Received: 29 April 2021 | Accepted: 11 June 2021

DOI: 10.1111/coin.12474

SPECIAL ISSUE ARTICLE

Computational Intelligence WILEY

An ontology matching approach for semantic modeling: A case study in smart cities

Youcef Djenouri[1] | **Hiba Belhadi**[2] | **Karima Akli-Astouati**[2] | **Alberto Cano**[3] | **Jerry Chun-Wei Lin**[4]

[1]Department of Mathematics and Cybernetics, SINTEF Digital, Oslo, Norway

[2]Department of Computer Science, USTHB, Algiers, Algeria

[3]Department of Computer Science, Virginia Commonwealth University, Richmond, Virginia, USA

[4]Department of Computer Science, Electrical Engineering and Mathematical Sciences, Western Norway University of Applied Sciences, Bergen, Norway

Correspondence
Jerry Chun-Wei Lin, Department of Computer Science, Electrical Engineering and Mathematical Sciences, Western Norway University of Applied Sciences, Bergen, Norway.
Email: jerrylin@ieee.org

Abstract

This paper investigates the semantic modeling of smart cities and proposes two ontology matching frameworks, called Clustering for Ontology Matching-based Instances (COMI) and Pattern mining for Ontology Matching-based Instances (POMI). The goal is to discover the relevant knowledge by investigating the correlations among smart city data based on clustering and pattern mining approaches. The COMI method first groups the highly correlated ontologies of smart-city data into similar clusters using the generic k-means algorithm. The key idea of this method is that it clusters the instances of each ontology and then matches two ontologies by matching their clusters and the corresponding instances within the clusters. The POMI method studies the correlations among the data properties and selects the most relevant properties for the ontology matching process. To demonstrate the usefulness and accuracy of the COMI and POMI frameworks, several experiments on the DBpedia, Ontology Alignment Evaluation Initiative, and NOAA ontology databases were conducted. The results show that COMI and POMI outperform the state-of-the-art ontology matching models regarding computational cost without losing the quality during the matching process. Furthermore, these results confirm the ability of COMI and POMI to deal

with heterogeneous large-scale data in smart-city environments.

KEYWORDS

clustering, ontology Matching, pattern mining, semantic modeling, smart city

1 | INTRODUCTION

Today's World-Wide Web has billions of web pages, but the vast majority of them is readable by human (in HTML format). As the result, machines cannot understand and process this information, so much of the web's potential goes untapped. To do this, the researchers created the Semantic Web, where ontologies describe the semantics of data. When data is in the form of ontologies, machines can better understand semantics and therefore locate and integrate data for a wide variety of tasks. On the Semantic Web, data comes from many different ontologies, and processing information through ontologies is not possible without knowing the semantic links between them. Ontology matching is the process of finding the mappings between two ontologies represented in different domains. It can be applied to several real-world problems, such as biomedical data,[1] e-learning,[2] and Natural Language Processing.[3] Cities are rapidly growing as they strive to accommodate more than 2.5 billion smart citizens by 2050. Understanding city dynamics is crucial to harmonizing internal conflicting demands in housing, business, leisure, mobility, energy, or ecology, as well as managing external shocks. Heterogeneous data in smart cities is rapidly growing in volume and types, which makes ontology matching play an important role in smart-city semantic modeling to improve city planning knowledge.

1.1 | Motivation

Trivial methods for ontology comparison analyze the ontology instances by considering all the characteristics of both ontologies. Thus, it takes the number of $n \times n' \times m \times m'$ comparisons to find the alignment, where n and n' are defined as the numbers of instances, and m and m' correspond to the numbers of the data properties of the first ontology and the second ontology, respectively. Ontology matching is a polynomial problem since many instances and properties are required to be considered for high-accuracy matching. For instance, if we consider a large-scale dataset, such as DBpedia[1] with 4,233,000 instances and 2795 different properties, 144×10^{18} comparisons are needed. This results in a very time-consuming matching process. The DBpedia ontology and its number of properties are shown in Figure 1 to support this declaration of the computational complexity using well-known ontology matching algorithms: Extended Inverse Functional Property Suite (EIFPS)[4] that is a semi-supervised learning approach. Shao et al.[5] then introduced an iterative matching framework using a blocking technique to minimise the number of comparison. For data properties of less than 10%, the runtime of both models was less than 20 s, the results are obtained with an Intel *i7* processor and 16 GB of main memory. However, these approaches have runtimes greater than 700 s, with data properties equal

[1]http://wiki.dbpedia.org/Datasets

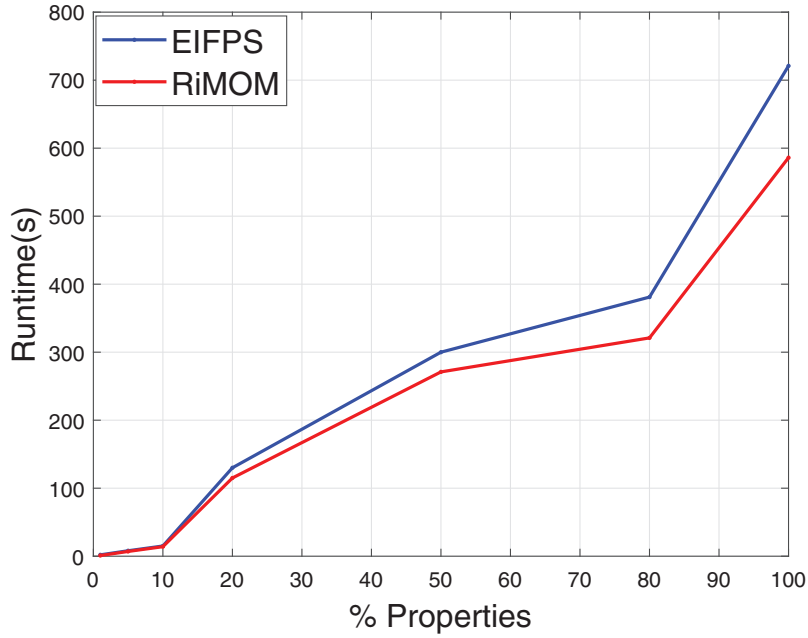

FIGURE 1 Runtime of extended inverse functional property suite and RiMOM using the DBpedia varying from 1% to 100% the data properties

to 100%. More sophisticated solutions to ontology problems attempt to improve the matching process by exploring the search space with the partitioning algorithms,[6-10] high-performance computing (HPC),[11-13] and evolutionary computation approaches,[14-17] among others. However, the overall performance of the ontology matching still needs improvements in particular for complex applications such as related to smart cities. Data mining aims at discovering the relevant information, knowledge, and/or hidden patterns from large and big databases. Clustering[18] and pattern mining[19-21] are well-known data mining tasks that are aimed at partitioning the whole data into similar groups to study the correlation among the different data features. Clustering and pattern mining have been also applied to ontologies[6-8] by considering description logic to decompose ontology database into several modules that can be used to study the relationships between the relevant concepts of the given ontologies. However, the above approaches cannot be straightforwardly applied to the matching problem among different ontologies since they cannot extract the smallest modules from complex ontologies. Moreover, a higher computational cost is required when the data is huge. Motivated by the success of clustering and pattern mining in solving several complicated problems, such as information retrieval,[22] traffic transportation,[23] and business intelligence,[24] this paper presents a data-driven approach and outlines how these powerful data mining techniques can be explored to solve the problem of ontology matching.

1.2 | Contributions

To our best of our knowledge, this is the first study that explores the methods of clustering and pattern mining to solve the ontology matching problem. Furthermore, a case study on smart-city

semantic modeling is shown to demonstrate an application of this work. The main contributions can be summarized as follows:

1. We present a new framework, called Clustering for Ontology Matching-based Instances (COMI), which adopts clustering techniques to decompose the set of instances of the given ontologies. The framework can group the most relevant features into a cluster, which can be used to greatly improve the matching problem of different ontologies. To speed up the computation of the ontology matching, an improved k-means algorithm[25] is proposed to deal with clustering of the instances within the ontologies.
2. We present a new framework, called Pattern mining for Ontology Matching-based Instances (POMI), which adopts the pattern mining techniques to study the different correlations among the data properties. The designed framework obtains the most relevant features by exploiting frequent pattern mining on both ontologies. To speed up the computation of the whole ontology matching process, an improved SSFIM algorithm,[26] with an efficient pruning strategy is proposed to deal with pattern mining-based instances within the ontologies.
3. Extensive experiments were carried out to demonstrate the usefulness of the proposed frameworks COMI and POMI. The results reveal that both COMI and POMI outperformed the state-of-the-art ontology matching algorithms in terms of runtime while obtaining high-quality solutions.
4. A case study on smart-city semantic modeling is shown to demonstrate the validity of COMI and POMI in dealing with big and heterogeneous data in smart-city environments.

1.3 | Outline

The rest of this paper is structured as follows. Section 2 discusses related works in the ontology matching problem. Section 3.1 gives the formal definition used in the ontology matching problem. Section 3 presents the COMI framework whereas Section 4 introduces the POMI framework. A performance evaluation of the COMI and POMI frameworks is provided in Section 5. Finally, Section 6 draws the conclusions and future work in the ontology matching problem.

2 | RELATED WORK

Several approaches have been introduced in the last decade to solve the ontology matching problem.[14-16,27,28] Matching strategies based on instances are also appropriate for connecting database records.[29,30] Much research has explored methods for improving the efficiency of ontology matching. Solutions regarding the ontology matching problem can be categorized into two groups: (i) solutions based on the reduction of the search space by employing computational intelligence, data mining, and machine learning methods;[6-8] and (ii) solutions based on HPC while parallel matching is established.[11-13] This work focuses on the solutions based on the reduction of a search space and approaches in this category are overviewed in the following section.

2.1 | Traditional techniques

An instance matching approach, named VMI, was developed by Wang et al.[31] For each instance, it builds two distinct vectors, such as the vector name and the virtual document vector. The VMI

method reduces the number of similarity measurements by using multiple indexing and candidate selection and operates effectively only in large cases with a limited number of data properties. The best results are obtained when users specify all the corresponding data properties and methods of retrieving the values. Thus, their approach is based on a generic instance matching algorithm, whereas some processes are applied to particular domains; that is to say, a simple string comparison of names and data characteristics is utilized for obtaining comprehensive instance information. In the 2009 OAEI competition for small ontology datasets, VMI obtained successful matching. However, with increasing instances, its quality decreases. Li et al.[32] developed an approach that is based on the hypothesis that, two entities of the same real-world object may be matched when they are related to previously matched entities. This technique incorporates multiple lexical matches using a new voting aggregation process and only uses the structural information and the correspondences observed to locate the additional information, which can then primarily be broken down into two stages:

1. Identification of highly accurate seminal correspondences by lexical information.
2. The derivation of additional matching outcomes based on the semantic matching of the previous stage with a structural matching strategy.

Based on the findings of the 2010 OAEI study, this method obtains a reasonable accuracy for certain medium and small ontology databases. Hu et al.[5] presented RiMOM at the OAEI competitions in 2013 and 2016. It introduces an iterative matching framework in which the distinctive information is centered on a blocking technique for minimizing the number of pairs of candidates. As a key to the index of the instances, it uses predicates, and its distinctive object. Moreover, a weighted, exponential similarity averaging method is used to ensure that the instance matching fits with the high precision. The new blocking approach decreases the computational cost significantly without losing precision and recall. RiMOM achieves 99% accuracy in small and medium ontology datasets. Alam et al.[33] developed an expansion of MERGILO, a method to reconcile knowledge graphs extracted from the text by graph alignment and word similarity. Compared with the generic approaches, the results of the extended MERGILO show significant improvement. Rosaci[34] found that ontology matching can be used to link various smart agents. The ontology of an agent simulates the actions of an agent, and, if an agent proposes, then any agent in the group will know the relation between itself and another agent. Rosaci[35] then used the hierarchical model to identify semantic associations between web data. The semantic connections represented by metadata are discussed in the context of a collection of network entities. The usefulness of this approach has been demonstrated in well-known web user recommendation systems. The interlinking issue was first addressed as problems of duplication or record linkage by the database community, where Elmagarmid et al.[36] based their research on several methods to tackle the problems of heterogeneity in ontology matching and proposed a method of handling a set in organized property-segmented documents.

2.2 | Data mining-driven solutions

Linked open data (LOD) is data that is structured and interconnected with each other, so that it becomes more useful by semantic queries. To address the matching problem in LOD by using rules taken from the association rule mining technique, Niu et al.[4] developed the EIFPS technique, which is considered to be a semi-supervised learning approach. A limited number of current

matches *owl:sameAs* are used as seeds and the related rules as criteria for optimizing precision are considered. The authors presented a graphic metric that measures the likelihood and law of Dempster while integrating confidence values. The theory makes it possible to combine instances from different datasets and to arrive at a degree of belief that takes into account all the available instances. The degrees of belief may or may not have the mathematical properties of probabilities; they differ depends on the degree of correlation between the two data sets. Then, by presenting the power of resource homogeneity for the e-learning context, Sergio et al.[2] presented the LOM framework. To expand and improve the available tools for online learning semantically, the use of the initial associative classifier for ontology matching was then developed and investigated. This model uses a feature-based similarity function that needs historical knowledge as the training set. This method was evaluated and verified at the 2014 OAEI ontology database competition. The results for several larger ontology databases showed 90% precision. Ochieng et al.[37] presented an approach that splits an ontological graph into many partitions. Cluster-based similarity aggregation (CSA)[38] is a system integrating varied factors (i.e., five measures, a string-similarity calculation, and a WordNet-based similarity measure) to derive the alignment of ontology concepts. Algergawy et al.[39] then proposed a large-scale ontology matching clustering approach. The main concept is to divide the schema graph by using context-driven structural node similarities into clusters. The Vector Space Model is also defined after the partitioning of each ontology to discover similar clusters and generate the same concepts. In the context of smart-city semantic modeling, several ontology matching based solutions have been proposed. Bellini et al.[40] introduced a system for the management of large-volume data from a range of sources that considers both static and dynamic data in smart cities. Qui et al.[41] developed a semantic graph-based method by incorporating semantic graph structure information and context information that can be used to identify the nontaxonomic relationships in smart-city environments. A unified consolidated and live view for heterogeneous city data sources was given by Le et al.[42] It addresses billions of historical and current records together to accumulate and enrich millions of triples for linking to a graph in real-time per hour. Qui et al.[43] proposed a graph method for semanticizing knowledge accurately from heterogeneous information on smart cities. Smart-city data are first computed with the word co-occurrence as a result of similarities. A semantic graph is then constructed based on the similarities between the smart-city data. A community detection algorithm is finally used to divide the smart-city data into different communities where each community acts as a concept.

2.3 | Tools

Several works regarding review and analytics have been studied and analyzed for finding the ontology matching solutions that are discussed and studied here. Through analyzing the state-of-the-art matching issues, Shaviako et al.[27] evaluated the matching problem solutions. Assessments and application analyses were provided using the competitive OAEIontology databases competition [2]. Abubakar et al.[29] studied the current ontological situation rather than popular conceptual matching with specific considerations of ontological instance-based matching. To estimate relative effectiveness and performance, Nentwig et al.[30] then investigated the comparative evaluations of link discovery (DL) frameworks. Mohammadi et al .[44] presented statistical methods to compare two or more alignment systems in terms of efficiency.

[2]http://oaei.ontologymatching.org

The statistical procedures were then discussed[45] to show comparisons between the two alignment systems. First, the database community considered interconnections as problems with duplication or record linkage. Elmagarmid et al.[36] aimed various techniques at resolving the heterogeneity issues of ontology matching and proposed the solution of a series of structured property segmented record data. The classification of the ontology-based models was also incorporated into the methods of character-based similarity metrics, phonetic similarity metrics, token-based similitude metrics, and numeric similarity metrics. There are certain detection methods for duplicated records, and duplicated detection tools have been developed. Otero et al.[28] addressed a variety of approaches and their functional applications in real-life, involving more than 50 ontology matching systems. Heflin et al.[46] gave an overview of the ontology relationships of ontology instances. They also summed up some matching instance algorithms, such as the scalable entity co-reference systems and manual and automated blocking key selection. They also introduced the generic algorithms that use logical reasoning based on string matching. Moreover, two extensive evaluations were made of the ontology matching systems: (1) ASMOV,[47] N2R,[48] RiMOM,[49] CODI,[50] PARIS,[51] EPWNG,[52] SiGma,[53] and MA[54] were evaluated and verified on the OAEI (Person1, Person2, and Restaurant) benchmark; and (2) EdJoin,[55] DisNGram,[56] PPJoin+,[57] and FastJoin[58] were then compared to the large scale databases, RKB and SWAT.

2.4 | Discussion

Table 1 illustrates the benefits and drawbacks of the current ontology matching approaches. In particular, the current works regarding ontology matching have good results on small-scale databases (i.e., many small and medium instances) and lower-dimensional data (instances with a small or medium data properties) in terms of runtime and the solution quality. However, the current approaches have several limitations, and two key of them being inability to deal with large-scale and high-dimensional data. In this work, we present two data mining-based frameworks to address both these limitations for exploring clustering and pattern mining regarding ontology matching.

3 | COMI: CLUSTERING FOR ONTOLOGY MATCHING

3.1 | Ontology matching problem

Definition 1. Consider the set of l ontologies $\mathcal{O} = \{\mathcal{O}_1, \ldots, \mathcal{O}_l\}$, each ontology $\mathcal{O}_i$ shows the set of m_i instances such that $\mathcal{I}^i = \{\mathcal{I}^i_1, \ldots, \mathcal{I}^i_{m_i}\}$, and n_i properties or attributes $\mathcal{P} = \{\mathcal{P}^i_1, \ldots, \mathcal{P}^i_{n_i}\}$. The purpose and problem statement of the ontology matching problem by instances is to determine the common properties among ontologies, that is, to determine the function $\mathcal{M}$ such that:

$$\mathcal{M}(\mathcal{O}_i, \mathcal{O}_j) = \left| \left\{ \bigcup_{l \leq m_i, s \leq m_j} \mathcal{M}_{ij}(I^i_l, I^j_s) \right\} \right|, \tag{1}$$

The Equation (1) refers to the union of all the common instances between two ontologies, where two instances are similar to a set of data properties (see Equation (2)).

TABLE 1 Classification of ontology matching approaches and their limitations

Strategy	Models and Algorithms	Limitations
Traditional	VMI[31]	
	RiMOM[5]	Unable to deal with large-scale data.
	MERGILO[33]	
	Li et al.[32]	Matching based on prior results, which decreases
	CILIOS[34]	the overall accuracy performance.
	Rosaci[35]	
	Elmagarmid et al.[36]	Unable to deal with a high number of data properties.
Data mining	EIFPS[4]	
	LOM[2]	Use an old matching mechanism.
	CSA[37]	
	Algergawy et al.[39]	High time consumption due to:
	Xue et al.[59]	1. The similarity graph mechanism;
	Xue et al.[60]	2. Combination of different measures.

Abbreviations: CSA, Cluster-based similarity aggregation; EIFPS, Extended Inverse Functional Property Suite.

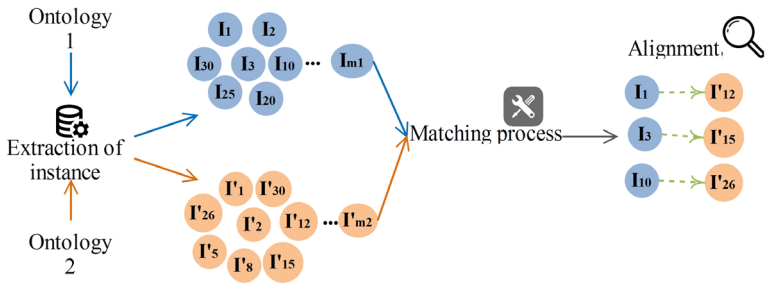

FIGURE 2 Ontology matching-based instance

$$\mathcal{M}_{ij}(I_l^i, I_s^j) = \{p | p \in I_l^i \wedge p \in I_s^j\}. \quad (2)$$

The naive approach of the ontology matching problem is to scan all values of the instances among the ontologies and make comparisons. The process of matching determines the outcome of the alignment. Each matching may lead to different alignment instances. Each result of alignment is then evaluated and compared to the alignment of reference. The reference alignment is an alignment proposed by a user or expert in the particular domain. The alignment of references includes all the common ontology instances.

For instance, Figure 2 presents a simple example for ontology matching by instances. Consider two ontologies in the running example, such as $\mathcal{O}_1$ and $\mathcal{O}_2$. The first step aims at extracting the set of instances $\mathcal{I}^1_{m_1}$ and $\mathcal{I}^2_{m_2}$ and grouping them into several subsets. The matching process is then performed to derive an alignment among the ontologies. The reference alignment represents the set of the common instances among two ontologies. Thus, the optimal matching between $\mathcal{O}_1$ and $\mathcal{O}_2$ is, for example, $i_1 = i'_{12}$, $i_3 = i'_{15}$, and $i_{10} = i'_{26}$.

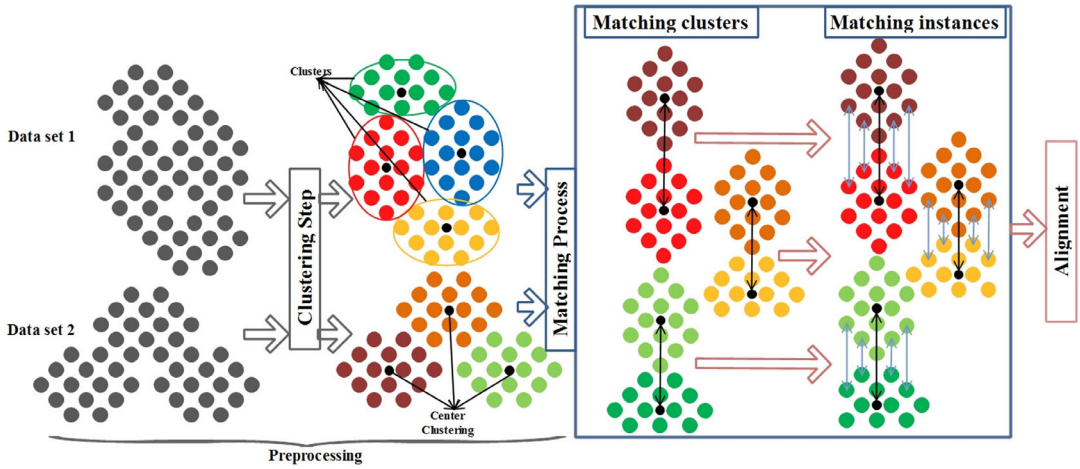

FIGURE 3 COMI: Clustering for ontology matching-based instances

In the ontology matching problem by instances, the most important issue is to find the maximum real-world across two large-scale ontologies. Consider m_1 and m_2 as the number of instances of two ontologies. If the size of the instances is very large, for example, more than 10 million, then it requires high computational cost (e.g., GeoNames [3] dataset with more than 10 million geographical names). To handle the large-scale ontology data, we present the clustering-based method to find the highly correlated subsets for ontology matching by instances.

3.2 | Principle

The aim of COMI is splitting the whole set of instances of each ontology into several dependent clusters. Each cluster then contains highly correlated instances to be processed later. Next, as explained in Figure 3, COMI explores the instances of the clusters to find the common features. It mainly includes the clustering and matching processes. In the clustering process, the instance set is divided into several collections of subinstances (clusters) using data mining techniques. This step is considered to be preprocessing. The set of instances is then grouped into different clusters with a small number of instances. Each cluster of instances shares the maximum number of common properties; thus, the instances of a cluster are highly correlated. During the matching process, COMI explores the instances of the clusters to find the alignments. Instead of performing the alignment operation between the instances of ontologies one by one, the alignment is established between the instances of the two ontologies and their representative clusters. Algorithm 1 presents the COMI pseudo-code. The set of instances $\mathcal{I}$ is considered as input, and the best alignment as $\mathcal{A}$. The set of clusters is represented by $\mathcal{G}$, and the set of centroids is stated as g. The first step is to randomly initialize the centroids using the function *InitializeCenters()*. The first loop is performed from lines 6 to 17, which scans all the set of instances I. The function $\mathcal{D}_{\text{instance}}(e, g_1)$ calculates the distance between the instance and the first centroid g_1. Consider $e = \{(\text{Name, Joe}), (\text{age, 26}), \text{and } (\text{type, man})\}$ and the centroid is set as $g_1 = (26, \text{man, USA})$, $\mathcal{D}_{\text{instance}}(e, g_1)$ to calculate the intersection of values, which is set to 2. The loop from lines 9 to 13 finds the smallest distance

[3] http://www.geonames.org/about.html

Algorithm 1. COMI: Clustering for Ontology Matching

Input : $\mathcal{I}^i = \{\mathcal{I}^i_1, \mathcal{I}^i_2 ... \mathcal{I}^i_{n_i}\}$: the set of n instances of the ontology $\mathcal{O}^i$.
Output: $\mathcal{A}$: Alignment set.
********** **centroid initialization** **********
InitializeCenters(g^i)
for each instance e $\in$ $\mathcal{I}^i$ **do**
  dis $\leftarrow$ $\mathcal{D}_{\text{instance}}$(e,$g^i_1$) ▷ See Equation (3)
  r $\leftarrow$ 1
  for j=2 to k **do** ▷ Search the minimal distance between the instance and all centroid
    d $\leftarrow$ $\mathcal{D}_{\text{instance}}$(e,$g^i_j$) ▷ See Equation (3)
    if d < dis **then**
      dis $\leftarrow$ d
      r $\leftarrow$ j
    end if
  end for
  AddElement(e,c^i_r) ▷ Add the instance in the appropriate cluster
end for
repeat ▷ Update centroids centers
  change $\leftarrow$ false
  g^i_{new} $\leftarrow$ UpdateCenter (g,$\mathcal{G}^i$)
  if g^i != g^i_{new} **then**
    change $\leftarrow$ true
  end if
until *change == false*
******** $\mathcal{MIC}^i(n_i \times k_i)$**: Matrix of distances between** $\mathcal{I}^i$ **and** g^i ********
for i = 1 to n **do**
  for j=1 to k **do**
    $MIC^i[i,j]$ $\leftarrow$ $\mathcal{D}_{\text{instance}}$ ($\mathcal{G}^i_{ij}, g_j$) ▷ $\mathcal{G}^i_{ij}$ is the ith instance of the $\mathcal{G}^i_j$ cluster
  end for
end for
  return $\mathcal{MIC}^i$
********** **Matching Process** **********
list $\leftarrow$ $\emptyset$
for p = 1 to k_i **do** ▷ Finding the similar clusters
  min $\leftarrow$ $\mathcal{D}_{\text{centroid}}$ (g^i_p, g^j_1) ▷ See Equation (4)
  indice $\leftarrow$ 1
  for q = 2 to k_j **do**
    d $\leftarrow$ $\mathcal{D}_{\text{centroid}}$($g^i_p, g^j_q$) ▷ See Equation (4)
    if d < min **then**
      min $\leftarrow$ d
      r $\leftarrow$ j
    end if
  end for
  list $\leftarrow$ list $\cup$ AddClusters (c^i_p, c^j_r)
end for
$\mathcal{A}$ $\leftarrow$ $\emptyset$
for each (p, q) $\in$ list **do** ▷ Finding the similar instances in the similar clusters
  min $\leftarrow$ $n_i \times n_j$ ▷ Initialize minimum distance
  for each instance (e_1, e_2) $\in$ $(\mathcal{G}^i_p, \mathcal{G}^j_q)$ **do**
    d $\leftarrow$ $\mathcal{D}_{\text{matching}}$ (e_1, e_2) ▷ See Equation (5)
    if d $\leq$ min **then**
      min $\leftarrow$ d
      $\mathcal{A}_{p,q}$ $\leftarrow$ (e_1, e_2)
    end if
  end for
  $\mathcal{A}$ $\leftarrow$ $\mathcal{A}$ $\cup$ $\mathcal{A}_{p,q}$
end for
  return $\mathcal{A}$

between the instance e and all the centroids in g, where it conserves the range r. Line 16 affects the instance e to the list of cluster r, which represents the minimum distance using the function *AddElement()*. From lines 18 to 24, the centers are updated and kept in the set g'. If g_{new} is equal to the previous center in g, then the clustering process is then terminated; otherwise, the same process is repeated until g_{new} and g become the same. The final clustering results are then kept in a matrix structure, which is called $\mathcal{MIC}$. Each element $\mathcal{MIC}[i][j]$ is the distance between the centroid g_j and the ith instance of the jth cluster, denoted as $\mathcal{C}_j^i$ (lines 25–29). From lines 35 to 45, the algorithm scans the set of centroids G^i, G^j of the two ontologies O^i and O^j, and the minimum distance between two centroids with the function $\mathcal{D}_{\text{centroid}}(g_{l_1}^i, g_{l_2}^j)$ is determined. The minimum distance is selected and the two clusters are added to the list of the alignment clusters *list* using the function *AddClusters*(). From lines 48 to 58, the algorithm scans the whole instances of the two aligned clusters. Here, p and q are represented as the two selected clusters, and the loop from lines 50 to 56 scans all the instances e_1 and e_2 for both clusters p and q, and the minimum distance can be computed using the formula $\mathcal{D}_{\text{instance}}$. For the set of aligned $\mathcal{A}$, the alignment results of the clusters p and q are then added and denoted as $\mathcal{A}_{p,q}$. This process is repeated for all the clusters in *list*. Next, the decomposition and matching steps are described in detail.

3.3 | Decomposition

The ontology matching problem usually deals with a large number of instances, which is a non-trivial task, especially when the ontology is large scale. Thus, it is necessary to decompose the huge data into a small number of clusters that reduce the difficultly of the matching process (Algorithm 1). In this section, we investigate the partitioning-based approach and utilize the k-means[25] algorithm for the matching problem. The distance and the centroid computation are defined below.

Definition 2 (distance between instances). We note p_{jl}^i as the value of the property $\mathcal{P}_j^i$ in the instance $\mathcal{I}_l^i$ of the ontology $\mathcal{O}_i$. The distance $\mathcal{D}_{\text{instance}}$ between two instances $\mathcal{I}_{l_1}^i$ and $\mathcal{I}_{l_2}^i$ is then defined as

$$\mathcal{D}_{\text{instance}}(\mathcal{I}_{l_1}^i, \mathcal{I}_{l_2}^j) = n_i - \left(\left| \bigcup_{j=1}^{n_i} p_{jl_1}^i \cap p_{jl_2}^i \right| \right). \tag{3}$$

To compute the centroids, we consider the set of instances of the cluster $\mathcal{G}_s = \{\mathcal{I}_1^{(s)}, \mathcal{I}_2^{(s)}, \ldots, \mathcal{I}_{|G_s|}^{(s)}\}$. The aim is to find a gravity center of this set that is also an instance. Inspired by the centroid formula developed in prior work,[61] we compute the centroid μ_s. The frequency of each value is calculated for all the instances of the cluster G_s. The values of instances in G_s are sorted according to their frequency, and only the n_i frequent value is assigned to μ_s as $\mu_s = \{j | j \in \mathcal{F}_{n_i}\}$, where $\mathcal{F}_{n_i}$ denotes the set of the n_i frequent items of the cluster G_s. k-means is a well-known partitioning-based clustering algorithm. It defines k clusters and divides the set of instances of each ontology into k subsets by considering the correlation between the instances of the same cluster. The k-means process starts by initializing k clusters. The k instances from the given ontologies can be randomly selected. Then, it scans each instance from the whole set, calculates the distance between this instance and all the centroids, and assigns it to the cluster with the nearest centroid. After all the instances are examined, the centroid of each cluster is then updated. This process is repeated until the cluster centroid becomes stable.

3.4 | Matching process

This step benefits from the clustering step by defining a new matching strategy instead of computing the similarity between two pairs of instances of the given ontologies. The similarity measures between the centroids of the clusters and the instances are then determined. Two distances are defined: the first distance aims at determining the similarity between two centroids in different ontologies while the second represents the distance between two instances in different ontologies (Algorithm 1). The principal idea of the matching process is to find two highly correlated clusters among ontologies by considering the minimum distances of them. After that, the instances among the clusters are check to attempt to find the rough instances. Consider $g^i_{l_1}$ and $g^j_{l_2}$ as two centroids of the input ontologies.

Definition 3 (distance between centroids). g, and g' are considered to be two centroids of two different ontologies. The distance $\mathcal{D}_{\text{matching}}$ between the two centroids g and g' is defined as

$$\mathcal{D}_{\text{centroid}}(g, g') = |g| + |g'| - |g \cap g'|. \tag{4}$$

It should be noted that $|g|$, $|g'|$, and $|g \cap g'|$ are the number of properties of the centroids g and g' and their intersection, respectively.

Definition 4 (matching instances). We define the distance $\mathcal{D}_{\text{matching}}$ between two instances $\mathcal{I}^i_{l_1}$ and $\mathcal{I}^j_{l_2}$ as the sum of distances between each instance and its centroid and the distance between the two centroids of these instances as

$$\mathcal{D}_{\text{matching}}(\mathcal{I}^i_{l_1}, \mathcal{I}^j_{l_2}) = \mathcal{D}_1(g_i, g_j) + \mathcal{D}_2(\mathcal{I}^i_{l_1}, g_i) + \mathcal{D}_2(\mathcal{I}^j_{l_2}, g_j), \tag{5}$$

where $\mathcal{D}_1$ is $\mathcal{D}_{\text{centroid}}$ and $\mathcal{D}_2$ is $\mathcal{D}_{\text{instance}}$.

The complexity of COMI depends on the number of instances n, the number of properties m, the number of clusters k, and the number of matchings r. The decomposition step needs $O(n \times m \times k)$. This process is performed only once for each ontology whatever the number of matchings to be used. Only similar clusters are used during the matching process. This requires $O(\frac{n \times m}{k})$. The total cost of COMI for perform r matching is $O(n \times m \times k + r \times \frac{n \times m}{k})$, which is significantly lower than the baseline solutions that require $O(n \times m \times r)$.

4 | POMI: PATTERN MINING FOR ONTOLOGY MATCHING

4.1 | Principle

POMI, as shown in Figure 4, investigates the correlation between data properties of the ontological systems to obtain the best characteristics for a matching process. It extracts the most relevant data properties that cover as many instances as possible from the pattern mining process.[62] FIM refers to the extraction from the transactions database of the relevant itemsets that accomplish the minimum support limit (*minsup*). In the designed three phases model (mining, pruning, and selection), we follow a classical pattern mining method to efficiently discover the best features of the ontologies. The pruning process is a significant difference between the previous mining strategies and our pattern-mining-based model. Existing strategies list all the patterns that

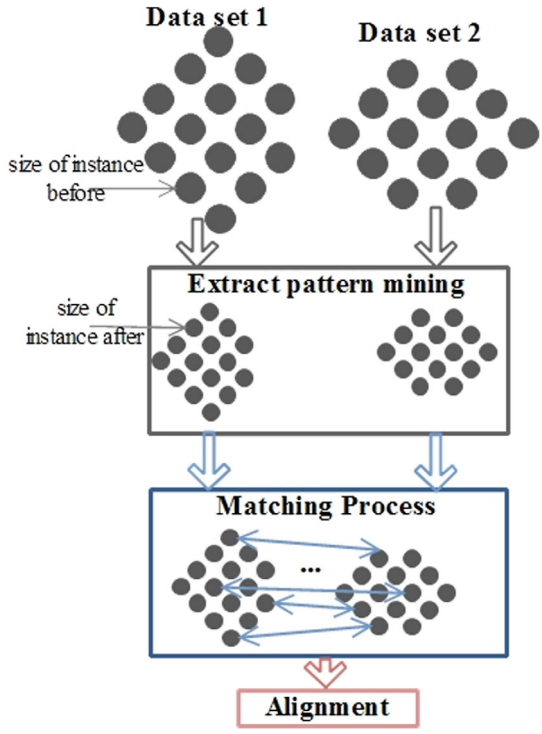

FIGURE 4 POMI: Pattern mining for ontology matching-based instances

exceed minimum support constraints, while our approach considers other measures by discovering a subset of relevant patterns that cover a maximum of transactions in the database (i.e., the instances in the study). The algorithm is presented in the pseudo-code given in Algorithm 2. The mining step is performed from lines 4 to 18, and the pruning strategy runs from lines 21 to 38. The selection and the matching processes are detailed in lines 33–56.

4.2 | Pattern discovery

In the pattern-mining field, the fundamental algorithms that were presented, such as Apriori,[62] DIC,[63] or FPGrowth,[64] require a huge amount of time cost and memory usage to discover the set of frequent itemsets regarding the predefined minimum support threshold. SSFIM[26] was recently presented to discover frequent itemsets within one single pass, and it is an insensitive algorithm for the minimum support threshold. The experimental results showed that the SSFIM has a better performance compared to the state-of-the-art pattern mining algorithms. Thus, in this study, SSFIM is utilized in the designed model to discover the frequent literals (labeled as S) from the set of the instances I. Two main steps are taken for SSFIM: generation and extraction. In the generation stage, beginning with I_1, we refer to Pattern(I_1) in all possible literal combinations of this instance. The outcome is applied to H by generating Pattern(I_1) for each pattern. In the hash table H, the frequency of each pattern is initiated one by one. Then, I_2 for each pattern in Pattern(I_2) is generated in the second instance. If the pattern is available at H, then its frequency must increase by one or a new entry with a fixed frequency of 1 will be made. This is repeated until I is processed for all the instances. The second step discovers the frequent patterns (i.e., frequent literals in the

Algorithm 2. POMI: Pattern mining for Ontology Matching

Input : $\mathcal{I}^i = \{\mathcal{I}^i_1, \mathcal{I}^i_2 ... \mathcal{I}^i_{n_i}\}$: the set of n instances of the ontology $\mathcal{O}^i$.
Output: $\mathcal{A}$: Alignment set.
************mining step**************
for each instance e $\in$ I^i **do** ▷ Extract the frequent itemsets using only a singlepass.
  $F^i_e \leftarrow$ Itemset(e)
  for each element i $\in$ F^i_e **do**
    if i $\in$ H^i **then**
      $Freq_i$++
    else
      AddH(i,1)
    end if
  end for
end for
$S^i \leftarrow \emptyset$
for each element h $\in$ H^i **do** ▷ Prove if the itemsets are insensitive to the minimum support value.
  if Support(h) $> \sigma$ **then**
    $S^i \cup$ h
  end if
end for
************pruning step**************
sol $\leftarrow$ InitialSol(S^i)
$S^{i,*} \leftarrow S^i$
iter $\leftarrow$ 0.
while Pruning$_{max}$(S^i) < m and iter < IMAX **do** ▷ Select the smallest itemsets that cover the largest number of instances.
  neighbors $\leftarrow$ ComputeNeighbors(sol).
  best $\leftarrow$ BestNeighbors(neighbors).
  if Pruning$_{max}$(best) $> S^{i,*}$ **then**
    $S^{i,*} \leftarrow$ best.
  end if
  iter $\leftarrow$ iter + 1.
end while
************selection step**************
$SP^i \leftarrow \emptyset$
for each property p $\in$ P^i **do**
  if Probability(p, S^i) $> \mu$ **then** ▷ A threshold is used to select the appropriate data properties.
    $SP^i \cup$ p
  end if
end for
************matching step**************
for each instance j $\in$ I^i **do**
  $P^i \leftarrow$ SetProperties(j)
  for each instance l $\in$ I^j **do**
    $P^i \leftarrow$ SetProperties(l)
    L $\leftarrow \emptyset$
    for each property p $\in$ P^i **do**
      for each property p' $\in$ P^j **do**
        if Value(p, p') **then** ▷ Comparison of the two instances i and j by taking in consideration the selected set of properties $<P^i, I^i>$ for the i^{th} ontology, and $<P^j, I^j>$ for the jth ontology respectively.
          L $\cup$ {p, p'}
        end if
      end for
    end for
    if L !=$\emptyset$ **then**
      $\mathcal{A} \cup (\{ID_i, ID_j\} \cup$L)
    end if
  end for
end for
  return $\mathcal{A}$

study) from the H hash table. The support for each t pattern is determined (see Equation (6)). If the frequency of t is no less than *minsup*, then t is considered as the frequent literal and is put into the set of S, which is the set of frequent literals.

$$\text{Support}(t) = \frac{h(t).\text{freq}}{|I|}. \tag{6}$$

4.3 | Pruning

The limitation of generic pattern mining is that a large number of frequent patterns are discovered, which results in inefficiency while handling situations with many ontologies. It is a time-consuming and a nontrivial task to analyze a huge number of the discovered patterns. To overcome this limitation, a new strategy is presented to well filter the mined frequent patterns in the mining progress; thus, a small number of meaningful and significant patterns can be discovered to well explain and illustrate the ontology database. Here, we use a novel idea, called **Coverage**, in the designed pruning strategy, which results in keeping fewer and more representable patterns based on the Minimum Description Length principle[65] to cover the largest number of instances from an ontology (Algorithm 2). The number of frequent patterns can be significantly reduced. The discovered patterns in the developed model are different to the maximal[66] or closed[67] frequent patterns. More detailed explanations for the proposed solutions are given below.

Definition 5. Let $S = \{S_1, S_2, \ldots, S_r\}$ be the set of the discovered frequent patterns in the mining progress. The coverage pruning problem is defined by maximizing $\text{Pruning}_{\max}$ as

$$\begin{array}{rccc} \text{Pruning}_{\max} & : & S & \rightarrow & \mathfrak{R} \\ & & S' & \mapsto & \text{Pruning}_{\max}(S'). \end{array} \tag{7}$$

Definition 6. The $\text{Pruning}_{\max}$ is defined as a function that can be used to cover the maximum number of records from the given ontology database. Let $\mathcal{I}(S_i)$ denote the set of instances covered by a pattern S_i. The purpose of the coverage pruning function is to return a subset $S' \subset S$ that maximizes the coverage value and can be defined as

$$\begin{array}{rccc} \text{Pruning}_{\max} & : & S & \rightarrow & \mathfrak{R}^+ \\ & & S' & \mapsto & |\bigcup_{S_i \in S'} \mathcal{I}(S_i)|. \end{array} \tag{8}$$

Definition 7. Finding the minimum subset $S^* \subset S$ is an optimal solution to the coverage pruning problem in an ontology that includes m instances. Here, S^* covers all the records and is then defined as follows:

$$\begin{cases} \text{Pruning}_{\max}(S^*) = m \\ \forall S' \subset S, \\ \text{Pruning}_{\max}(S') = \text{Pruning}_{\max}(S^*) \Rightarrow |S'| \geq |S^*|. \end{cases} \tag{9}$$

As a frequent set of S patterns can be selected from 2^r subsets of possible S subset, to find the optimal subset that meets the limitations of coverage pruning is an NP-complete problem. A thorough search would, therefore, be extremely time-consuming or even impractical if the S

cardinality is large. To tackle this problem, the greedy search approach can be combined with neighboring search to reduce the search space and to provide a reasonable solution rather than an optimal solution globally. We were inspired by the work of Hosseini et al.,[68] where the greedy algorithm is used to list the search tree and perform local searches on each generated node. The set of frequent patterns S, a maximum number of iterations, and the number instances in the given ontology are first considered, and the output result is the set of patterns as S^*. The first progress is created by randomly selecting frequent patterns from S. The solution is then placed in an S^* variable that is the best solution for now. Then, an iterative process is performed to improve the current solution so that a better solution can be obtained. This progress is repeated until S^* is less than m in the number of instances covering the patterns or the iteration number is less than the maximum number of iterations. To improve the current solution, the neighborhood *neighbors* of the solution is determined. All the solutions are produced that can be accomplished by adding another frequent pattern to the current solution. The best solution among those solutions is denoted as *best*, and if it is better than the best solution S^* at the current stage, then the variable S^* is set as *best* based on the pruning function. It should be noted that if the two solutions, such as sol_1 and sol_2, hold the condition as $\text{Pruning}_{\max}(\text{sol}_1) \leq \text{Pruning}_{\max}(\text{sol}_2)$ and $|\text{sol}_1| \leq |\text{sol}_2|$, then sol_1 is considered to be a better solution than that of the sol_2. The reason for this is that the number of patterns should be minimized. A greedy model is first presented to obtain a set of the smallest number of frequent patterns that maximize the number of events covered by the patterns. It should be noted that that other pruning functions can be used for other requirements.

4.4 | Selection

The set of SP is properly selected according to the pruning strategy and the mined frequent literals S. Let $P(i, S)$ denote the probability of the ith property appearing in the set of S frequent literals. A threshold is set in a range of $[0, 1]$ that is used to find the data properties properly. If the probability value for each property is higher than μ, then it is added to the SP set (Algorithm2).

Definition 8. Consider the data property of p and S, which is the set of frequent literals discovered by the pruning step. The p is obtained for matching progress if it satisfies the condition

$$P(p, S) > \mu, \tag{10}$$

where $P(p, S)$ is the probability of the property p in the frequent literals S and μ is the interestingness degree threshold.

4.5 | Matching process

The instances of fundamental ontology are compared to the instances of the second ontology after the selection of the correct data properties. The fundamental ontology of BO in this part is matched with the second ontology of O. Moreover, $< P, I >$ is then considered to be the set of data properties of P and the instances of the fundamental ontology of I. Furthermore, $< P', I' >$ is considered to be the set of data properties of $'P$ and the instances of the second ontology of I'. For this situation, P and P' are then considered to be two sets and are respectively obtained from the described feature selection models. For iterative matching, the entire set of instances for the fundamental ontology of I is then determined and compared to the set of instances in the

second ontology of I'. Those two instances are then compared by determining each value of the i^{th} instance from BO for all the jth instance values from O.

The complexity of POMI depends to the number of instances n, the number of properties m, the number of selected properties m', and the number of matchings r. The pattern mining step needs $O(n \times m)$. This process is performed only once for each ontology, whatever the number of matchings to be used. During the matching process, only the selected properties are used. It should be noted that $m' <<< m$. This requires $O(n \times m')$. The total cost of POMI to perform r matching is $O(n \times m + r \times n \times m')$, which is significantly lower than the baseline solutions which require $O(n \times m \times r)$.

5 | PERFORMANCE EVALUATION

Extensive experiments were conducted on well-known ontology databases to validate the usefulness of proposed COMI and POMI frameworks. The experiments were carried out on a desktop with an Intel *i7* processor and 16 GB of main memory. Java language was used for all the implemented algorithms. The experiments employed three well-known ontology databases that are often used in the ontology matching community (Regarding the tests, each experiment is assigned to the same dataset for all systems). Details are described below.

1. DBpedia[4] is a superficial cross-domain ontology, it was created manually based on Wikipedia. We extract structured content from the information created in Wikipedia. This structured information is available on the World Wide Web. The ontology currently covers 2795 data properties and 4,233,000 instances.
2. The information (i.e., number of instances and data properties) of Ontology Alignment Evaluation Initiative (OAEI)[5] databases is shown in Table 2. OAEI is an international initiative. The increasing number of methods available for the matching ontologies has arisen to this company for the evaluation of these methods. Among the objectives of OAEI, it is to assess the strengths and weaknesses of alignment systems, compare the performance of techniques, and improve assessment techniques to help improve the work on the matching ontologies.
3. The Smart City Use case[6] contains more than 400,000 sensing objects allocated around the world. It also has varied aspects for the data distribution. Moreover, it has more than 8.5 billion sensor records in the dataset.

5.1 | Performance on DBpedia

Two baseline algorithms, EIFPS[4] and RiMOM,[5] were considered in this experiment. The quality of the matching process of the ontology was evaluated using the F-measure, which is used to define the output of the matching process A and a reference alignment R as

$$\text{F} - \text{measure}(A, R) = \left(\frac{2 \times \text{Precision} \times \text{Recall}}{Precision + \text{Recall}} \right) \times 100. \tag{11}$$

[4]http://wiki.dbpedia.org/Datasets
[5]http://oaei.ontologymatching.org
[6]http://www.noaa.org/

TABLE 2 Ontology alignment evaluation initiative databases description

Ontology Name	Number of instances	Number of data properties
OntoA_dis	29,645	11
OntoB_dis	15,556	11
OntoA_rec	15,556	11
OntoB_dis	1708	11
Onto_a_id	1330	5
Onto_b_id	2649	4
Onto_a_sim	173	5
Onto_b_sim	172	5
*onto*101	57	46
*onto*104	56	46
*onto*202	57	46
*onto*230	47	49
*IIMB*000	12,333	13
*IIMB*104	12,338	13
*person*11	500	14
*person*12	500	12
*person*21	600	14
*person*22	400	12

The precision was computed as $\frac{|R \cap A|}{|A|}$, and the recall was computed as $\frac{|R \cap A|}{|R|}$. It should be noted that the ground-truth represented by the best alignment was annotated by domain experts, which is a human-being procedure.

5.1.1 | Runtime performance

The first set of experiments was performed to compare the runtime of COMI with state-of-the-art approaches under varied clusters. COMI$|X|$, where $|X|$ is the number of the clusters, was used in the COMI approach. The runtime computed in this experiment was the runtime of the whole COMI process including the decomposition and matching steps. Figure 5 shows the runtime of the five approaches (COMI2, COMI5, COMI10, EIFPS, and RiMOM), where the percentages of instances varied from 25% to 100%. When the number of matchings increased from 1000 to 100,000, COMI outperformed the two other approaches. Moreover, the runtime of COMI remained stable, while the baseline approaches required additional computing time for a large number of instances and many matchings. Thus, the two compared approaches (EIFPS and RiMOM) needed more than 600 s for handling the 100,000 matchings in the whole DBpedia ontology database, and the designed COMI10 (COMI with 10 clusters) required only 54 s. These results are explained by the fact that our approach only considers highly correlated instances in the matching process by developing an efficient strategy to explore the information provided in each cluster of instances. The results also show that by increasing the number of clusters from 2

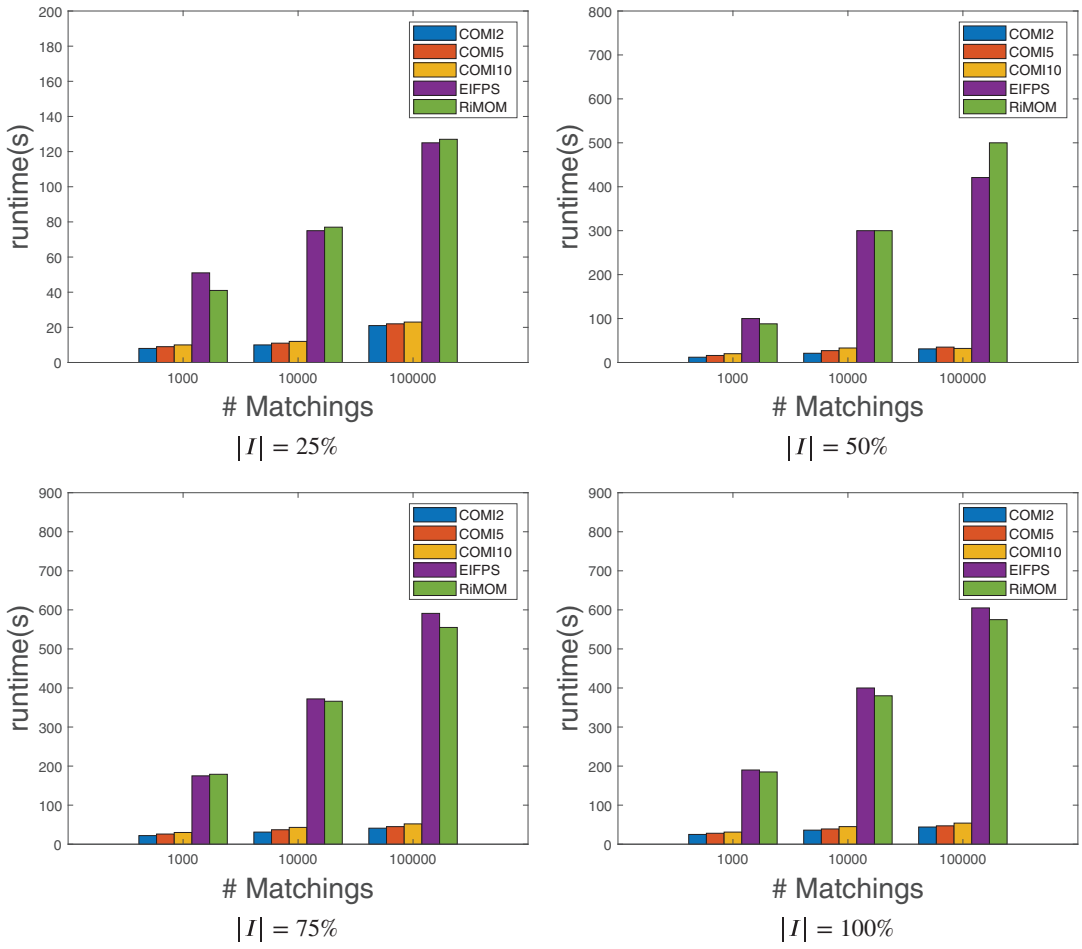

FIGURE 5 A comparison of the clustering for ontology matching-based instances, extended inverse functional property suite, and RiMOM computational costs using DBpedia varying from 25% to 100% and the number of matchings varying from 1000 to 100,000

to 10, a slight difference in terms of execution time could be obtained. The clustering process was only adopted in the preprocessing step.

5.1.2 | Solution quality

A second set of experiments was performed to compare the quality of the solutions by COMI with the state-of-the-art EIFPS and the RiMOM algorithms using the DBPedia ontology database. Figure 6 shows the results of the five approaches (COMI2, COMI5, COMI10, EIFPS, and RiMOM), where the percentages of the instances and the properties varied from 25% to 100%, respectively. The results reveal that the COMI10, EIFPS, and RiMOM approaches had a similar quality, while COMI5 and COMI2 provided less quality compared to the first ones. Thus, if more clusters are generated, then the designed COMI can achieve better results; for example, 10 clusters for DBPedia data. Moreover, COMI10 had better performance than the EIFPS and RiMOM algorithms

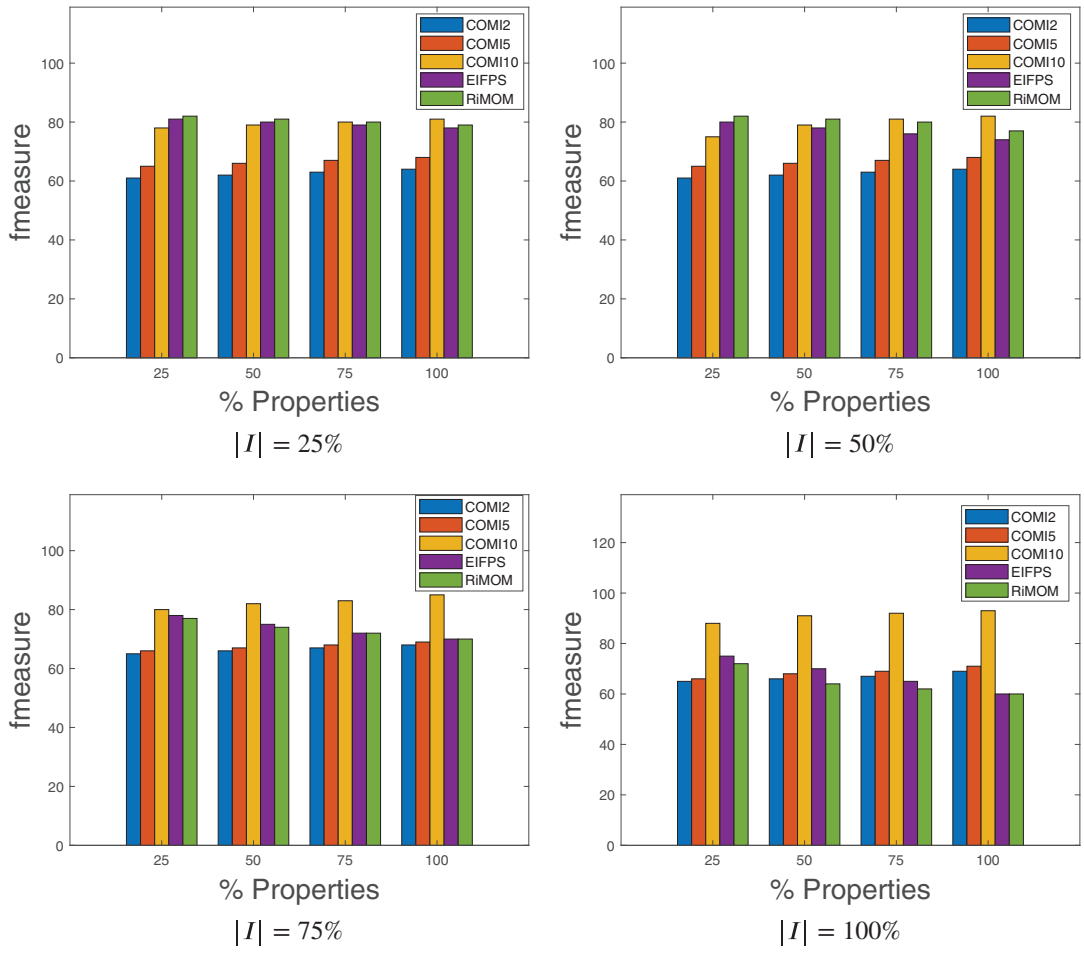

FIGURE 6 The compared results of clustering for ontology matching-based instances, extended inverse functional property suite, and RIMOM's F-measure performance with DBpedia by varying the percentage of instances and the percentage of properties from 25% to 100%

under large- and high-dimensional ontology data. For instance, when the percentage of properties and instances was set to 25%, the F-measure of EIFPS and RiMOM, respectively, were 81% and 82%, while COMI10 did not reach 80%. However, for 100% of data, the F-measure of COMI was 93%, while the F-measure of the two other approaches was around 60%. We explain this issue by the fact that the clustering quality with $k = 10$ was better than $k = 2$, and $k = 5$. More similar clusters sharing a high number of properties were obtained with $k = 10$, instead of more heterogeneous clusters with different properties that were determined by exploring two and five clusters. Only 2, 5, and 10 clusters were studied in this experiment because the clustering quality was reduced when setting the number of clusters above 10.

It can be concluded from these results that COMI achieved the best results in terms of runtime compared to the existing ontology matching algorithms, particularly for large ontologies like the DBPedia database. Moreover, this issue does not degrade the quality of the solution if the appropriate number of clusters is chosen. The quality of the matching between the POMI framework and baseline algorithms (i.e., EIFPS and RiMOM) conducted on the OEAI ontology database is

TABLE 3 The compared results recall, precision, and F-measure of Pattern mining for Ontology Matching-based Instances (POMI), Extended Inverse Functional Property Suite (EIFPS), and RiMOM using DBpedia varying from 20% to 100% of instances (%I) and data properties (%P) from 20% to 100%

		POMI			EIFPS			RiMOM		
% I	%P	Rec.	Prec.	F-meas.	Rec.	Prec.	F-meas.	Rec.	Prec.	F-meas.
20	20	0.97	0.95	0.96	0.97	0.94	0.95	**0.98**	**0.95**	**0.96**
	50	**0.97**	**0.95**	**0.96**	0.92	0.92	0.92	0.95	0.93	0.94
	80	**0.97**	**0.95**	**0.96**	0.90	0.91	0.90	0.93	0.92	0.92
	100	**0.97**	**0.95**	**0.96**	0.88	0.90	0.89	0.89	0.90	0.89
50	20	**0.96**	**0.94**	**0.95**	0.93	0.92	0.92	0.95	0.92	0.93
	50	**0.96**	**0.94**	**0.95**	0.89	0.87	0.88	0.91	0.89	0.90
	80	**0.96**	**0.94**	**0.95**	0.87	0.84	0.85	0.89	0.86	0.87
	100	**0.96**	**0.94**	**0.95**	0.85	0.82	0.83	0.87	0.83	0.85
80	20	**0.95**	**0.92**	**0.93**	0.90	0.89	0.89	0.91	0.90	0.90
	50	**0.95**	**0.92**	**0.93**	0.88	0.86	0.87	0.89	0.88	0.88
	80	**0.95**	**0.92**	**0.93**	0.82	0.80	0.81	0.83	0.81	0.82
	100	**0.95**	**0.92**	**0.93**	0.78	0.75	0.76	0.80	0.79	0.79
100	20	**0.94**	**0.90**	**0.92**	0.85	0.82	0.83	0.87	0.86	0.86
	50	**0.94**	**0.90**	**0.92**	0.83	0.81	0.82	0.84	0.82	0.83
	80	**0.94**	**0.90**	**0.92**	0.78	0.75	0.76	0.80	0.77	0.78
	100	**0.94**	**0.90**	**0.92**	0.74	0.70	0.72	0.73	0.72	0.72

compared in Table 3. The POMI framework exceeded the other two algorithms on quality (recall, precision, and F-measure) in all the cases by changing the percentage of the data properties and the percentage of instances from 20% to 100% in all the cases except in the first case that included 20% of databases and instances. This also shows that the increase in data properties and the number of instances did not affect POMI quality. Thus, the POMI quality was up to 92%, while the EIFS quality and the RiMOM quality were below 70% and 72%, respectively. These results were achieved by the pattern mining techniques that obtained the most relevant data properties of ontologies.

5.2 | Performance on OAEI

In this experiment, the scalability of the COMI and POMI frameworks were evaluated. Several criteria, such as the quality of the solutions, the computational cost (i.e., runtime), and memory usage, were evaluated on the OAEI ontology databases. Standard Java API was used in the experiments to show the memory usage of the compared algorithms. Results in Table 4 present the F-measure, CPU time, and memory usage of POMI and COMI under various ontology databases and strategies (i.e., exhaustively enumerates all possible matching of the two ontologies). As shown, POMI achieved the best results compared to the other two strategies in terms of F-measure for 15 and 18 cases. The quality of POMI in all the cases was up to 92%, while the quality of the

TABLE 4 A comparison of the F-measure, CPU (sec.), and memory usage (MB) of the three approaches (Exhaustive, clustering for ontology matching-based instances [COMI], and Pattern mining for Ontology Matching-based Instances [POMI])

	Exhaustive			COMI			POMI		
Data	F-meas.	CPU Time	Mem.	F-meas.	CPU Time	Mem.	F-meas.	CPU Time	Mem.
Dis_1	0.76	3.25	115	0.84	**2.12**	**95**	**0.93**	2.61	102
Dis_2	0.77	4.26	158	0.88	**3.91**	**112**	**0.95**	3.99	115
Rec_1	0.79	5.21	136	0.93	**4.96**	**118**	**0.97**	4.99	119
Rec_2	0.75	7.12	159	0.91	**6.02**	**124**	**0.96**	6.15	126
ID_1	0.79	6.15	171	0.92	6.36	**105**	**0.98**	**6.21**	101
ID_2	0.80	10.23	166	0.91	9.12	**99**	**0.95**	**9.02**	103
Sim_1	0.74	12.98	147	0.91	10.12	**113**	**0.95**	**9.85**	117
Sim_2	0.72	14.13	151	0.90	**10.18**	**88**	**0.94**	11.02	92
O_{101}	0.74	11.02	110	0.92	11.00	101	**0.93**	**10.02**	**97**
O_{104}	0.73	14.15	187	**0.94**	**12.25**	102	0.92	13.02	**95**
O_{202}	0.81	18.69	**84**	0.92	17.65	111	**0.93**	**14.23**	88
O_{230}	0.82	19.36	82	**0.95**	21.02	78	0.92	**18.26**	**67**
B_{000}	0.89	**8.26**	**83**	0.95	9.26	87	**0.95**	10.29	93
B_{104}	0.81	12.25	112	**0.96**	**4.12**	92	0.95	5.26	**89**
P_{11}	0.83	5.26	129	0.97	4.26	**81**	**1.0**	**3.77**	85
P_{12}	0.85	12.25	127	0.97	10.25	83	**1.0**	**9.36**	**81**
P_{21}	0.84	11.29	124	0.98	12.03	80	**1.0**	**8.76**	**78**
P_{22}	0.87	15.23	119	0.99	**12.36**	**75**	**1.0**	13.62	77

COMI and exhaustive was less than 84% and 72%. These results were achieved with the knowledge discovered by POMI, which allowed the dimensional space of ontology databases to be reduced better. The results also showed that the memory usage and runtime performance of both COMI and POMI converged to the same values. The exhaustive approach, however, achieved the worst results of both measures, which can be attributed to the fact that the exhaustive strategy listed all the combinations without increasing the search process. The other two strategies enhanced the exploration of solution space by using the clusters and the relevant discovered patterns.

5.3 | Case study on smart-city semantic modeling

The last set of experiments aimed to show the ability of COMI and POMI algorithms to deal with semantic modeling in smart-city environments. While plenty of proposals have been made related to smart-cities data, the semantic modeling from these data is an open research problem in the smart-city community. In this study, we deal with this challenging issue by applying the ontology matching process on the smart-city data described in http://www.noaa.org/. Table 5 shows

TABLE 5 A comparison of F-measure and CPU of pattern mining for ontology matching-based instances (POMI), clustering for ontology matching-based instances (COMI), and the RiMOM using the smart-city data by varying both the percentage of instances (%I) and the percentage of the data properties (%P) from 20% to 100%

		POMI		COMI		RiMOM	
% I	%P	F-measure	CPU (s)	F-measure	CPU (s)	F-measure	CPU (s)
20	20	0.97	0.95	0.96	0.97	0.94	0.95
	50	**0.97**	**0.95**	**0.96**	0.92	0.92	0.92
	80	**0.97**	**0.95**	**0.96**	0.90	0.91	0.90
	100	**0.97**	**0.95**	**0.96**	0.88	0.90	0.89
50	20	**0.96**	**0.94**	**0.95**	0.93	0.92	0.92
	50	**0.96**	**0.94**	**0.95**	0.89	0.87	0.88
	80	**0.96**	**0.94**	**0.95**	0.87	0.84	0.85
	100	**0.96**	**0.94**	**0.95**	0.85	0.82	0.83
80	20	**0.95**	**0.92**	**0.93**	0.90	0.89	0.89
	50	**0.95**	**0.92**	**0.93**	0.88	0.86	0.87
	80	**0.95**	**0.92**	**0.93**	0.82	0.80	0.81
	100	**0.95**	**0.92**	**0.93**	0.78	0.75	0.76
100	20	**0.94**	**0.90**	**0.92**	0.85	0.82	0.83
	50	**0.94**	**0.90**	**0.92**	0.83	0.81	0.82
	80	**0.94**	**0.90**	**0.92**	0.78	0.75	0.76
	100	**0.94**	**0.90**	**0.92**	0.74	0.70	0.72

the results of the three approaches (POMI, COMI, and RiMOM), where the percentages of the instances and the properties varied from 20% to 100%. The results revealed that the COMI and POMI outperformed RiMOM in terms of runtime and solution quality. These results confirm again the usefulness of COMI and POMI for solving the ontology matching problem and their ability to deal with heterogeneous large-scale data. From our extensive experiments dealing with smart-city data, some perspectives remain to be studied:

1. Outlier Detection: Many outliers were found in the experiments. These outliers reduced the overall performance of the ontology matching process. It would be beneficial to remove them in the preprocessing step. One solution is to apply the existing outlier detection algorithms, such as the local outlier factor and k nearest neighbors. A local reachability distance between properties and instances should be developed to adapt these algorithms for an ontology.
2. Crowdsourcing: Ontology matching solutions could identify different alignments from the same data. The problem is how to decide which alignments are useful for the city planners. A crowdsourcing approach may be applied to improve the usefulness of the detected alignment, where different ontology matching approaches should work together to identify the best alignments delivered to city planners. Agents represented by approaches and programs could find locally the alignments and send them to the city planners. Then, the city planners could use crowd-sourcing environments to find the best alignment for the smart city semantic modeling.

3. Missing of ground truth: Missing of the ground truth is a common problem in evaluating ontology matching algorithms, in particular, for real scenarios, such as smart-city semantic modeling. As challenges for future research regarding the quality assessment of ontology matching results, the following issues and research questions remain to be addressed:

- Defining useful, publicly available benchmark smart-city data for semantic modeling problems is beneficial for analyzing the ontology matching algorithms.
- It would be very useful to identify the meaningful criteria for an internal evaluation of ontology matching. One way to address this challenging issue is to provide unified ranking-function scores to rank the alignments. These functions should be independent of the whole process for identifying the best alignments.

6 | CONCLUSIONS

This paper presented two new frameworks, called COMI and POMI, which are cluster-based and pattern mining-based approaches, to solve the ontology matching problem. COMI utilizes the clustering method to solve the matching problem among the ontologies, and it mainly consists of two steps. The first step aims at grouping the highly correlated instances of each ontology into similar clusters using the k-means approach. This is a preprocessing step and is only performed once. Then, the extracted knowledge is then used to find the matching between the instances within the ontologies. POMI selects the most frequent data properties that describe the overall instances of that ontology and explore different correlations between data properties. To evaluate the performance of COMI and POMI, several experiments were carried out on the DBpedia and OEAI ontology databases. The experimental results showed that COMI is much faster than the baseline EIFPS and RiMOM algorithms, and POMI gives good quality compared to EIFPS and RiMOM. Furthermore, a case study on smart-city semantic modeling was given, demonstrating the ability of COMI and POMI to deal with heterogeneous large-scale smart-city data. In our future work, other data mining techniques, such as more pruning strategies[69,70] and high-utility pattern mining,[19,71] could be used for extracting more relevant knowledge for helping the ontology matching process. Using emergent HPC, such as GPU,[72-74] to handle the very large-scale ontology databases will also be considered as an extension of this in future works. In addition, using the clustering in other semantic modeling such the integration of existing databases and building of shareable databases are the further research topics in the future.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are openly available in DBpedia at http://wiki.dbpedia.org/Datasets, OAEI at http://oaei.ontologymatching.org, Smart City Use case at http://www.noaa.org/.

ORCID

Alberto Cano https://orcid.org/0000-0001-9027-298X
Jerry Chun-Wei Lin https://orcid.org/0000-0001-8768-9709

REFERENCES

1. Smith B, Ashburner M, Rosse C, et al. The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. *Nat Biotechnol*. 2007;25(11):1251.

2. Cerón-Figueroa S, López-Yáñez I, Alhalabi W, et al. Instance-based ontology matching for e-learning material using an associative pattern classifier. *Comput Hum Behav*. 2017;69:218-225.
3. Iwata T, Kanagawa M, Hirao T, Fukumizu K. Unsupervised group matching with application to cross-lingual topic matching without alignment information. *Data Min Knowl Disc*. 2017;31(2):350-370.
4. Niu X, Rong S, Wang H, Yu Y. An effective rule miner for instance matching in a web of data. Paper presented at: Proceedings of the ACM International Conference on Information and Knowledge Management. Maui, HI, USA; 2012:10851-094.
5. Shao C, Hu L-M, Li Juan- Z, Wang Z-C, Chung T, Xia J-B. RiMOM-IM: a novel iterative framework for instance matching. *J Comput Sci Technol*. 2016;31(1):185-197.
6. Del Vescovo C, Parsia B, Sattler U, Schneider T. The modular structure of an ontology: Atomic decomposition. *Joint Conf Artif Intell*. 2011;22(3):2232.
7. Grau BC, Horrocks I, Kazakov Y, Sattler U. Modular reuse of ontologies: theory and practice. *J Artif Intell Res*. 2008;31:273-318.
8. Grau BC, Parsia B, Sirin E, Kalyanpur A. Modularity and web ontologies. *Proceedings KR-2006*. Menlo Park, California, USA: AAAI Press; 2006:198-209.
9. Belhadi H, Akli-Astouati K, Djenouri Y, Chun-Wei LJ. Exploring pattern mining for solving the ontology matching problem. Paper presented at: Proceedings of the European Conference on Advances in Databases and Information Systems. Bled, Slovenia; 2019:85-93.
10. Belhadi H, Akli-Astouati K, Djenouri Y, Chun-Wei LJ. Data mining-based approach for ontology matching problem. *Appl Intell*. 2020;50(4):1204-1221.
11. Amin MB, Batool R, Khan WA, Lee S, Huh E-N. SPHeRe. *J Supercomput*. 2014;68(1):274-301.
12. Thayasivam U, Doshi P. Speeding up batch alignment of large ontologies using MapReduce. Paper presented at: Proceedings of the IEEE International Conference on Semantic Computing. Irvine, CA, USA; 2013:110-113.
13. Ochieng P, Kyanda S. A statistically-based ontology matching tool. *Distrib Parallel Databases*. 2018;36(1):195-217.
14. Xue X, Pan J-S. An overview on evolutionary algorithm based ontology matching. *J Inf Hiding Multimed Signal Process*. 2018;9:75-88.
15. Acampora G, Loia V, Salerno S, Vitiello A. A hybrid evolutionary approach for solving the ontology alignment problem. *Int J Intell Syst*. 2012;27(3):189-216.
16. Xue X, Liu J. Collaborative ontology matching based on compact interactive evolutionary algorithm. *Knowl-Based Syst*. 2017;137:94-103.
17. Belhadi H, Akli-Astouati K, Djenouri Y, Chun-Wei LJ, Wu JMT. GFSOM: genetic feature selection for ontology matching. Paper presented at: Proceedings of the International Conference on Genetic and Evolutionary Computing. Changzhou, China; 2018:655-660.
18. Carpineto C, Osiński S, Romano G, Weiss D. A survey of web clustering engines. *ACM Comput Surv (CSUR)*. 2009;41(3):17.
19. Djenouri Y, Djenouri D, Chun-Wei LJ, Belhadi A. Frequent itemset mining in big data with effective single scan algorithms. *IEEE Access*. 2018;6:68013-68026.
20. Belhadi A, Djenouri Y, Lin J, Zhang C, Cano A. Exploring pattern mining algorithms for hashtag retrieval problem. *IEEE Access*. 2020;8:10569-10583.
21. Belhadi A, Djenouri Y, Lin J, Cano A. A general-purpose distributed pattern mining system. *Appl Intell*. 2020;50:2647-2662.
22. Djenouri Y, Belhadi A, Fournier-Viger P, Chun-Wei LJ. Fast and effective cluster-based information retrieval using frequent closed itemsets. *Inf Sci*. 2018;453:154-167.
23. Djenouri Y, Zimek A. Outlier detection in urban traffic data. Paper presented at: Proceedings of the International Conference on Web Intelligence, Mining and Semantics. Novi Sad, Serbia; 2018:1-12.
24. Djenouri Y, Belhadi A, Fournier-Viger P. Extracting useful knowledge from event logs: a frequent itemset mining approach. *Knowl-Based Syst*. 2018;139:132-148.
25. MacQueen J. Some methods for classification and analysis of multivariate observations. *Berkeley Symp Math Stat Probab*. 1967;1(14):281-297.
26. Djenouri Y, Comuzzi M, Djenouri D. SS-FIM: single scan for frequent itemsets mining in transactional databases. Paper presented at: Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining. Jeju, Korea; 2017:644-654.

27. Shvaiko P, Euzenat J. Ontology matching: state of the art and future challenges. *IEEE Trans Knowl Data Eng*. 2013;25(1):158-176.
28. Otero-Cerdeira L, Rodríguez-Martínez FJ, Gómez-Rodríguez A. Ontology matching: a literature review. *Expert Syst Appl*. 2015;42(2):949-971.
29. Abubakar M, Hamdan H, Mustapha N, Aris TNM. Instance-based ontology matching: a literature review. Paper presented at: Proceedings of the International Conference on Soft Computing and Data Mining. Johor, Malaysia; 2018:455-469.
30. Nentwig M, Hartung M, Ngonga NA-C, Rahm E. A survey of current link discovery frameworks. *Semantic Web*. 2017;8(3):419-436.
31. Li J, Wang Z, Zhang X, Tang J. Large scale instance matching via multiple indexes and candidate selection. *Knowl-Based Syst*. 2013;50:112-120.
32. Wang Z, Li J, Zhao Y, Setchi R, Tang J. A unified approach to matching semantic data on the web. *Knowl-Based Syst*. 2013;39:173-184.
33. Alam M, Recupero DR, Mongiovi M, Gangemi A, Ristoski P. Event-based knowledge reconciliation using frame embeddings and frame similarity. *Knowl-Based Syst*. 2017;135:192-203.
34. Rosaci D. CILIOS: connectionist inductive learning and inter-ontology similarities for recommending information agents. *Inf Syst*. 2007;32(6):793-825.
35. Rosaci D. Finding semantic associations in hierarchically structured groups of web data. *Form Asp Comput*. 2015;27(5-6):867-884.
36. Elmagarmid AK, Ipeirotis PG, Verykios VS. Duplicate record detection: a survey. *IEEE Trans Knowl Data Eng*. 2007;19(1):1-16.
37. Ochieng P, Kyanda S. A K-way spectral partitioning of an ontology for ontology matching. *Distrib Parallel Databases*. 2018;36:643-673.
38. Tran Q-V, Ichise R, Ho B-Q. Cluster-based similarity aggregation for ontology matching. *Ontol Matching*. 2011;814:142-147.
39. Algergawy A, Massmann S, Rahm E. A clustering-based approach for large-scale ontology matching. Paper presented at: Proceedings of the East European Conference on Advances in Databases and Information Systems. Vienna, Austria; 2011:415-428.
40. Bellini P, Benigni M, Billero R, Nesi P, Rauch N. Km4City ontology building vs data harvesting and cleaning for smart-city services. *J Vis Lang Comput*. 2014;25(6):827-839.
41. Qiu J, Chai Y, Liu Y, Gu Z, Li S, Tian Z. Automatic non-taxonomic relation extraction from big data in smart city. *IEEE Access*. 2018;6:74854-74864.
42. Le-Phuoc D, Quoc HNM, Quoc HN, Nhat TT, HM. The graph of things: a step towards the live knowledge graph of connected things. *J Web Semant*. 2016;37:25-35.
43. Qiu J, Chai Y, Tian Z, Du X, Guizani M. Automatic concept extraction based on semantic graphs from big data in smart city. *IEEE Trans Comput Soc Syst*. 2019;7(1):225-233.
44. Mohammadi M, Hofman W, Tan Y-H. A comparative study of ontology matching systems via inferential statistics. *IEEE Trans Knowl Data Eng*. 2019;31(4):615-628.
45. Mohammadi M, Atashin AA, Hofman W, Tan Y. Comparison of ontology alignment systems across single matching task via the McNemar test. *ACM Trans Knowl Discov Data*. 2018;12(4):51.
46. Heflin J, Song D. Ontology instance linking: towards interlinked knowledge graphs. Paper presented at: Proceedings of the AAAI Conference on Artificial Intelligence. Phoenix, Arizona USA; 2016:4163-4169.
47. Jean-Mary YR, Shironoshita EP, Kabuka MR. Ontology matching with semantic verification. *Web Semant Sci Serv Agents World Wide Web*. 2009;7(3):235-251.
48. Saïs F, Pernelle N, Rousset MC. Combining a logical and a numerical method for data reconciliation. *J Data Semant*. 2009;XII:66-94.
49. Wang Z, Zhang X, Hou L, et al. RiMOM results for OAEI 2010. *Ontol Matching*. 2010;689:195-202.
50. Noessner J, Niepert M, Meilicke C, Stuckenschmidt H. Leveraging terminological structure for object reconciliation. Paper presented at: Proceedings of the Extended Semantic Web Conference. Heraklion, Crete, Greece; 2010:334-348.
51. Suchanek FM, Abiteboul S, Senellart P. Paris: probabilistic alignment of relations, instances, and schema. *Proc VLDB Endow*. 2011;5(3):157-168.
52. Song D, Heflin J. Domain-independent entity coreference for linking ontology instances. *J Data Inf Qual*. 2013;4(2):7.

53. Lacoste-Julien S, Palla K, Davies A, Kasneci G, Graepel T, Ghahramani Z. Sigma: Simple greedy matching for aligning large knowledge bases. Paper presented at: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Chicago, USA; 2013:572-580.
54. Xue X, Wang Y. Using memetic algorithm for instance coreference resolution. *IEEE Trans Knowl Data Eng*. 2016;28(2):580-591.
55. Xiao C, Wang W, Lin X. Ed-join: an efficient algorithm for similarity joins with edit distance constraints. *Proc VLDB Endow*. 2008;1(1):933-944.
56. Song D, Heflin J. Automatically generating data linkages using a domain-independent candidate selection approach. Paper presented at: Proceedings of the International Semantic Web Conference. Bonn, Germany; 2011:649-664.
57. Xiao C, Wang W, Lin X, Yu JX, Wang G. Efficient similarity joins for near-duplicate detection. *ACM Trans Database Syst*. 2011;36(3):15.
58. Wang J, Li G, Feng J. Extending string similarity join to tolerant fuzzy token matching. *ACM Trans Database Syst*. 2014;39(1):7.
59. Xue X, Liu J. A compact hybrid evolutionary algorithm for large scale instance matching in linked open data cloud. *Int J Artif Intell Tools*. 2017;26(04):1750013.
60. Xue X, Chen J, Chen J, Chen D. Using compact coevolutionary algorithm for matching biomedical ontologies. *Comput Intell Neurosci*. 2018;2018:2309587.
61. Djenouri Y, Djamel D, Djenoouri Z. Data-mining-based decomposition for solving MAXSAT problem: towards a new approach. *IEEE Intell Syst*. 2017;32(4):48-58.
62. Agrawal R, Imieliński T, Swami A. Mining association rules between sets of items in large databases. *ACM SIGMOD Rec*. 1993;22(2):207-216.
63. Brin S, Motwani R, Ullman JD, Tsur S. Dynamic itemset counting and implication rules for market basket data. *ACM SIGMOD Rec*. 1997;26(2):255-264.
64. Han J, Pei J, Yin Y. Mining frequent patterns without candidate generation. *ACM SIGMOD Rec*. 2000;29(2):1-12.
65. Barron A, Rissanen J, Yu B. The minimum description length principle in coding and modeling. *IEEE Trans Inf Theory*. 1998;44(6):2743-2760.
66. Gouda K, Zaki MJ. Efficiently mining maximal frequent itemsets. Paper presented at: Proceedings of the International Conference on Data Mining. San Jose, CA, USA; 2001:163-170.
67. Pei J, Han J, Mao R. Closet: an efficient algorithm for mining frequent closed itemsets. Paper presented at: Proceedings of the ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery. Boston Massachusetts, USA; Vol 4, 2000:21-30.
68. Hosseini S, Kalam S, Barker K, Ramirez-Marquez JE. Scheduling multi-component maintenance with a greedy heuristic local search algorithm. *Soft Comput*. 2020;24(1):351-366.
69. Djenouri Y, Drias H, Bendjoudi A. Pruning irrelevant association rules using knowledge mining. *Int J Business Intell Data Mining*. 2014;9(2):112-144.
70. Djenouri Y, Chun-Wei LJ, Nørvåg K, Ramampiaro H. Highly efficient pattern mining based on transaction decomposition. Paper presented at: Proceedings of the IEEE International Conference on Data Engineering. Macao, China; 2019:1646-1649.
71. Chun-Wei LJ, Shao Y, Fournier-Viger P, Djenouri Y, Guo X. Maintenance algorithm for high average-utility itemsets with transaction deletion. *Appl Intell*. 2018;48(10):3691-3706.
72. Mittal S, Vetter JS. A survey of CPU-GPU heterogeneous computing techniques. *ACM Comput Surv (CSUR)*. 2015;47(4):69.
73. Cano A. A survey on graphic processing unit computing for large-scale data mining. *Wiley Interdiscip Rev Data Mining Knowl Discov*. 2018;8(1):e1232.
74. Djenouri Y, Djenouri D, Belhadi A, Fournier-Viger P, Chun-Wei LJ, Bendjoudi A. Exploiting GPU parallelism in improving bees swarm optimization for mining big transactional databases. *Inf Sci*. 2019;496:326-342.

How to cite this article: Djenouri Y, Belhadi H, Akli-Astouati K, Cano A, Lin JC-W. An ontology matching approach for semantic modeling: A case study in smart cities. *Computational Intelligence*. 2021;1–27. https://doi.org/10.1111/coin.12474