiency of our system. To do that, we design three validating patterns that use some structural information to filter-out obtained mappings and eliminate wrong ones, which enhance the efficiency of the proposed approach:

   a) If both instances of the same cluster are from the same source dataset then the resulting mapping will be eliminated.

   b) If both instances of the same cluster are from different datasets but belonging to disjoint classes then the resulting mapping will be eliminated.

   c) If one instance from the first dataset has more than one similar instance from the other dataset then these mappings (redundant mappings) are eliminated. This case can occur when source ontology contains redundant instances (redundant error in ontology evaluation).

5. Experimental Study

In this section, we evaluate the efficiency of our approach. In the following, we describe the benchmark and the datasets used for the experiments, the obtained results as well as their comparison with state of the art systems:

1. Benchmark and Data sets

To improve the efficiency of our system and to enable the comparison of obtained results with those of other approaches, we have conducted experiments on standard and recent benchmarks, OAEI'2019, OAEI'2020 and OAEI'2021. we have used datasets from Semantic Publishing Instance Matching BENCHmark (SPIMBENCH) ontologies of the instance matching track:

- OAEI is an international coordinated initiative that yearly organizes the evaluation of increasing number of ontology matching systems. Since 2004, OAEI aims to compare ontology matching systems on different datasets from different tracks. With the emergence of instance matching technologies, OAEI integrated instance matching track, since 2009.

The goal of SPIMBENCH task from instance matching track is to determine when two owl instances describe the same creative work. The datasets are generated and transformed using SPIMBENCH by altering a set of original data through value-based, structure-based and semantics aware transformations. For more details about SPIMBENCH ontologies, the reader can refer to [30].

2. Experimentations and Results

In our experimentations, we have used jaccard similarity to measure similarity between instances (virtual documents) and Min distance to measure similarity between clusters (groups of instances). The clusters diameter is equal to 0 (instance similarity is equal to 1). We have excuted our system on a 32 GigaByte (GB) Random Access Memory (RAM) and 2.3GigaHertZ (GHZ) Central Processing Unit (CPU) server. In addition to measuring the time performance, we have computed the standard evaluation measures (precision, recall and F-measure) against the reference alignments given by the campaign OAEI. These measures are defined by the following Equations:

$$precision = \frac{TP}{TP + FP} \quad (1)$$

$$recall = \frac{TP}{TP + FN} \quad (2)$$

$$F - measure = \frac{2 * precision * recall}{precision + recall} \quad (3)$$

Where, *TP* is the number of correct mappings. *FP* the number of wrong mappings and *FN* the number of missed mappings.

The reference alignment given by OAEI campaign contains 299 correct mappings. Our system detects 300 mappings with *TP=298* (correct), *FP=2* (wrong) and *FN=1* (missed). From where: *precision =0.9933, recall=0.9967 and F-measure=0.9949.*

By analysing wrong mappings, we found that the wrong aligned instances have exactly the same property values but they do not exist in the reference alignment because they are redundant instances in the target ontology (Abox2). And the 1 missed mapping cannot be detected by our system, because one of its matched instances was detected as similar to other instance in the same dataset (Abox2). This wrong mapping was filtered by the first filtering pattern of our system. Table 1 compares the results obtained by our system against those[3] of the systems participating in OAEI'2019. A graphical comparison with these systems is given by Figure 6:

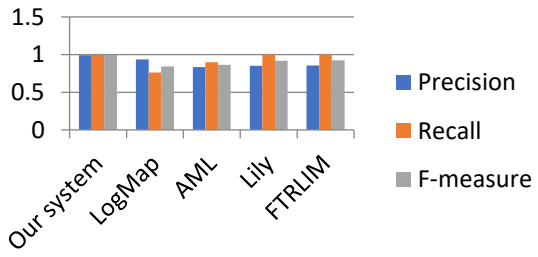

Figure 6. Graphical comparison of obtained results with OAEI'2019 participants.

Table 1. Obtained results on spimbench ontoldgies compared with oaei'2019 participant.

System	Precision	Recall	F-measure	Time performance
Our system	0,9933	0,9967	0,9949	4216
LogMap	0,9383	0,7625	0,8413	6919
AML	0,8349	0,8963	0,8645	6223
Lily	0,8494	1	0,9186	2032
FTRLIM	0,8543	1	0,9214	1474

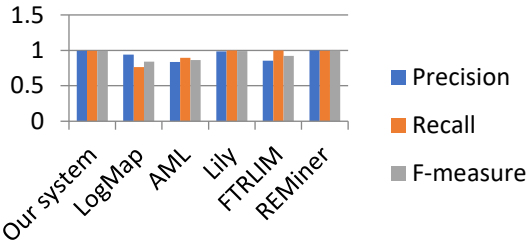

Figure 7. Graphical comparison of obtained results with OAEI'2020 participants.

Table 2 compares the results obtained by our system against those [4] of the systems participating in OAEI'2020. A graphical comparison with these systems is given by Figure 7.

Table 2. Obtained results on spimbench ontologies compared with OAEI'2020 participants.

System	Precision	Recall	F-measure	Time performance
Our system	0,9933	0,9967	0,9949	4216
LogMap	0,9382	0,7625	0,8413	7483
AML	0,8348	0,8963	0,8645	6446
Lily	0,9835	1	0,9917	2050
FTRLIM	0,8542	1	0,9214	1525
REMiner	1	0,9966	0,9983	7284

Table 3 compares the results obtained by our system against those [5] of the systems participating in OAEI'2021. A graphical comparison with these systems is given by Figure 8.

Table 3. Obtained results on spimbench ontologies compared with oaei'2021 participants.

System	Precision	Recall	F-measure	Time performance
Our system	0,9933	0,9967	0,9949	4216
LogMap	0,9382	0,7625	0,8413	5699
AML	0,8348	0,8963	0,8645	7966
Lily	0,9835	1	0,9917	1845

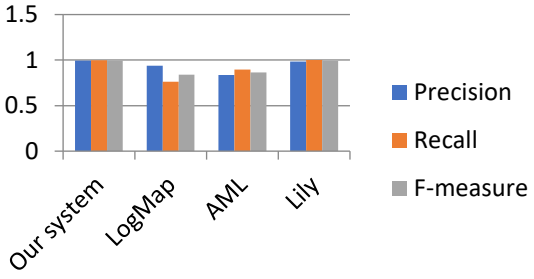

Figure 8. Graphical comparison of obtained results with OAEI'2021 participants.

From Tables 1, 2, 3 we state that our system obtained very encouraging and comparable results with participating systems. It outperforms OAEI'2019 and OAEI'2021 participants in F-measure score and gives results very close to the best performer, RE-Miner in OAEI'2020. The runtime performance of our system is acceptable and comparable with participating systems.

[3]https://project-hobbit.eu/challenges/om2019/
[4]https://hobbit-project.github.io/OAEI_2020.html
[5]https://hobbit-project.github.io/OAEI_2021.html

6. Conclusions

In this paper, we proposed a property-independent instance matching approach that avoids the problem of heterogeneity at the property level. We presented a new method for building virtual documents corresponding to instances. We also transformed the instance matching problem into a document similarity problem and we solved it by Ascendant Hierarchical Clustering algorithm. The main advantages of using Ascendant Hierarchical Clustering algorithm is that the clusters containing matched instances can be easily obtained from the model itself, automatically. In addition, the resulting dendrogram gives us a clear visualization which is practical and easy to understand. However, hierarchical clustering of clusters is unnecessary in our application, and it can increase time and space complexity needlessly, (we only need the first iteration of clustering that groups each pair of similar instances). To better guide the clustering process, we used a popular approach to stop cluster combination by achieving a specified cluster diameter. This approach significantly reduces the running time as well as the memory space complexity, compared with sheer ascendant hierarchical clustering. Furthermore, we designed multiple validation patterns that use some structural information to filter-out obtained results and eliminate wrong mappings. We compared our approach to state-of-the-art systems on benchmark datasets, and we achieved very promising results.

As a future work, it will be interesting to use contextual information, such as neighboring instances (i.e., instances with similar neighbors are similar). This will detect more correct mappings and improve the F-Measure score. It will also be interesting to propose an extension for the validating patterns that may eliminate more false positives and enhance the quality of the results.

References

[1] Amrouch S. and Mostefai S.,"Ascendant Hierarchical Clustering for Instance Matching," *in proceeding of the 22nd International Arab Conference on Information Technology*, Oman, pp. 1-6, 2021.

[2] Assi A., Mcheick H., Karawash A., and Dhifli W., "Context-aware Instance Matching Through Graph Embedding in Lexical Semantic Space," *Knowledge-Based Systems*, vol. 186, p. 422-433, 2019.

[3] Berners-Lee T., Hendler J. and Lassila O.,"The Semantic Web," *Scientific American*, vol. 284, no. 5, pp. 34-43, 2001.

[4] Bhattacharya I. and Getoor L., *Mining Graph Data*, Wiley and Sons, 2006.

[5] Bizer C., Heath T., and Berners-Lee T., "Linked Data-The Story So Far," *Semantic Web and Information Systems*, vol. 5, no. 3, pp. 1-22, 2009.

[6] Bruynooghe M., *Large Data Set Clustering Methods Using the Concept of Space Contraction*, Physika Verlag, 1978.

[7] Cruz I., Antonelli F., and Stroe, C., "AgreementMaker: Efficient Matching for Large Real-World Schemas and Ontologies," *Journal of VLDB*, vol. 2, no. 2, pp. 1586-1589, 2009.

[8] Efthymiou V., Papadakis G., Stefanidis K., and Christophides V., "Minoaner: Schema-Agnostic, Non-Iterative, Massively Parallel Resolution of Web Entities," *in Proceeding of the 22nd International Conference on Extending Database Technology*, lisbon, pp. 373-384, 2019.

[9] Ell B., Vrandecic D., and Simperl E., "Labels in the Web of Data," *in Proceeding of the 10th International Semantic Web Conference*, Bonn, pp. 162-176, 2011.

[10] Elmagarmid A., Ipeirotis P., and Verykios V., "Duplicate Record Detection: A Survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 1, pp. 1-16, 2007.

[11] Faria D., Pesquita C., Santos E., Palmonari M., Cruz I., and Couto F., "The AgreementMakerLight Ontology Matching System", *in Proceeding of the On the Move to Meaningful Internet System*, pp. 527-541, 2013.

[12] Ferrara A., Nikolo A., Noessner J., and Scharffe F., "Evaluation of Instance Matching Tools: The Experience of Oaei," *Journal of Web Semantics*, vol. 21, pp. 49-60, 2013.

[13] Ferrara A., Nikolov A., and Scharffe F.,"Data Linking for the Semantic Web," I*nternational journal on Semantic Web and Information systems*, vol. 7, no. 3, pp. 46-76, 2011.

[14] Gruber T., "A Translation Approach to Portable Ontology Specifications," *Knowledge Acquisition*, vol. 5, no. 2, pp. 199-220, 1993.

[15] Hu Y., Bai S., Zou S., and Wang P., "Lily Results for OAEI 2020," *in Proceeding of the 15th International Semantic Web Conference*, Athens, pp. 194-200, 2020.

[16] Hu W . and Jia C., "Bootstrapping Approach to Entity Linkage on the Semantic Web," *Journal of Web Semantics*, vol. 34, pp. 1-12, 2015.

[17] Hu W., Chen J., and Qu Y., "Self-Training Approach for Resolving Object Coreference on The Semantic Web," *in Proceeding of the 20th International Conference on World Wide Web*, Hyderabad, pp. 87-96, 2011.

[18] Jimenez-Ruiz E., "LogMap Family Participation in the OAEI 2020," *in Proceeding of the 15th International Semantic Web Conference, Workshop on Ontology Matching*, Athens, pp. 201-203, 2020.

[19] Jimenez-Ruiz E., Grau B., Horrocks I., and Berlanga R., "Logic-based Assessment of the Compatibility of UMLS Ontology Sources,"

Journal of Biomedical Semantics, vol. 2, no. 1, pp. 1-16, 2011.

[20] Li J., Wang z., Zhang x., and Tang j., "Large Scale Instance Matching Via Multiple Indexes and Candidate Selection," *Knowledge-Based Systems*, vol. 50, pp. 112-120, 2013.

[21] Li C., Jin L., and Mehrotra S., "Supporting Efficient Record Linkage For Large Data Sets Using Mapping Techniques," *World Wide Web*, vol. 9, no. 4, pp. 557-584, 2006.

[22] Lima B., Faria D., Couto F., Cruz I., and Pesquita C., "Results for OAEI 2020 AML and AMLC," *in Procceding of the 15th International Semantic Web Conference*, Athens, pp. 154-160, 2020.

[23] Madhulatha T., "An Overview on Clustering Methods, IOSR," *Journal of Engineering*, vol. 2, no. 4, pp. 719-725, 2012.

[24] McMahan H., Holt G., Sculley D., Young M., Ebner D., Grady J., Nie L., Phillips T., Davydov E., Golovin D., Chikkerur S., Liu D., Wattenberg M., Hrafnkelsson A., Boulos T., and Kubica J., "Ad Click Prediction: A View From the Trenches," *in Proceeding of the 19th International Conference on Knowledge Discovery and Data Mining*, Chicago, pp. 1222-1230, 2013.

[25] Nassiri A., Pernelle N., Saïs F., and Quercini G., "RE-miner for Data Linking Results For OAEI 2020," *in Proceeding of the 15th International Semantic Web Conference, workshop on Ontology Matching*, Athens, pp. 211-215, 2020.

[26] Nentwig M ., Hartung M., Ngomo A., and Rahm E., "A Survey Of Current Link Discovery Frameworks," *Journal of Semantic Web*, vol. 8, no. 3, pp. 419-436, 2017.

[27] Noessner J., Niepert M., Meilicke C., and Stuckenschmidt H., "Leveraging Terminological Structure for Object Reconciliation," *in Proceeding of the 7th Extended Semantic Web Conference*, Heraklion, pp. 334-348, 2010.

[28] Omran M., Engelbrecht A., and Salman, A., "An Overview of Clustering Methods," *Intelligent Data Analysis*, vol. 11, no. 6, pp. 583-605, 2007.

[29] Pernelle N., Saïs F., and Symeonidou D., "An Automatic Key Discovery Approach for Data Linking," *Journal of Web Semantics*, vol. 23, pp. 16-30, 2013.

[30] Pour M., Algergawy A., Amini R., Faria D., Fundulaki I., Harrow I., Hertling S., Jimenez-Ruiz E., Jonquet C., Karam N., Khiat A., Laadhar A., Lambrix P., Li H., Li Y., Hitzler P., Paulheim H., Pesquita C., Saveta T., Shvaiko P., Splendiani A., Thieblin E., Trojahn C., VatascinovA J., Yaman B., Zamazal O., and Zhou L., "Results of the Ontology Alignment Evaluation Initiative 2020," *in Proceeding of the 15th International Workshop on Ontology Matching*, Athens, PP. 42-138, 2020.

[31] Pulido J., Ruiz M., Herrera R., Cabello C., Legrand S., and Elliman D., "Ontology Languages For The Semantic Web: A Never Completely Updated Review." *Knowledge-Based Systems*, vol. 19, no. 7, pp. 489-497, 2006.

[32] Raimond Y., Sutton C., and Sandler M., "Automatic Interlinking of Music Datasets on the Semantic Web," *in Proceeding of the 1st Workshop about Linked Data on the Web*, Beijing, 2008.

[33] Saïs F., Pernelle N., and Rousset M-C., "Combining A Logical and A Numerical Method for Data Reconciliation," *Data Semantics XII*, vol. 12, no. 12, pp. 66-94, 2009.

[34] Sleeman J. and Finin T., "Computing Foaf Co-Reference Relations With Rules And Machine Learning," *in proceeding of the 3rd International Workshop on Social Data on the Web*, China, 2010.

[35] Suchanek F., Abiteboul S., and Senellart P., "Paris: Probabilistic Alignment Of Relations, Instances, And Schema," *Proceedings of the VLDB Endowment*, vol. 5, no. 3, pp. 157-168, 2011.

[36] Symeonidou D., Armant V., Pernelle N., and Saïs F., "Sakey: Scalable Almost Key Discovery In Rdf Data," *in Procceding of the 13th International Semantic Web Conference*, Riva del Garda, pp. 33-49, 2014.

[37] Uschold M., "Where Is the Semantics in the Semantic Web?," *AI Magazine*, vol. 24, no. 3, pp 25-36, 2003.

[38] Wang X., Jiang Y., Fan H., Zhu H., and Liu Q., "FTRLIM results for OAEI 2020", *in Proceeding of the 15th International Semantic Web Conference, Workshop on Ontology Matching*, Athens, pp. 187-193, 2020.

Siham Amrouch graduated from Badji Mokhtar University, Annaba, Algeria, as a state engineer in Computer Science. She received a Magistere degree in Artificial Intelligence, then, she obtained her PhD in the field of ontology matching and merging from the same university. Currently, she is Associate professor, and full researcher at LIM Laboratory in Souk Ahras University, Algeria. Her research areas comprise: ontology matching and merging, knowledge management, semantic web technology and Arabic handwriting recognition.

Sihem Mostefai graduated from Mentouri University, Constantine, Algeria as a state engineer in Computer Science. She obtained a Magistere degree in computer graphics, then she received a PhD in the field of Information integration applied to PLM (Product Lifecycle Management) from the same university. She is presently Associate professor, and full researcher at MISC Laboratory in Constantine 2 University, Algeria. Her research interests include: semantic web technology, ontology engineering, information retrieval and network security.