International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Ingénierie des Systèmes d'Information
Vol. 29, No. 1, February, 2024, pp. 57-64
Journal homepage: http://iieta.org/journals/isi

A Hidden Markov Model-Based Approach for Lightweight Ontology Modularization Using K-Means Clustering

Lazarre Warda[1*], Soraya Setti Ahmed[2], Oumarou Hayatou[3], Guidedi Kaladzavi[4], Amaria Samdalle[1], Kolyang[3]

[1] Department of Mathematics and Computer Science, Faculty of Sciences, The University of Maroua, Maroua 814, Republic of Cameroon
[2] Department of Computer Science, Faculty of Exact Sciences, Mascara University, Mascara 29000, Algeria
[3] Department of Computer Science, Higher Teachers' Training College, The University of Maroua, Maroua 55, Republic of Cameroon
[4] Department of Computer Science and Telecommunications, National Advanced School of Engineering, The University of Maroua, Maroua 58, Republic of Cameroon

Corresponding Author Email: lzwarda2015@gmail.com

https://doi.org/10.18280/isi.290107

Received: 4 May 2023
Revised: 26 August 2023
Accepted: 11 October 2023
Available online: 27 February 2024

Keywords:
semantic web, ontology, modularization, hidden Markov model, RDF triple, ontology module, K-means clustering

ABSTRACT

Nowadays, ontologies are backbone of Semantic Web. Several domains use ontologies as knowledge models. As their number is constantly increasing, designers are opting to reuse some of those that exist to build new ones. When it is impossible to reuse a part depending on its organization, they import the whole ontology and this makes the manipulation cumbersome especially if the ontology has large concepts. Therefore, segmenting ontologies into partitions, if they are not yet, becomes a constant challenge for designers. This paper presents an approach to modularize ontology using hidden Markov model. Ontology triples are extracted within ontology through SPARQL queries and labelled with integers. The labelled triples constituted a Markov chain where ontology concepts are states and ontology relationships are symbols. This set is used to initialize HMM parameters such as states transition probabilities and symbols observation probabilities matrix and initial states probabilities vector. The transition probabilities matrix of HMM is then used as input of K-Means algorithm to generated modules of ontology concepts. This approach does not handled ontology axioms, which characterize heavy ontologies, and only lightweight ontologies are considered. Experiment on eighteen ontologies, obtained modules satisfied ontology modularization criteria such as independence, non-redundancy, correctness and completeness.

1. INTRODUCTION

Ontologies are considered as the heart of the architecture of the semantic web and among the use of ontologies we have resource annotations (documents, images, videos, etc.). Formally, Gruber defined ontology as "Ontology is an explicit specification of a conceptualization" [1]. This definition was completed by Borst as "Ontology is an explicit and a formal specification of a shared conceptualization" [2]. With these definitions, we can outline that ontology is a set of concepts and axioms which describe certain domain knowledge. Hence, ontology contents are: classes or concepts, relationships (properties), instances (individuals) and axioms. Concepts represent a set of entity classes within the domain. Relationships specify the interaction among classes. Instances indicate the concrete examples of classes within the domain and axioms denote statements which are always true [3]. The construction of these ontologies follows a life cycle whose essential points are specification, conception, implementation, validation and maintenance. Validation step requires reasoning on the knowledge represented and maintenance step needs a probable evolution. Most of ontologies thus built are monolithic, that is to say, the concepts of these ontologies are in a single block and one concept can be linked to any concept. With this type of ontologies, especially if they contain several concepts (large) or are complex in their organization, it is sometimes difficult to ensure scalability or to conduct efficient reasoning. With a plethora of ontologies for the same domain, the authors opt for a reuse of part of ontology to keep persistent definitions of the concepts, which is not easy with monolithic ontologies, if so it is necessary to reuse all the ontology. To deal with these difficulties on monolithic ontologies, the authors opt for modular ontologies either by composition (building small independent partitions of ontologies and merging them) or by decomposition (partitioning monolithic ontologies into coherent modules) [4]. Regardless of the axis chosen, it must follow the goals concerning scalability for querying data and efficient reasoning on ontologies, scalability for evolution and maintenance, complexity management, understandability, context-awareness, personalization and reuse. These goals are more described in studies [4-6] and maintain by evaluation criteria outlined in the study of d'Aquin et al. [7], such as correctness, completeness, connectedness, module cohesion, richness of representation, which are applied for technique validation.

In the literature, we distinguish five categories of ontology modularization techniques subdivided into two groups according to the chosen axis. The axis of composition of ontology modules includes Distributed Description Logics, ε-Connexions, Package-based Descriptive Logics and the Conservative Extensions and for the axis of decomposition we have Graph-based Ontology Segmentation [4]. In the field of Graph-based Ontology Segmentation, five approaches [8-12] have been illustrated and the best known are PATO [8] and OAPT [11] which are interested in the partitioning of hierarchically organized concepts. However, these ontology segmentation approaches come up against limits, in particular the redundancy of ontology concepts in the modules, the manual assignation of isolated concepts to the modules, the merging of modules whose number of elements does not reach the fixed threshold and sometimes lack of semantics.

Since managing knowledge represented by ontologies using machine learning tools becomes increase, authors introduced possibility to capture ontology characteristics through hidden Markov Model (HMM) [13] and proposed in the study of Warda et al. [14] an approach to turn ontology into HMM based on ontology triples produced by SPARQL queries on it.

As a result, this study aims to propose a method to modularize lightweight ontologies using machine learning technique more precisely the hidden Markov model. Ontology axioms are avoided in this approach and then heavy ontologies are not considered because they are characterized with axioms. However, if heavy ontologies are used, axioms are not handled. Ontology triples are extracted within ontology through SPARQL queries and are used to initialize a HMM parameters such as states transition probabilities distribution matrix, symbols observation probabilities distribution matrix and initial states probabilities distribution vector via some equations. Ontology concepts are considered as HMM states and ontology relationships as HMM symbols. Since partitioning ontology consists in dividing its concepts into partitions, only the state transition probability distribution matrix of the HMM will be used as input for the K-Means algorithm to compute ontology modules. The rest of this paper is organized as follow: state of the art regarding HMM definition and related works are described in Section 2. The proposed method is detailed in Section 3. Section 4 focuses on experimental findings and discussion. The last Section is devoted to the conclusion and potential future trends and challenges.

2. STATE OF THE ART

2.1 Hidden Markov Model (HMM)

A Markov model is a stochastic phenomenon. This model verifies a certain number of properties such as: the model changes state at determined instants of time (the space of time is discrete) and the Markovian property (the current state of the model depends only on the last known state). HMM expression is used when the states of Markov model are hidden. HMMs were introduced in the 1960s - 1970s by Baum and his collaborators [15]. The formal definition of a Hidden Markov Model (HMM) denoted as λ, which consists of the set {N, B, A, B, π}, is provided by Warda et al. [14].

(1) N is the number of HMM states, we note $\mathrm{S} = \{\mathrm{S}_1, \mathrm{S}_2, \dots, \mathrm{S}_\mathrm{N}\}$ the set of states;

(2) M is the number of HMM symbols, we note $V = \{v_1, v_2, \dots, v_M\}$ the set of symbols;

(3) $A = [a_{ij}], 1 \leq i, j \leq N$, is the states' transition probabilities distribution matrix of the model;

(4) $B = [b_j(k)], 1 \leq k \leq M$, is the observation symbols probabilities distribution matrix of the model;

(5) $\pi = [\pi_i], 1 \leq i \leq N$, is the initial states probabilities distribution of the model.

HMMs are models used in several areas of daily life including speech processing, handwritten text recognition, biological sequence analysis, image recognition, medical signal modelling and many others. They solve three main types of problems: evaluation, decoding and learning. The evaluation problem consists in calculating the probability of observing a sequence knowing the model and is solved with the Forward-Backward algorithm. The decoding problem is to determine the best possible sequence of states from the model. This optimal sequence is obtained with the Viterbi algorithm. As for the learning problem, it consists in improving the parameters A, B and π of the model with respect to a given sequence of observations and this is done with the Baum-Welch algorithm [16]. HMMs have been used either to populate ontologies or combine them together to build systems [14]. Indeed, HMMs are models that preserve the semantics between elements and their graphical representation brings them closer to ontology, which would widely contribute to the modularization of ontologies.

2.2 Related works

The first reference approach was introduced in 2006 by Schlicht and Stuckenschmidt [8]. They proposed PATO, an approach for large hierarchical concepts partitioning. This approach is based on tree main steps. In the first step, they created a dependency graph extracted from ontology file. The second step concerned the determination of dependencies strength between the concepts in the graph and in the last step, they detected sets of concepts which formed modules. These principal steps are followed by two additional steps. The fourth step is reserved to the assignation of isolated concepts to the appropriated existed module and in the fifth step, for some modules where the size (obtained by some formulas) is under a threshold, they are merged into adjacent module with a lower size. This approach is the commonly used by authors for ontology modularization problems. Guided by some properties of modules, Ensan and Du [17, 18] proposed a framework to develop modular ontology through interfaces-based formalism. To achieve their goal, they supposed that a modular ontology is a set of independent modules which are interfaces-based joined. Designers are free to develop each module independently to each other's language and signature and modules can cover separate domain knowledge. Similarly, Doran [19] began by revisiting principles of ontology modularization and discussed about some of techniques. Then he proposed an approach to extract ontology module. To achieve this goal, firstly, he defined competency of module to be extract. Secondly he selected target ontology based on ontology evaluation. At the end he selected appropriate module. Based on ontology organization, Özacar et al. [20] proposed Anemone, a methodology for modular ontology development. The modular ontology obtained with this methodology behaves like a monolithic ontology which is organized into modules classed into layers. The bottom layer module (small ontology) imports the directly above layer module. Another approach reposed in classification-based learning was proposed by Alaya et al. [9]. They defined five steps. In the first step, they used hierarchical classification to partitioning ontology concepts by identifying their dependences. In the second step, they enriched each partition resulted in the previous step by adding appropriated axioms and others properties for the whole ontology to be modularized. When there is redundancy between modules (partitions) according to a certain threshold, in the third step, these modules are fused. In the fourth step, they update modules by adding entities of whole ontology which are not treat during previous steps. In the last step, they mapped modules to established semantic links among them. In the study of Ahmed et al. [10], a graph partitioning approach was proposed by Soraya et al. This methodology turned around five steps. At the first step, OWL constructors (classes, relationships, properties, axioms) are extracted within the ontology. The second step concerned the creation of Dijkstra-based algorithm level graph. They compute similarity measures (six measures) between classes in the third step. Based on these similarity measures, the fourth step is devoted to the clustering process using K-Means algorithm with some adjustments. The last step concerned the validation of the approach. To deal with the limits of previous approaches, Alsayed et al. proposed OAPT [11], an Ontology Analysis and Partitioning Tool, which regrouped five components. In the pre-processing component, they checked the input format of ontology and if it is necessary, they parsed and represented it into concept graph. The analysis component concerned the determination of ontology design metrics referred to structural, semantic and syntactic categories. These metrics helped them to evaluate the richness of the ontology. The next components referred to the modularization and determination of optimal number of modules. To do it, they began by ranking each concept, determined the heads of clusters and then performed partitions using seed-based algorithm and finished by generating modules. In the last component, they evaluated the methodology. Guided by the plethora number of ontologies in BioPortal, Alsayed and Birgitta [12], experimented the possibility for partitioning these ontologies. These experiments are based in four steps. Firstly, they used BioPortal to get all accessible ontologies and transformed them into OWL or OBO formats using OWL API in the second step. In the third step, they partitioned these ontologies through PATO, OAPT and AD algorithms before analysing these results in the last step. Recently, Shimizu et al. widely contributed to ontology modularization by developing MODL in the study of Shimizu et al. [21], a modular ontology design library. It is a collection of documented ODP (ontology design patterns). An ODP is a small and reusable set of concepts and axioms which solve an invariant problem in various domains. They collected five categories of ODP and documented them. Based on these components, they developed again CoModIDE in the study of Shimizu and Hammar [22, 23], a Comprehensive Modular Ontology Engineering IDE, which is a plugging for *Protégé*. It is composition of ODP. Le Clair and Khedri [24] developed algebraic technique based on logical technique to propose a modularization approach. The modules derived from set of Boolean sub-algebras which covered evaluation metrics for ontology modularization techniques: local correctness and local completeness. Finally, Shimizu et al. [23], proposed a method which automated the type-based generation of abstraction modules. They created five algorithms associated to the five defined types of abstraction: axiom abstraction, vocabulary abstraction, high-level abstraction, weighted abstraction and feature expressiveness. The approaches here mentioned are not the only ones but the most cited in the literature. Nevertheless, others are cited in the study of d'Aquin et al. [7, 25-27]. Among these approaches hereinbefore cited, only those described by Schlicht and Stuckenschmidt [8-12] concerned the top-down method which refers to the splitting up of ontology into modules. One of the common limits of these approaches is the manual assignation of isolated concepts to modules. At the end of steps, on the one hand, it happens that there are some concepts that remain isolated [8-10, 12]. They therefore assigned to modules according to the defined criterion. On the other hand, the number of concepts in some modules does not reach a fixed threshold. A module in this condition is systematically merged with the one that is close [8, 9, 12]. Among the modularization criteria, the non-redundancy of concepts is a determining factor, yet this criterion is not fully taken into account in the approaches proposed by Algergawy et al. [11, 12] because sometimes concepts are present in several modules. The last point that the OAPT approach comes up against is the loss of semantics [11]. The approach described here overcomes these limitations. Indeed, the ontology triples extracted from the ontology guarantee the semantics of the ontology and the incidence matrix corresponding to the state transition probability distribution matrix of the HMM highlights the different relationships between the concepts. The application of the K-means algorithm leads to the disjunction of the modules and any assignation of the concepts to the modules is automatically done. The modules obtained are formed according to the relationships that exist between the different concepts and the size of the modules is consequently linked. Table 1 summarized and outlined these limits.

Table 1. Decomposition-based approaches and their limits

Approaches	Main Idea	MAC	FMM	R	LST
[8]	Partitioning of large hierarchical concepts (PATO)	Yes	Yes	No	No
[9]	Classification-based learning	Yes	Yes	No	No
[10]	Partitioning ontology due adapted K-Means Algorithm	Yes	No	No	Yes
[11]	Partitioning ontology based on the seeding-based scheme (OAPT)	No	No	Yes	No
[12]	Portioning of BioPortal ontologies	Yes	Yes	Yes	No

(MAC: Manual assignation of concepts, FMM: Fusion or merging modules, R: Redundancy, LST: Lack of semantics from triples)

3. PROPOSED APPROACH

The Figure 1 describes the steps of this approach.

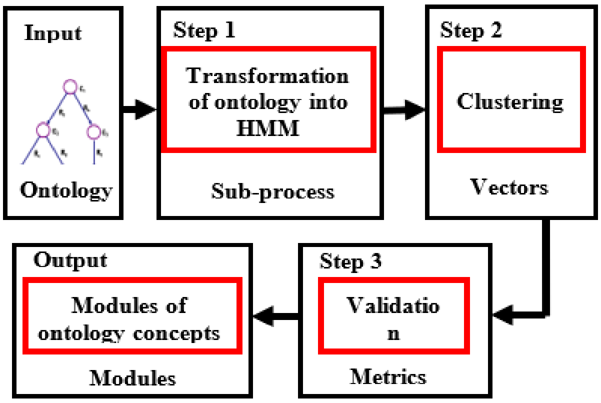

Figure 1. HMM-based process of ontology partitioning

3.1 Step 1: Transformation of ontology into HMM

Ontology can be seen as a set of triples. A triple is a *(subject, predicate, object)* set where *subject* and *object* are classes; *predicate* denotes the relationship between *subject* and *object*. Various methods for learning ontology properties using HMMs have been discussed in the study by Warda et al. [14]. The following step is adapted from their findings [14]:

(1) Firstly, they extracted ontology triples. These triples derived from SPARQL queries. Triples can be filter according to the type of predicate.

(2) Secondly, they labelled obtained triples to replace concepts and predicates names (string type) by numbers to facilitate manipulations.

(3) Thirdly, they initiated HMM parameters with the labelled triples using Eqs. (1)-(3).

$$a_{ij} = \frac{\text{number of triples (i = subject and j = object)}}{\text{number of triples (i = subject) + } \varepsilon} \qquad (1)$$

In Eq. (1), a_{ij} is the ratio of the number of ontology triples where concept *i* is the subject and concept *j* is the object by the number of ontology triples having concept *i* as the subject. This ratio corresponds to the probability that there are relations between the concepts numbered respectively with *i* and *j* consequently it is the probability that there is transition moving from the states *i* to *j* of the HMM.

$$b_j(k) = \frac{\text{number of triples (j = subject and k = predicate)}}{\text{number of triples (k = predicate) + } \varepsilon} \qquad (2)$$

Similarly, in Eq. (2), $b_j(k)$ corresponds to the ratio of the number of ontology triples where concept *j* is the subject and relationship *k* is the predicate. This ratio is the probability that there are relations resulting from the concept numbered by *j* consequently it is the probability of observing the symbol *k* in state *i* of the HMM.

$$\pi_i = \frac{\text{number of triples (i = subject)}}{\text{total number of triples + } \varepsilon} \qquad (3)$$

In the same manner, in Eq. (3), $\boldsymbol{\pi_i}$ corresponds to the ratio of the number of ontology triples where concept *i* is the subject by the total number of ontology triples. This ratio is the probability that the concept numbered with *i* is the subject of the triples and consequently it is the probability that *i* is an initial state of the HMM.

Since a_{ij}, $b_j(k)$ andπ_i are probabilities distributions, by adding ε to the denominator of Eqs. (1)-(3), probabilities laws are not satisfied. Hence the rest value to reach 1 is equitably redistributed to each of them.

All these steps are more detailed in the study of Warda et al. [14] and this approach has been used here to turn ontology into HMM.

3.2 Step 2: Clustering

In this step, states transition matrix (A) of obtained HMM is used as a set of vectors in which the clustering algorithm should be applied. The following configurations are considered:

(1) The transition matrix (A) of the obtained HMM is divided into set of vectors: $A = [A_j], 1 \leq j \leq N$, where $A_j = [a_{ij}], 1 \leq i \leq N$, is a vector.

(2) The set of states, $S = \{S_1, S_2, \dots, S_N\}$, is equivalent to the set of ontology concepts coming from the SPARQL queries. States are independent because associate concepts are different.

(3) For each S_j, $1 \leq j \leq N$, component of *S*, we can obtain all the transition probabilities to other states. These probabilities correspond to the vector A_j of transition matrix. Hence $\langle S_1, S_2, \dots, S_N \rangle$ is considered as a basis where each vector A_j can be expressed.

(4) Clustering algorithm is then applied and vectors to be partitioned are $A = [A_j], 1 \leq j \leq N$. Appropriate distance measures and number of clusters can be defined.

Among the clustering algorithms described by Xu et al. [28, 29], K-Means is the most widely used in the literature for ontology segmentation. The organization of the data (concepts) to be partitioned represented by the different vectors from the HMM incidence matrix (A) conforms to the K-Means input. We could not apply the hierarchical grouping because in addition to the hierarchical organization, the ontology concepts are linked by other relations like *object Property* type. So our focus was on K-Means and distance measures can be changed to analyse the behaviour of the results. Several distance measures can be applied: Euclidean, Cosine similarity, Dice, etc. The optimal number of clusters is obtained by the Elbow method integrated to the K-Means algorithm [30].

3.3 Step 3: Validation

This step consists to validate the proposed approach. Several criteria come from software engineering and are outlined in d'Aquin et al. [7] and some of them can be applied:

Independence: the proposed approach partitions a set of concepts generated by ontology triples. The resultant clusters are independent according to K-Means algorithm.

Local correctness and local completeness: all concepts inside each module and its signature generated by this approach belonged to the huge used ontology because the triples were generated within this ontology.

Size: Since Elbow method is used to determine the number of clusters, the size of each module is consequently related to the organization of target ontology.

Intra-module distance and connectedness: since the components a_{xy} of vectors are probabilities distributions, if this value rich 1, it means that class represented by x and class represented by y are strongly linked with a certain predicate (hierarchical link or object property). To ensure this validation metric, we defined a value of module distance:

$$W_M = \frac{1}{m}\sum_{x,y\in M} a_{xy} \quad (4)$$

In this formula, *M* is the module which the intra-module is computed and *m* is the number of elements in module *M*. If W_M reach 1 then the module is coherent. At the end, for all modules, we defined a weight *W* which refers to the average of their intra-module distances:

$$W = \sum_{k=1}^{K} W_k \quad (5)$$

where, *K* is the number of modules obtained by the Elbow method before partitioning huge ontology. This value gives the degree of the modules quality.

Non-redundancy: intrinsically, this criterion is ensured by the proposed approach. All modules are disjoined according to the description of this methodology.

4. EXPERIMENTAL RESULTS

4.1 Experiments on ontologies

To experiment this approach, we used eighteen ontologies freely download on BioPortal website (https://bioportal.bioontology.org/ontologies). The SPARQL queries used for these experiments are:

Table 2. Queries used for experiments

Query 1	Query 2

SELECT **?s ?o**
WHERE{
?s rdfs:subClassOf **?o** .
?s a owl:Class .
?o a owl:Class .
}
	SELECT **?s ?p ?o**
	WHERE{ **?s** a owl:Class .
	?o a owl:Class .
	?p a owl:ObjectProperty .
	?p rdfs:domain **?s** .
	?p rdfs:range **?o** .
	}

In Table 2, Query 1 extracts ontology triples where the predicate is the *subclass* relationship, the subject is subclass and the object is the associate parent class. In other word, this query brings out the hierarchical relationship among ontology concepts. Query 2 extracts ontology triples where the predicate is an *object property* relationship, the subject is the domain of this object property and the object is its range. There are not only these queries that we can used. Someone can add own queries and produces the associated triples. It is important to note that the number of relationships between ontology classes influences the quality of ontology modules. The higher number of relationships means better quality of modules and if this number rich 1, the ontology is seen like a hierarchical classification of concepts.

For these experiments, two distance measures were used for the K-Means algorithm: Euclidean Distance and Cosine Similarity.

The numbers of modules (obtained with Elbow method), concepts per module and intra-module distances are summarized in Table 3 (for Cosine Similarity) and Table 4 (for Euclidean Distance). These tables show also the numbers of ontology classes and relationships among these classes derived from the triples obtained with the above queries. The number of modules can change according to the distance measure chosen. For instance, for *OntoRepliCov* ontology, with Euclidean Distance, the number of partitions is 4 and it is 6 with Cosine Similarity. User can directly define the number of partitions however the results cannot satisfy modularization criteria. During the validation step, it is easy to compare results related to the number of relationships within the ontology based only on module size criterion. Qualities of obtained modules for this experiment are strongly influenced by the expansion of hierarchical organization of these ontologies. Thus, the *subclass* relationship among classes subdued others relationships or properties, hence influenced the number of concepts per module.

The comparative study of Tables 3 and 4 shows that three ontologies (1.5 Covid19-IBO, Bspo and OntoRepliCov) have higher module numbers with Cosine Similarity than with Euclidean Distance. The choice of the number of partitions for a clustering algorithm often causes a problem because a large number K can lead to too fragmented partitioning, limiting the discovery of interesting data patterns; on the other hand, a number K that is too small potentially leads to too general clusters containing a lot of data. Thus, the results obtained for the proposed modularization approach with the Euclidean Distance are better than those obtained with the Cosine Similarity because the number of clusters obtained gives intra-module distances very close to 1 and then ensure modules quality.

Browsing Table 3 and Table 4, the number of modules and the number of concepts per module for each ontology are the same for some ontologies and for the two tables, sometimes, the first module has the high number of ontology classes (1.5 Covid19-IBO, Biotope, Bspo …). This situation consequently depends to the influence of *subclass* relationship in these ontologies. To deal with this situation, user can avoid this relationship and consider only *object properties* if they can be sufficient to extract the most quantity of ontology concepts through SPARQL queries. Furthermore, ontologies with small number of concepts are not strongly influenced with *subclass* relationship (e.g. OntoRepliCov, Syndromes, Hom).

A comparison is taken between two ontologies: Hio (495 classes) and Hom (65 classes). Each of these two ontologies has only one relationship: *subclass* relationship. In the case of Hio, distribution of concepts among modules follows level distributions of concepts in ontology whereas this situation is not the same in the case of Hom. Although this relationship has influenced the distributions of ontology concepts among modules, the weights of modules show that intra-module distance defined hereinbefore are acceptable.

4.2 Discussion

The contribution of modularization is not to be demonstrated in the field of ontology management. Furthermore, HMM is a machine learning tool which ensures links among events of elements. Thus learning knowledge within ontology with HMM can help applications and users to more precisely handle deservedly this knowledge. The approach presented here attempts to overcome previous papers limits concerning redundancy, manual assignation of ontology concepts to partitions and sometimes lack of semantics. Using ontology triples extracted within ontologies, the semantics is preserved if and only if optimal queries are used. On the one hand, the reminder is that axioms, which are the restrictions on concepts, are not handled and if the richness of ontology reposes in its axioms, then all the semantics will not be took into account.

On the other hand, as is mentioned hereinbefore, the number of relationships among ontology concepts influences the quality of modules by biasing them. Experiencing this approach on some freely download ontologies, we realized that the *subclass* relationship which describes hierarchical organization of ontology concepts has real impact on results.

To attempt to overcome these two situations which can constitute the drawbacks of this approach, some dispositions can be taken to ameliorate it: (1) adding another types of relations such as *data properties* or OWL constructors such as *equivalence class* to reduce the number of ontology concepts before turning it into HMM, (2) axioms can be handled when extracting ontology triples with queries by splitting the object as single concept although the axioms also come to increase the number of triples having the relation of *subclass* like predicate and (3) *subclass* relationship can be avoided when others relations can be sufficient to extract the maximum of ontology concepts with queries.

Table 3. Number of classes, relationships and classes per modules for ontologies with Cosine Similarity

No.	Ontologies	C	R	Number of Modules	Concepts per Module						W
					1st	2nd	3rd	4th	5th	6th	
1	1.5 Covid19-IBO	159	34	5	129	9	6	10	5	/	0.88
2	Biotope	337	22	4	300	9	18	10	/	/	0.97
3	Bspo	172	30	6	97	23	20	14	12	6	0.69
4	Cvdo	290	42	5	29	22	205	8	26	/	0.84
5	FishOntology	385	14	6	314	12	21	18	12	8	1.00
6	Hio	495	1	4	446	26	12	11	/	/	1.00
7	Hom	65	1	5	8	13	5	35	4	/	0.46
8	Htn	600	65	4	523	27	36	14	/	/	0.94
9	InBiOn	361	10	4	305	16	22	18	/	/	0.99
10	Ontobio_01072013	185	23	5	138	9	7	23	8	/	0.89
11	OntoFood	289	19	3	256	22	11	/	/	/	0.97
12	OntoPBM	177	2	4	134	22	7	14	/	/	0.95
13	OntoRepliCov	85	10	6	6	15	22	18	16	8	2.17
14	Pco	219	32	3	6	16	197	/	/	/	1.00
15	PlantDiversityOntology	380	22	5	256	25	46	25	19	/	1.00
16	Ppo	443	14	2	56	387	/	/	/	/	0.86
17	Syndromes	171	12	4	42	88	24	17	/	/	0.74
18	vio_merged	79	24	2	64	15	/	/	/	/	0.95

(C: number of classes, R: number of relationships, W: average intra-module distance)

Table 4. Number of classes, relationships and classes per modules for ontologies with Euclidean Distance

No.	Ontologies	C	R	Number of Modules	Concepts per Module						W
					1st	2nd	3rd	4th	5th	6th	
1	1.5 Covid19-IBO	159	34	4	137	9	10	3	/	/	0.97
2	Biotope	337	22	4	300	9	18	10	/	/	0.97
3	Bspo	172	30	4	115	14	20	23	/	/	0.74
4	Cvdo	290	42	5	29	22	205	8	26	/	0.84
5	FishOntology	385	14	6	314	12	21	18	12	8	1.00
6	Hio	495	1	4	446	26	12	11	/	/	1.00
7	Hom	65	1	5	8	13	5	35	4	/	0.46
8	Htn	600	65	4	523	27	36	14	/	/	0.94
9	InBiOn	361	10	4	305	16	22	18	/	/	0.99
10	Ontobio_01072013	185	23	5	138	9	7	23	8	/	0.90
11	OntoFood	289	19	3	256	22	11	/	/	/	0.98
12	OntoPBM	177	2	4	134	22	7	14	/	/	0.95
13	OntoRepliCov	85	10	4	34	15	22	14	/	/	0.94
14	Pco	219	32	3	6	16	197	/	/	/	1.00
15	PlantDiversityOntology	380	22	5	265	25	46	25	/	/	1.00
16	Ppo	443	14	2	55	388	/	/	/	/	0.87
17	Syndromes	171	12	4	42	88	24	17	/	/	0.74
18	vio_merged	79	24	2	64	15	/	/	/	/	0.95

(C: number of classes, R: number of relationships, W: average intra-module distance)

5. CONCLUSION

In this paper, we proposed a HMM-based method for ontology modularization. It is a partitioning approach which handles lightweight ontologies. This technique used ontology triples extracted within ontology using defined SPARQL queries to initialise a HMM parameters for capturing knowledge stored in ontology. A parameter of this HMM (transition states probabilities distribution matrix) is used as vectors for input of a clustering algorithm – K-Means algorithm – to generated ontology modules. Modules generated overcome drawbacks of existing approaches such as automatic assignation of all ontology concepts to target module, avoiding redundancy of ontology concepts, module size and independence, which are some criteria of ontology modularization techniques. With experiments, we noted that the clustering using Euclidean distance gave better results that once using cosine similarity and *subclass* relationship among ontology concepts can influence quality and size of generated modules and then alter this technique. Future trends will explore the possibilities to overcome mentioned limitations by taking into account axioms and others types of relationships among concepts when extracting triples within complex ontologies.

REFERENCES

[1] Gruber, R.T. (1993). A translation approach to portable ontology specifications. Knowledge Systems Laboratory, 5(2): 199-220. https://doi.org/10.1006/knac.1993.1008

[2] Brost, W.N. (1997). Construction of engineering ontology for knowledge sharing and reuse. Phd thesis, University of Twente, Enschede.

[3] Subhashini, R., Akilandeswari, J. (2011). A survey on ontology construction methodologies. International Journal of Enterprise Computing and Business Systems, 1(1): 60-72.

[4] Pathak, J., Johnson, T.M., Chute, C.G. (2009). Survey of modular ontology techniques and their applications in the biomedical domain. Integrated Computer-Aided Engineering, 16(3): 225-242. https://doi.org/10.3233/ICA-2009-0315

[5] Abbes, S.B., Sheuermann, A., Meilender, T., d'Aquin, M. (2012). Characterizing modular ontologies. In 7th International Conference on Formal Ontologies in Information Systems-FOIS 2012, pp. 13-25.

[6] Parent, C., Spaccapietra, S. (2009). An overview of modularity. In: Stuckenschmidt, H., Parent, C., Spaccapietra, S. (eds) Modular Ontologies. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 5445: 5-23. https://doi.org/10.1007/978-3-642-01907-4_2

[7] d'Aquin, M., Schlicht, A., Stuckenschmidt, H., Sabou, M. (2009). Criteria and evaluation for ontology modularization techniques. Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization, 67-89. https://doi.org/10.1007/978-3-642-01907-4_4

[8] Schlicht, A., Stuckenschmidt, H. (2008). A flexible partitioning tool for large ontologies. In 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, Sydney, NSW, Australia, pp. 482-488. https://doi.org/10.1109/WIIAT.2008.398

[9] Alaya, N., Yahia, S.B., Lamolle, M. (2012). Modlson: Une nouvelle approche de modularisation d'ontologies à grande échelle. In EGC, pp. 279-284. https://www.researchgate.net/publication/277685144_MODLSON.

[10] Ahmed, S.S., Malki, M., Benslimane, S.M. (2015). Ontology partitioning: Clustering based approach. International Journal of Information Technology and Computer Science, 7(6): 1-11. https://doi.org/10.5815/ijitcs.2015.06.01

[11] Algergawy, A., Babalou, S., Klan, F., König-Ries, B. (2020). Ontology modularization with OAPT. Journal on Data Semantics, 9: 53-83. https://doi.org/10.1007/s13740-020-00114-7

[12] Algergawy, A., König-Ries, B. (2019). Partitioning of BioPortal ontologies: An empirical study. In SWAT4HCLS, pp. 84-93.

[13] Lazarre, W., Guidedi, K., Amaria, S., Kolyang. (2022). Modular ontology design: A state-of-art of diseases ontology modeling and possible issue. Revue d'Intelligence Artificielle, 36(3): 497-501. https://doi.org/10.18280/ria.360319

[14] Warda, L., Kaladzavi, G., Samdalle, A., Kolyang. (2022). Integration of ontology transformation into hidden Markov model. Information Dynamics and Applications, 1(1): 2-13. https://doi.org/10.56578/ida010102

[15] Rabiner, L., Juang, B. (1986). An introduction to hidden Markov models. IEEE ASSP Magazine, 3(1): 4-16. https://doi.org/10.1109/MASSP.1986.1165342

[16] Bréhélin, L., Gascuel, O. (2000). Modèles de Markov cachés et apprentissage de séquences. Le temps, l'espace et l'évolutif en sciences du traitement de l'information, Eds. Cépaduès.

[17] Ensan, F., Du, W. (2008). An interface-based ontology modularization framework for knowledge encapsulation. In International Semantic Web Conference, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 517-532. https://doi.org/10.1007/978-3-540-88564-1_33

[18] Ensan, F., Du, W. (2010). A modular approach to scalable ontology development. In: Du, W., Ensan, F. (eds) Canadian Semantic Web. Springer, Boston, MA, pp. 79-83. https://doi.org/10.1007/978-1-4419-7335-1_4

[19] Doran, P. (2009). Ontology modularization: Principles and practice. Doctoral dissertation, University of Liverpool.

[20] Özacar, T., Öztürk, Ö., Ünalır, M.O. (2011). ANEMONE: An environment for modular ontology development. Data & Knowledge Engineering, 70(6): 504-526. https://doi.org/10.1016/j.datak.2011.02.005

[21] Shimizu, C., Hirt, Q., Hitzler, P. (2019). MODL: A modular ontology design library. arXiv preprint arXiv:1904.05405. https://doi.org/10.48550/arXiv.1904.05405

[22] Shimizu, C., Hammar, K. (2019). Comodide–the comprehensive modular ontology engineering ide. In ISWC 2019 Satellite Tracks (Posters & Demonstrations, Industry, and Outrageous Ideas) co-located with 18th International Semantic Web Conference (ISWC 2019) Auckland, New Zealand, 2456: 249-252.

[23] Shimizu, C., Hammar, K., Hitzler, P. (2020). Modular graphical ontology engineering evaluated. In European Semantic Web Conference, Cham: Springer, pp. 20-35 https://doi.org/10.1007/978-3-030-49461-2_2

[24] Le Clair, A. (2021). A formal approach to ontology modularization and to the assessment of its related knowledge transformation. McMaster University Doctoral dissertation.

[25] Khan, Z., Keet, C.M. (2021). Structuring abstraction to achieve ontology modularisation. Advanced Concepts, Methods, and Applications in Semantic Computing, 21. https://doi.org/ 10.4018/978-1-7998-6697-8.ch004

[26] Bao, J., Caragea, D., Honavar, V.G. (2006). Modular

ontologies-A formal investigation of semantics and expressivity. In The Semantic Web–ASWC 2006: First Asian Semantic Web Conference, Beijing, China, pp. 616-631. https://doi.org/10.1007/11836025_60

[27] LeClair, A., Marinache, A., El Ghalayini, H., MacCaull, W., Khedri, R. (2022). A review on ontology modularization techniques-A multi-dimensional perspective. IEEE Transactions on Knowledge and Data Engineering, 35(5): 4376-4394. https://doi.org/10.1109/TKDE.2022.3152928

[28] Xu, R., Wunsch, D. (2005). Survey of clustering algorithms. IEEE Transactions on Neural Networks, 16(3): 645-678. https://doi.org/10.1109/TNN.2005.845141

[29] Xu, D., Tian, Y. (2015). A comprehensive survey of clustering algorithms. Annals of Data Science, 2: 165-193. https://doi.org/10.1007/s40745-015-0040-1

[30] Cui, M. (2020). Introduction to the K-Means clustering algorithm based on the elbow method. Accounting, Audition and Finance, 1(1): 5-8. https://doi.org/10.23977/accaf.2020.010102