Arbitrary Verification of Ontology Increments using Natural Language

First Order Logic (FOL) based Algorithmic Approach

Kaneeka Vidanage[1], Noor Maizura Mohamad Noor[2], Rosmayati Mohemad[3], Zuriana Abu Bakar[4]
Faculty of Computing, Department of Data Science, NSBM Green University, Sri Lanka[1]
Faculty of Ocean Engineering Technology & Informatics, University Malaysia Terengganu (UMT), Malaysia[1, 2, 3, 4]

***Abstract*—Parallel to the advancement of practical use cases in computers, the trend toward collaborative ontology engineering is accelerating. Both domain experts and ontologists must collaborate in collaborative ontology engineering processes. However, the bulk of domain experts are not computer experts (i.e. lawyers, medical doctors, bankers, etc.). Question and Answer on Linked Data (QALD) is a suggested method for non-computer domain experts to engage with the ontology increments as they evolve. Existing QALD methods and systems, on the other hand, have a number of drawbacks, including significant setup requirements, domain dependence, and user discomfort. As a result, a new QALD algorithm and QALD system designed with the usage of First Order Logic (FOL) are presented in order to address the shortcomings of current QALD mechanisms. The suggested FOL based, QALD mechanism was tested quantitatively and qualitatively over three distinct ontology increments. This experiment had an overall acceptance rate of 79 percent from all stakeholders.**

Keywords—First order logic; linked data; ontologist; iterative framework

I. Introduction

Ontology increment verification using QALD is critical for evaluating the correctness and relevance of a given ontology increment. Existing QALD methods, on the other hand, have a number of flaws that will be discussed in this article.

Both domain experts and ontologists must work in unison and with mutual understanding during collaborative ontology engineering [1]. Specialists in certain domains will help ontologists by sharing their specific domain expertise (i.e. COVID-19, Criminal Law, Aquaculture, etc.) [23]. On the other side, ontologists will conceive and build ontologies using the information collected from domain experts [24]. Thus, the information contained in the resultant ontology will become both human and machine readable [2], thereby allowing for an unbounded wide variety of application options.

However, the process of developing an ontology is iterative and incremental [3]. At the conclusion of each cycle, the ontologists will generate an ontology increment. Domain experts must then evaluate the ontology increment generated by the ontologists. When information is transferred from domain experts to ontologists, there is a possibility of misinterpretations and ambiguities that result in cognitive glitches. As a result, it is possible that the ontologists do not always replicate the precise cognitive interpretations conveyed by domain experts. Because neither ontologists nor domain experts are ontologists. Consequetly, there are many ways for knowledge errors to result in an incorrect schematic conceptualization at the ontology level. This might be hazardous if such ontologies were to be published directly into the production environment and produced illogical outcomes [4, 25]. QALD is a favored method for bridging this knowledge gap between domain experts and ontologists. Effectively built QALD systems may significantly aid domain experts in their ontology augmentation evaluation process. As a result, domain experts and ontologists may collaborate to debate and implement necessary improvements to the ontology increment under review [5-7].

However, current QALD systems have a slew of problems and restrictions that limit their potential. The bulk of them have a complicated technical curve that excludes non-computer domain experts such as bankers, attorneys, medical practitioners, and marketers from eligibility. it is not possible [8-10]. Therefore, this study introduces a new domain specialist-friendly, domain- and schema-independent, configuration-free algorithm to aid the QALD process in a more effective manner.

II. Literature Review

SPARQL or SQWRL querying capabilities are a high-level capabilities that cannot be acquired overnight. Even if that barrier is resolved, an individual cannot construct a valid SPARQL or SQWRL query without first understanding the schematic structure of the corresponding ontology increment. To understand the schematic structure of an ontology increment, one must be familiar with the semantic web's fundamental notions, such as triple concepts, data and object properties, and individuals. All of them are extra and unnecessary costs for domain experts, which may demotivate their participation [8-10]. However, in collaborative ontology engineering, the domain experts' participation in evaluating ontology increments is critical [2]. The following Table I summarizes an evaluation of various recently implemented QALD systems and their shortcomings.

TABLE I. EXISTING QALD SYSTEM ANALYSIS

QALD Tool	**Deficiency**
Neural Machine Translation [6]	-Training for the deep learning technique takes about 20 days. -Assemble and organize the domain-specified dataset. This is a vast overhead. -The accuracy of the SPARQL queries produced by the deep learning model is insufficient. Results that are ambiguous. -Because the dataset is required, the tool's whole operation is domain-dependent, since domain-independent datasets are not feasible.
Schema-Agnostic QALD [5]	-Using similarity assessment logic, SPARQL queries are generated based on the contents of the natural language query. -Accuracy is very low. Results that produces tempt to be very ambiguous.
Question and Answering on Linked Data (QALD) [11]	- The QALD tool is statically associated with a single ontology. It is incompatible to work with all other ontologies. -Extremely domain-specific. But the requirement is for domain independency
Regular Language to SPARQL questions [12]	-Based on the tool's domain-specific rule sets. -Generates numerous SPARQL queries, even for a single purpose. -Extensive operational overhead and extremely low precision.
ORAKEL [13]	-Protracted period of domain-specific setup utilizing the FrameMapper program. -Intense human interaction throughout the setup process.

In addition to that, an assessment of some of the latest existing QALD algorithms was conducted as depicted in Table II.

TABLE II. EXISTING QALD ALGORITHM ANALYSIS ACCORDING

QALD Algorithm	Deficiency
SPE Algorithm [14]	-An extensive domain-specific configuration effort needs the intense involvement of domain experts. -It is necessary to create predefined question and response pairings.
Conversational Question and Answering (CQA) with BERT [15]	This is accomplished via the use of a sophisticated machine learning model that has been trained on 104 languages. The BERT architecture is not suitable for querying linked data using SPARQL or SQWRL.
Visual Genome and Visual Question Answering [16]	-This is based on the visual genome dataset and a deep neural network trained on it. As a result, it is statically bound to a domain.
QA Optimization pipeline Algorithm [17]	-Frankenstein Framework is the basis for this algorithm. It is intended for the purpose of determining the optimum QA pipeline from 360 configurations and is not tailored to QALD needs.
Template-Based Question and Answering [18]	-Defining templates is a time-consuming manual process that needs significant human participation. -Additionally, each specified template is domain-specific.

As shown in Tables I and II, the current QALD methods have a number of drawbacks. The primary weakness of current QALD methods is as follows:

1) Domain-dependence.
2) Schema dependence.
3) Extensive effort required for manual configuration.
4) Inconvenience to the user.

The purpose of this study is to develop a more user-friendly, domain- and schema-independent QALD method that does not need knowledge of SPARQL or SQWRL to verify ontology increments. Additionally, this is free of lengthy manual setup procedures.

III. METHODOLOGY

The research methodology used in this study is shown in Fig. 1.

After numerous brainstorming sessions with ontologists and domain experts, the nature of the issue and the importance of resolving it were justified. Following that, a systematic evaluation of the most recent available tools and algorithms was performed to identify unsolved gaps. Thus, the objective of "need for a more user-friendly QALD method for verifying ontology increments" was created. Finally, the brainstorming findings resulted in the introduction of the following algorithm. As shown in Fig. 2, the suggested method is divided into four distinct stages.

The algorithm's first phase is responsible for extracting information from the associated ontology increment file. The corresponding ontology increment files may be in OWL (Web Ontology Language) or RDF (Resource Description Framework)format. To begin, this method requires a file containing the ontology increments. Phase I will extract and store the knowledge in a relational database. Below is a representation of the pseudocode for phase-I execution.

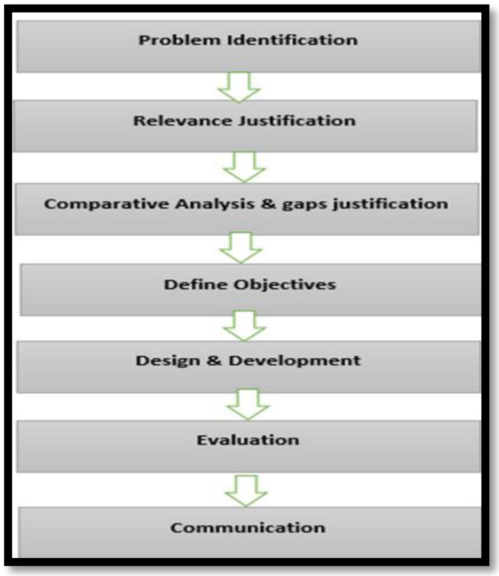


Fig. 1. Research Process.

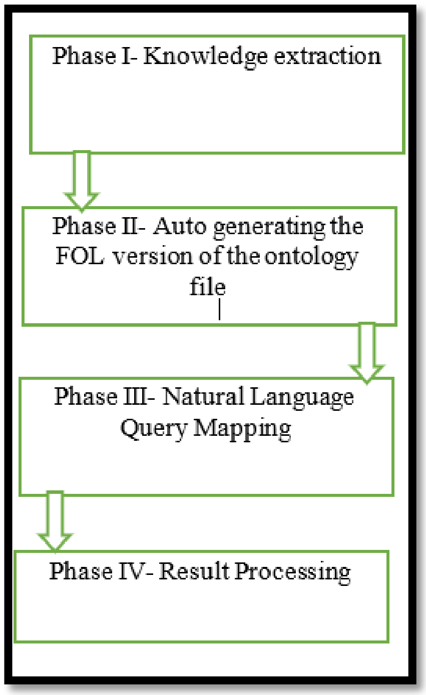
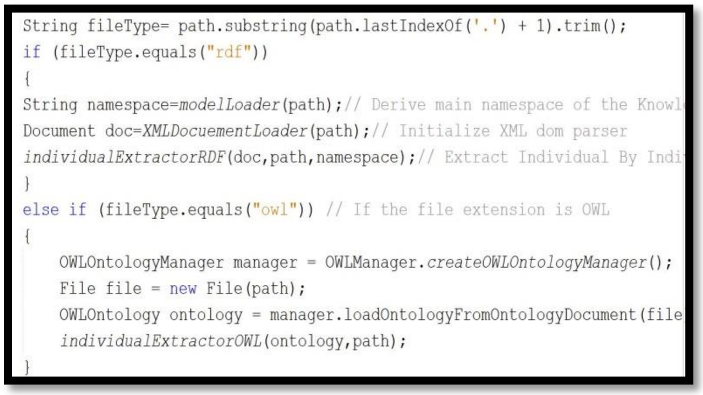

Phase I- Knowledge extraction
Phase II- Auto generating the FOL version of the ontology file
Phase III- Natural Language Query Mapping
Phase IV- Result Processing

Fig. 2. Phases of the Algorithm.

Phase-I - [Knowledge Extraction]

```
Start
Upload relevant RDF / OWL file of the ontology increment to be verified.
Conduct format verification as either RDF or OWL.
According to the verified format, trigger the relevant RDF / OWL extraction logics.
While [ Until EOF ]
        Extracts class info from the ontology increment file.
                [i.e.  Inheritance class info / Disjoint class info…]
        Extract property associated info from the ontology increment file
                [i.e. Data / Object properties]
        IF [INDIVIDUALS Exists]
                Extract INDIVIDUAL specific object and data properties
        End IF

        Store extracted info in the relevant relations of the database Schema without violating mapping sequences.

End While
End
```

The portion of the code snippet associated with the practical implementation of the pseudocode phase -I is depicted in Fig. 3.



```
String fileType= path.substring(path.lastIndexOf('.') + 1).trim();
if (fileType.equals("rdf"))
{
String namespace=modelLoader(path);// Derive main namespace of the Knowl
Document doc=XMLDocuementLoader(path);// Initialize XML dom parser
individualExtractorRDF(doc,path,namespace);// Extract Individual By Indi
}
else if (fileType.equals("owl")) // If the file extension is OWL
{
    OWLOntologyManager manager = OWLManager.createOWLOntologyManager();
    File file = new File(path);
    OWLOntology ontology = manager.loadOntologyFromOntologyDocument(file
    individualExtractorOWL(ontology,path);
}
```

Fig. 3. Implementation of the Phase-I of the Pseudocode.

Phase II is in charge of creating the FOL (First Order Logic) version of the ontology increment by accessing the database's relevant knowledge embeddings. This will be completely automated, with no human intervention required. The pseudocode below illustrates the execution of Phase II.

Phase-II - [Auto-generation of the First Order Logic Knowledge base]

```
Start
ArrayList<String>  folStoreArr=new ArrayList<> () // To store FOL elements
Connection con=DriverManager.getConnection(ConnectionString) // DB connection
Statement stmt=con.createStatement()
ResultSet rs=stmt.executeQuery("select count(*) from table") // Deriving the count of tuples from the relation

While(rs.next())
        Extract semantic element from the DB table.
        Determine the type of the semantic element
                [Either data/object property, class relationship etc..]

* / Organization of semantic elements to be presented in the First Order Logic (FOL) format /*

        Initialize identification flags accordingly for conditional formatting of the values retrieved from the table.
        Check the semantic type of the extracted element and re-write it in the appropriate standardized FOL format.
        folStoreArr.  Add  //Append all semantic elements re-formatted into its FOL version, one by one

End While
        FileWriter fw=new FileWriter("prologue.pl")
        BufferedWriter bufferedWriter = new BufferedWriter(fw);
        bufferedWriter.write(Iterate through the contents stored in folStoreArr);
End
```

The portion of the code snippet associated with the practical implementation of the pseudocode phase II is depicted in Fig. 4. Similarly, the autogenerated first-order logic (FOL) version of the inspecting ontology increment is depicted in Fig. 5. This FOL version of the inspecting ontology increment is completely autogenerated as an outcome of phase II of the algorithm.

Phase II transforms the RDF/OWL file's semantic components to a standardized FOL format series. For instance, among the specified standard format series are the following:

1) Classes are converted as:= "Class(Class Name)".

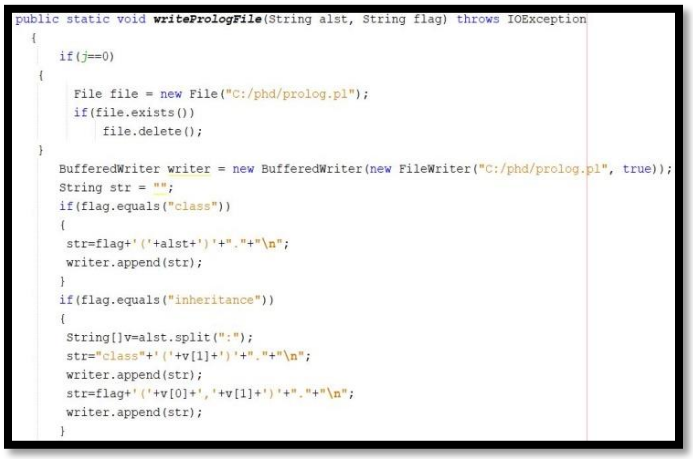
2) Inheritance relationships are converted as:=" Inheritance(Child Class, Parent Class)".

3) Disjoint relationships as :=" Disjoint(Class-1, Class-2)"

4) Individuals as:= "Individual(Class Name, Object Name)"

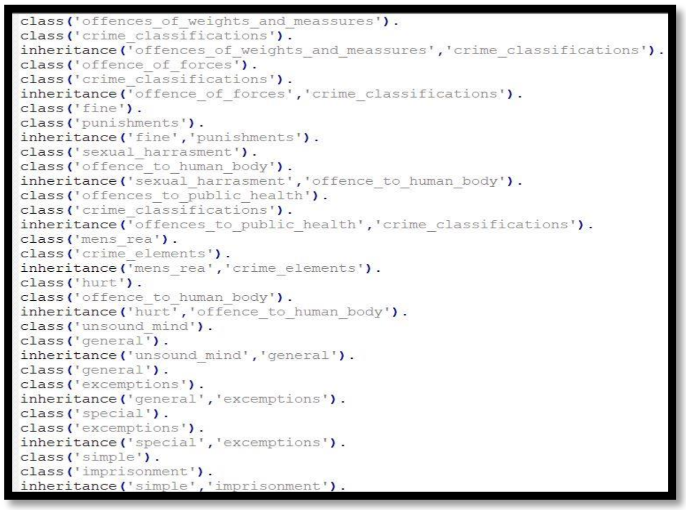
5) Data Properties as:="dProp(Individual, d_prop_1_name, d_prop_1_value, Domain).

6) Object Properties as:="oProp(Individual-1,obj_prop_name, Individual-2, Domain, Range).



```
public static void writePrologFile(String alst, String flag) throws IOException
  {
      if(j==0)
    {
        File file = new File("C:/phd/prolog.pl");
        if(file.exists())
            file.delete();
    }
      BufferedWriter writer = new BufferedWriter(new FileWriter("C:/phd/prolog.pl", true));
      String str = "";
      if(flag.equals("class"))
      {
       str=flag+'('+alst+')'+"."+"\n";
       writer.append(str);
      }
      if(flag.equals("inheritance"))
      {
       String[]v=alst.split(":");
       str="class"+'('+v[1]+')'+"."+"\n";
       writer.append(str);
       str=flag+'('+v[0]+','+v[1]+')'+"."+"\n";
       writer.append(str);
      }
```

Fig. 4. Implementation of Phase II of the Pseudocode.



```
class('offences_of_weights_and_meassures').
class('crime_classifications').
inheritance('offences_of_weights_and_meassures','crime_classifications').
class('offence_of_forces').
class('crime_classifications').
inheritance('offence_of_forces','crime_classifications').
class('fine').
class('punishments').
inheritance('fine','punishments').
class('sexual_harrasment').
class('offence_to_human_body').
inheritance('sexual_harrasment','offence_to_human_body').
class('offences_to_public_health').
class('crime_classifications').
inheritance('offences_to_public_health','crime_classifications').
class('mens_rea').
class('crime_elements').
inheritance('mens_rea','crime_elements').
class('hurt').
class('offence_to_human_body').
inheritance('hurt','offence_to_human_body').
class('unsound_mind').
class('general').
inheritance('unsound_mind','general').
class('general').
class('excemptions').
inheritance('general','excemptions').
class('special').
class('excemptions').
inheritance('special','excemptions').
class('simple').
class('imprisonment').
inheritance('simple','imprisonment').
```

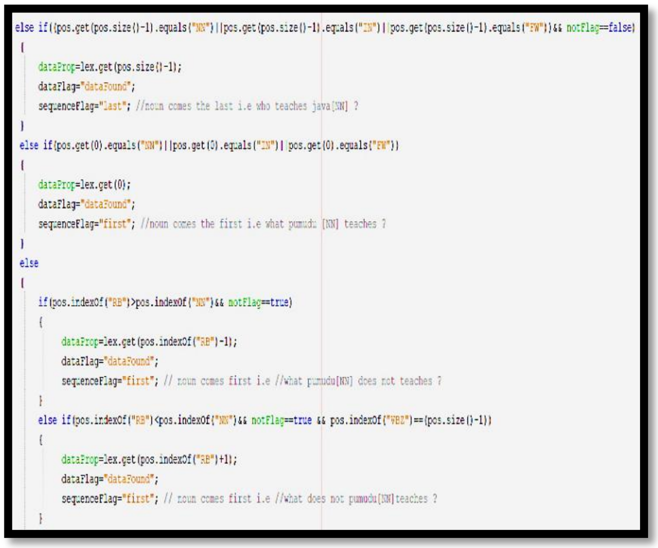
Fig. 5. Auto-Generated FOL Version of the Ontology Increment.

The relevant values retrieved and stored in the RDBMS, which was accomplished in phase I, will be replaced with the associated parameter values accessible in the gerenated standardized FOL rules displayed earlier. These standardized FOL rules along with the replaced parameter values will be utilized for the autogeneration of the FOL file as shown in Fig. 5. Therefore, the phase II of the algorithm converts the entire contents of the ontology increment into it`s respective FOL format. The algorithm's third phase is responsible for locating the FOL components that correspond to the plain language queries. Here, domain experts may immediately ask the necessary queries in English, obviating the requirement for SPARQL or SQWRL literacy entirely. The following pseudocode illustrates the execution of Phase III.

<u>Phase-III - [Natural Language Query Mapping]</u>

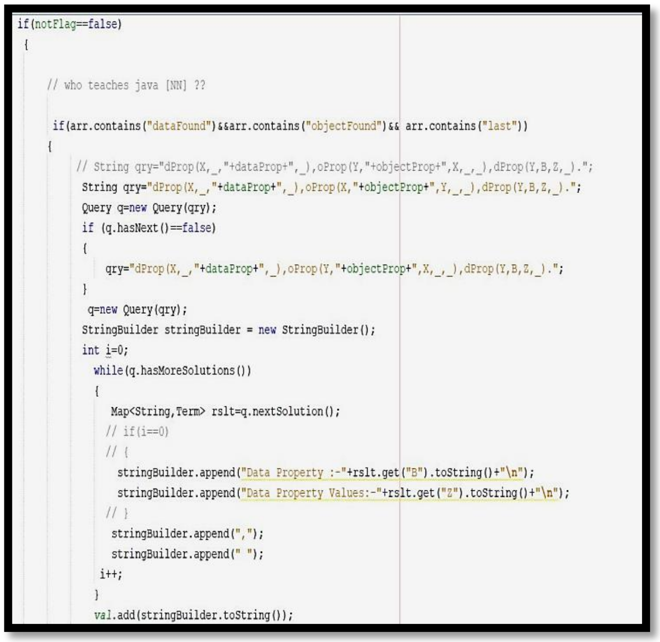
```
Start
Accept natural language (i.e.  English) user query.
Activate Part of Speech Tagging (POS) for the user query classification.
Derive POS sequences for the user query.
Locate positioning of various lexicons and special terms and update flag variables accordingly to classify the user query.
Remove all prepositions, modal verbs, pronouns. Nouns and Verbs will only be remaining.

Locate the nouns and verbs of concern by executing a verification prologue query on the generated prologue knowledge base (i.e. Fig. 5)

Eliminate all unnecessary nouns and verbs after the prologue verification.

IF [Verified nouns / verbs .  Has Both Domain && Range]
        Segment those as object properties.
End If
Else If  [Verified nouns/verbs.  Has Only a Domain && No Range]
        Segment those as data properties
End If

Derive the structure of the user query by analyzing the POS pattern sequences of the flag variables updated.
Map the POS pattern sequence of the flag updation with respective prologue query generation rules.

Generate appropriate chained prologue query
Locate the missing elements.

Execute it on the FOL knowledge base.
End.
```

The code snippet below (i.e. Fig. 6) illustrates the categorization and updating of flag variables depending on the Part Of Speech (POS) of the user inquiry. This technique can be used to extract particular requests for data or object attributes from a natural language-based user inquiry. The following figure (i.e. Fig. 7) illustrates the updating of POS sequence-specific flag variables and the creation of chained prologue queries. The chained prologue query structures are parameterized in this section. An appropriate parameterized chained prologue query will be initiated based on the POS sequences in the natural language user inquiry. At the moment, specified parameterized prologue query structures mapped to the user query's POS sequences are capable of extracting nearly all object and data property queries.



```
else if((pos.get(pos.size()-1).equals("NN")||pos.get(pos.size()-1).equals("IN")||pos.get(pos.size()-1).equals("FW"))&& notFlag==false)
 {
    dataProp=lex.get(pos.size()-1);
    dataFlag="dataFound";
    sequenceFlag="last"; //noun comes the last i.e who teaches java[NN] ?
 }
 else if(pos.get(0).equals("NN")||pos.get(0).equals("IN")||pos.get(0).equals("FW"))
 {
    dataProp=lex.get(0);
    dataFlag="dataFound";
    sequenceFlag="first"; //noun comes the first i.e what pumudu [NN] teaches ?
 }
 else
 {
    if(pos.indexOf("RB")>pos.indexOf("NN")&& notFlag==true)
    {
        dataProp=lex.get(pos.indexOf("RB")-1);
        dataFlag="dataFound";
        sequenceFlag="first"; // noun comes first i.e //what pumudu[NN] does not teaches ?
    }
    else if(pos.indexOf("RB")<pos.indexOf("NN")&& notFlag==true && pos.indexOf("VBZ")==(pos.size()-1))
    {
        dataProp=lex.get(pos.indexOf("RB")+1);
        dataFlag="dataFound";
        sequenceFlag="first"; // noun comes first i.e //what does not pumudu[NN]teaches ?
    }
```

Fig. 6. User Query Classification and Property Extraction.

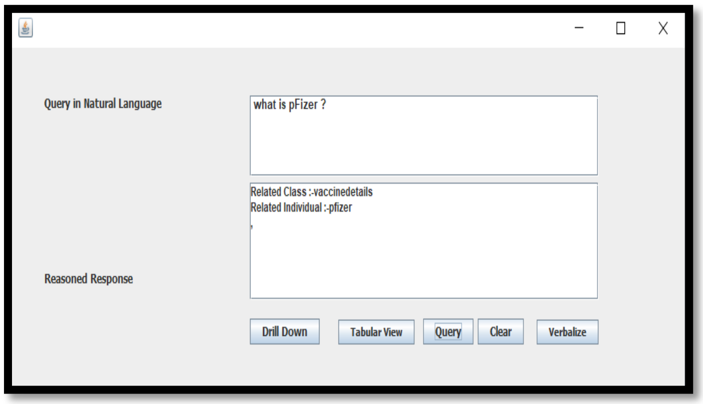


```
if(notFlag==false)
 {

     // who teaches java [NN] ??

      if(arr.contains("dataFound")&&arr.contains("objectFound")&& arr.contains("last"))
     {
         // String qry="dProp(X,_,"+dataProp+",_),oProp(Y,"+objectProp+",X,_,_),dProp(Y,B,Z,_).";
          String qry="dProp(X,_,"+dataProp+",_),oProp(X,"+objectProp+",Y,_,_),dProp(Y,B,Z,_).";
          Query q=new Query(qry);
          if (q.hasNext()==false)
          {
              qry="dProp(X,_,"+dataProp+",_),oProp(Y,"+objectProp+",X,_,_),dProp(Y,B,Z,_).";
          }
           q=new Query(qry);
          StringBuilder stringBuilder = new StringBuilder();
          int i=0;
            while(q.hasMoreSolutions())
            {
               Map<String,Term> rslt=q.nextSolution();
              // if(i==0)
              // {
                stringBuilder.append("Data Property :-"+rslt.get("B").toString()+"\n");
                stringBuilder.append("Data Property Values:-"+rslt.get("Z").toString()+"\n");
              // }
               stringBuilder.append(",");
               stringBuilder.append(" ");
             i++;
            }
            val.add(stringBuilder.toString());
```

Fig. 7. Part of Speech Tag Pattern Specific Parameterized Chained Prologue Queries.

Phase IV is in charge of processing the output in natural language (i.e. English) and ensuring that it is readily understandable by domain experts. The pseudocode below illustrates the execution of Phase IV.

```
Phase-IV - [Results Processing]
Start
   Use StringBuilder to append all returned results with "\n" as the delimiter.
   Split the appended output by the delimiter "\n"
   Return the result in Natural Language (i.e. English)
End
```



```
Related Class :-vaccinedetails
Related Individual :-pfizer
```

Fig. 8. Natural Language Interrogation on Ontology Increment.

The QALD interface designed is shown in Fig. 8 above. The user just has to enter the desired question in English and click the query button. Then, as visible, the relevant findings will be presented in natural language.

Let us attempt to understand the execution of this algorithm using a simple real-world scenario. Assume you have been given a university ontology increment that includes information of professors who teach various courses. Thus, the domain expert may simply ask the query in plain language (i.e. English) to verify that correct mappings are included to the ontology increment.

For instance, who teaches Financial Accounting?

After completing the first levels of processing, the aforementioned English question will be converted into its new structure as teach [VERB] "Financial Accounting" [NOUN]. Thus, a simple Prologue verification query on the phase II-generated 'Prologue. pl' file will validate "teach" as an object property and "Financial Accounting" as a data property. Because "teach" is a property of an object, it will have both a domain and a range. However, "Financial Accounting" will have a domain-exclusive scope. Due to the axiomatic difference between object and data, properties can readily be distinguished with a simple IF condition. Thus, extracted knowldge elements may be represented as follows using the specified standard FOL representation rules:

dPropIndividual, d_prop_1_name, "Financial Accounting", Domain).

Likewise, teach can be represented as:-

oProp(Individual-1," teach", Individual-2, Domain, Range)

As per the question asked our concern is to find a person. Therefore, we can negate the unnecessary fields of the prologue query as mentioned below.

dProp(Individual,_, "Financial Accounting", _).
oProp(Individual-1," teach", Individual-2, _, _)

Henceforth, a chained prologue query can be formulated as:

dProp(X,_," Financial Accounting ",_),oProp(Y,"teach",X,_,_),dProp(Y,B,Z,_).

Once the above-chained prologue query is executed in the fact base following conclusions can be derived.

dProp(X,_," Financial Accounting ",_) → X (Individual) => "Subject-1"

After, assign the result to the second prologue query.

oProp(Y,"teach",Subject-1,_,_) → Y(Individual)=>"Lect-1"

Henceforth, assign the result to the third prologue query.

dProp("Lect-1",B,Z,_).

Hence, B represents the data property names and the Z represents the data property values.

The algorithm's fourth step allows for the formatting and representation of returning results as cleaned string outputs, resulting in the names of individuals who teach Financial Accounting.

According to Lampa's comprehensive experiment report [19], prologue searches are straightforward and close to 10 times quicker than SPARQL or SQWRL queries. The primary reason for this is because the Prologue reasoning engine uses a backtracking search technique to explore the fact base in search of necessary axioms. SPARQL requires the generation of a parse tree after finding the ontology's schema, which is a highly complicated and resource-intensive operation [20]. As a result, our SPARQL-free natural language interrogation technique represents a major addition.

IV. Result and Discussion

The algorithm described above was exposed in three distinct ontology increments, one for COVID-19, one for aquaculture, and one for criminal law. This experiment included a total of fifteen stakeholders. Nine were domain experts, while the remaining six were professional ontologists. The appendix of this article contains snapshots of the three ontology increments used in this experiment. We did not go into detail on such ontology increments since they are outside the scope of this article.

The evaluation workflow for the suggested new algorithm is depicted in Fig. 9. This suggested evaluation workflow took into account both quantitative and qualitative perspectives.

Prior to it, the operationalization phase was accomplished. We prepared a list of open-ended questions regarding the study's goals. Operationalization entails matching questionnaire items to the research objective [21]. This guarantees that the questionnaire's questions elicit highly relevant and consistent answers. The following is a collection of open-ended questions that correspond to the assessment's study aim:

1) Have you been informed of the NLI processes that are currently in place?

2) In comparison to them, what are the favorable characteristics of this mechanism that you identified?

3) Do you believe it will ease the ontology increment verification function?

4) Could you expand on how this would ease the inspectors' work?

5) What flaws did you discover in the suggested verbalization mechanism?

Both ontologists and domain specialists were shown a specially created synoptic video clip about the research as part of the pre-warm-up setup. This phase acts as a retrospective, summarizing the major findings of the research performed by the evaluation's stakeholders. This was done prior to the formal commencement of the assessment process in order to clear up any remaining questions about the procedure.

A face-to-face interview series with nine domain experts in criminal law, COVID-19, and aquaculture was done during the controlled interview session. The five questions outlined above served as the basic foundation for the nine domain experts' interviews. All controlled interview sessions were videotaped to aid in subsequent analysis. All participants gave their previous permission and consent to the recording, which was utilized only for research purposes and not for personal benefit.

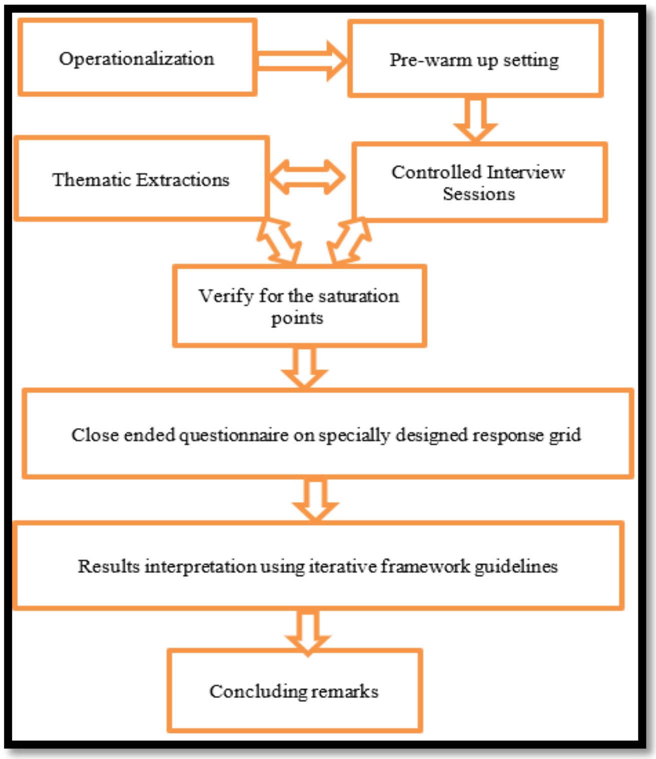

Operationalization
Pre-warm up setting
Thematic Extractions
Controlled Interview Sessions
Verify for the saturation points
Close ended questionnaire on specially designed response grid
Results interpretation using iterative framework guidelines
Concluding remarks

Fig. 9. Evaluation Workflow.

During the thematic extraction process, all recorded interviews were transcribed into textual format. Following that, the research team iteratively analyzed the transcribed texts for numerous turns. The data gathered over the course of the repetitive study were categorized into a few broad themes. At the outset of the research, new themes developed rapidly; however, as the study progressed to the ninth transcription, the development of new topics slowed significantly, while the same motifs reappeared repeatedly. This characteristic was identified as approaching saturation.

The extraction of themes facilitated the identification of the most intriguing aspects of the research. It was difficult to elicit all important views simply via quantitative procedures. As a consequence of the qualitative phase, which was conducted through controlled interview sessions, important user insights were identified.

Following that, we created a series of closed-ended questions to extract more information on the highlighted themes. This enables us to focus our attention on particular subjects while simultaneously emphasizing their numerical importance. Fig. 10 illustrates the process of extracting quantitative stakeholder views using a customized rating grid.

10	20	30	40	50	60	70	80	90	100
Very Poor	Fairly OK – Major Revisions				Good & Acceptable – Minor Revisions			Exceptional	

Fig. 10. Close-Ended Response Grid.

The four questions below were given in a closed-ended manner, with respondents invited to evaluate their level of agreement with the quantitative inspection criteria.

1) SPARQL / SQWRL is not required, and interrogations may be initiated using plain language (i.e. English).

2) Addresses needs for on-demand, ad-hoc knowledge verifications.

3) Operations are domain and schema-independent and configuration-free.

4) The retured findings were accurate.

5) How would you evaluate the tool support's NLI assistance?

The mean answer scores received from nine domain experts from three distinct domains and the ontologists participating in this research are summarized in Table III.

The following Table IV summarizes the qualitative interpretations elicited during the controlled interview session.

TABLE III. AVERAGED RESPONSE SCORES FROM DOMAIN SPECIALISTS

Natural Language Interrogation [NLI]	Law	80%
	COVID-19	76%
	Aquaculture	83%
	Averaged	**79%**

TABLE IV. QUALITATIVE RESPONSE SYNTHESIS

Natural Language Interrogation [NLI]	-Completely automated, configuration-free operation that is domain and schema agnostic. -Assists with ad-hoc, on-demand consolidation analysis during ongoing discussions -Eradicates technological obstacles associated with SPARQL or SQWRL -Comprehensible user inquiries expressed in natural language by domain specialists (i.e., English) -Perfect for arbitrary knowledge verification in real-time. - Returned findings are readily understandable and precise

The iterative framework was utilized as the last stage of the assessment process to keep the emphasis on the research objective. Iterative framework [22] is well-established method for evaluating the effectiveness of rationally accomplishing research objectives. Three different but connected questions control the iterative framework's functioning. Each part must provide reflective proof. Table V summarizes the discussion of iterative framework measurements.

TABLE V. ITERATIVE FRAMEWORK ASSESSMENT

Steps in Iterative Framework	Reflective Evidence
01 → What are the data telling me?	Quantitative Metrics: As shown in Table III above, various domain-specific qualitative opinion ratings were utilized to verify the efficacy and operational effectiveness of the developed QALD algorithm. It had yielded favorable outcomes. Qualitative Evaluation: The QALD algorithm was experimentally assessed with the involvement of stakeholders who participated to the development of the ontology increments. Precision, usability, and "technical help offered" were all cited as critical characteristics of the returned findings. Additionally, as indicated in Table IV above, the themes of stakeholders' reflective views were documented. The entire research yielded acceptable findings in both the quantitative and qualitative experimental stages.
02 → What do I want to know?	Need for a domain specialist friendlier QALD mechanism for the ontology increment`s arbitrary verification
03 → Is there a dialectical relationship between step 01 & 02?	The QALD method was exposed to multiple ontology increments in three different domains throughout the quantitative assessment phase. Quantitative matrices were utilized to evaluate the overall efficacy of the QALD algorithm in each of these tests, and it was apparent that the overall operation was a success. Throughout the qualitative assessment process, the views of stakeholders were analyzed thematically, and the distilled findings are summarized in Table IV. The quantitative and qualitative assessment stages were both completed successfully. As a consequence of the iterative framework's reasoning, it is feasible to infer that the connection between stages 01 and 02 is positive and acceptable, indicating the effectiveness of the newly suggested QALD algorithm.

V. Conclusion

QALD mechanisms will be very helpful throughout the collaborative ontology engineering process for doing ad-hoc verifications of the knowledge embeddings contained in the ontology increments. As previously mentioned, mutual understanding between domain experts and ontologists is critical for the successful development of applied ontologies. Domain experts, on the other hand, such as medical physicians, attorneys, bankers, and marketers, are not computer specialists fluent in SPARQL, SQWRL, or Semantic principles. Thus, establishing English-language-based QALD mechanisms will be critical for domain experts to communicate successfully with ontologists and accomplish their assigned tasks.

As previously stated, current QALD methods have a number of flaws, as shown in Tables I and II above. As a result, this study proposes a new algorithm that addresses those problems by enabling domain- and schema-independent, configuration-free QALD intervention. The proposed method was evaluated in three distinct domains and produced successful results with an overall of 79% acceptance from the involved stakeholders, as shown in Tables III, IV, and V. Therefore, it may be characterized as a new and important addition to the field of semantic technologies. In the future, it is planned to test the algorithm's effectiveness across a variety of other areas and to incorporate the chatbot capability to further enhance human-computer interaction views.

References

[1] De Nicola, A., & Missikoff, M. (2016). A lightweight methodology for rapid ontology engineering. Communications of the ACM, 59(3), 79-86. doi:10.1145/2818359doi:10.1093/bib/6.3.239.

[2] Abeysiriwardana, P. C., &amp; Kodituwakku, S. R. (2012). Ontology-Based Information Extraction for Disease Intelligence. International Journal of Research in Computer Science, 2(6), 7-19. doi:10.7815/ijorcs.26.2012.051 ACIS 2011 Proceedings. 80.

[3] Spasic, I., Ananiadou, S., McNaught, J., & Kumar, A. (2005). Text mining and ontologies in biomedicine: Making sense of raw text.Briefings in Bioinformatics, 6(3), 239-251.

[4] Ingram, J., & Gaskell, P. (2019). Searching for meaning: Co-constructing ontologies with stakeholders for smarter search engines in agriculture. NJAS - Wageningen Journal of Life Sciences, 100300. doi:10.1016/j.njas.2019.04.006\.

[5] Strohmaier, M., Walk, S., P Pschko, J., Lamprecht, D., Tudorache, T., Nyulas, C. Noy, N. F. (2013). How Ontologies are Made: Studying the Hidden Social Dynamics Behind Collaborative Ontology Engineering Projects. SSRN Electronic Journal. doi:10.2139/ssrn.3199036.

[6] Steinmetz, N., Arning, A., & Sattler, K. (2019). From Natural Language Questions to SPARQL Queries: A Pattern-based Approach. BTW.

[7] Soru, T., Marx, E., Valdestilhas, A., Esteves, D., Moussallem, D., & Publio, G. (2018). Neural Machine Translation for Query Construction and Composition. ArXiv, abs/1806.10478.

[8] Hamon, T., Grabar, N., & Mougin, F. (2017). Querying biomedical Linked Data with natural language questions. Semantic Web, 8, 581-599.

[9] Trokanas, N., & Cecelja, F. (2016). Ontology evaluation for reuse in the domain of Process Systems Engineering. Computers & Chemical Engineering, 85, 177-187.doi:10.1016/j.compchemeng.2015.12.003.

[10] Munir, K., & Anjum, M. S. (2018). The use of ontologies for effective knowledge modelling and information retrieval. Appl Comput Inform., 14(2): 116–126, ISSN 2210-8327.

[11] Elve, A. T., & Preisig, H. A. (2019). From ontology to executable program code. Computers & Chemical Engineering, 122, 383-394.

[12] Fathalla, S., Lange, C., & Auer, S. (2019). A HumanFriendly Query Generation Frontend for a Scientific Events Knowledge Graph. TPDL.

[13] Pradel, C., Haemmerlé, O., & Hernandez, N. (2013). Natural language query interpretation into SPARQL using patterns.

[14] Cimiano, P., Haase, P., Heizmann, J. (2007). Porting natural language interfaces between domains: an experimental user study with the orakel system. In: IUI '07: Proceedings of the 12th international conference on Intelligent user interfaces, 180–189. ACM, New York, NY, USA.

[15] Lai, Y., Lin, Y., Chen, J., Feng, Y., & Zhao, D. (2016). Open Domain Question Answering System Based on Knowledge Base. NLPCC/ICCPOL.

[16] Otegi, A., Gonzalez-Agirre, A., Campos, J.A., Soroa, A., & Agirre, E. (2020). Conversational Question Answering in Low Resource Scenarios: A Dataset and Case Study for Basque. LREC.

[17] Noh, H., Kim, T., Mun, J., & Han, B. (2019). Transfer Learning via Unsupervised Task Discovery for Visual Question Answering. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 8377-8386.

[18] Singh, K., Radhakrishna, A.S., Both, A., Shekarpour, S., Lytra, I., Usbeck, R., Vyas, A., Khikmatullaev, A., Punjani, D., Lange, C., Vidal, M., Lehmann, J., & Auer, S. (2018). Why Reinvent the Wheel: Let's Build Question Answering Systems Together. Proceedings of the 2018 World Wide Web Conference.

[19] Punjani, D., Singh, K., Both, A., Koubarakis, M., Angelidis, I., Bereta, K., Beris, T., Bilidas, D., Ioannidis, T., Karalis, N., Lange, C., Pantazi, D., Papaloukas, C., & Stamoulis, G. (2018). Template-Based Question Answering over Lilnked Geospatial Data. Proceedings of the 12th Workshop on Geographic Information Retrieval.

[20] Lampa, S. (2010). SWI-Prolog as a Semantic Web Tool for semantic querying in Bioclipse: Integration and performance benchmarking.

[21] Jespersen, L.N., Michelsen, S.I., Holstein, B., Tjørnhøj-Thomsen, T., & Due, P. (2018). Conceptualization, operationalization, and content validity of the EQOL-questionnaire measuring quality of life and participation for persons with disabilities. Health and Quality of Life Outcomes, 16.

[22] Srivastava, P., & Hopwood, N. (2009). A Practical Iterative Framework for Qualitative Data Analysis. International Journal of Qualitative Methods, 8, 76 - 84.

[23] Narock, T., & Fox, P. (2015). The Semantic Web in Earth and Space Science. Current Status and Future Directions. Amsterdam, Netherlands: IOS Press.

[24] McDaniel, M., &Storey, V. C. (2019). Evaluating Domain Ontologies. ACM Computing Surveys, 52(4), 1-44. doi:10.1145/3329124.

[25] Façanha, R. L., Cavalcanti, M. C., & Campos, M. L. (2019). A Systematic Approach to Review Legacy Schemas Based on Ontological Analysis. Metadata and Semantic Research, 63-75. doi:10.1007/978-3-030-14401-2_6.

Appendix

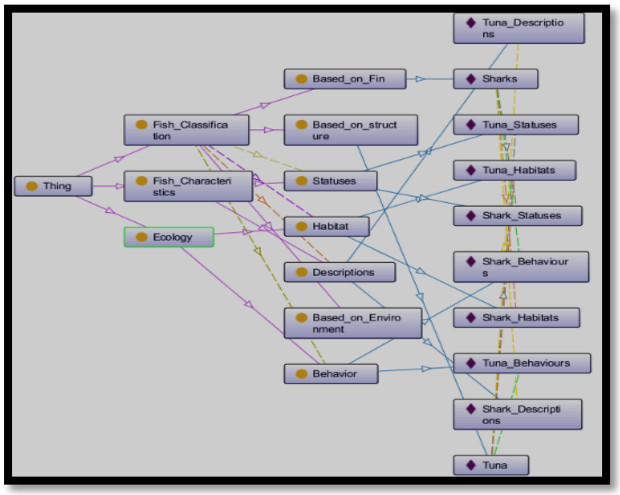


Fig. 11. Ontology Increment for Aquaculture Domain.

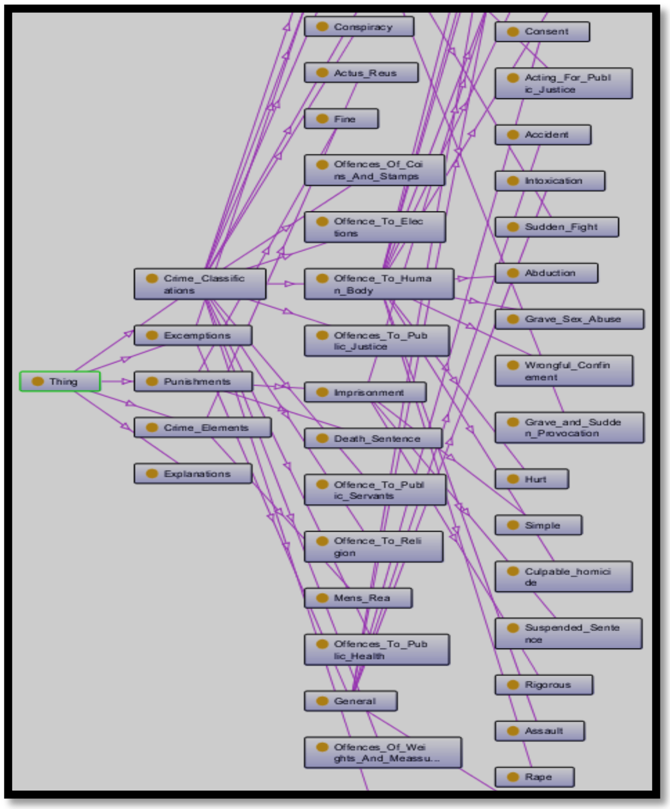


Fig. 12. Ontology Increment for Criminal Law.

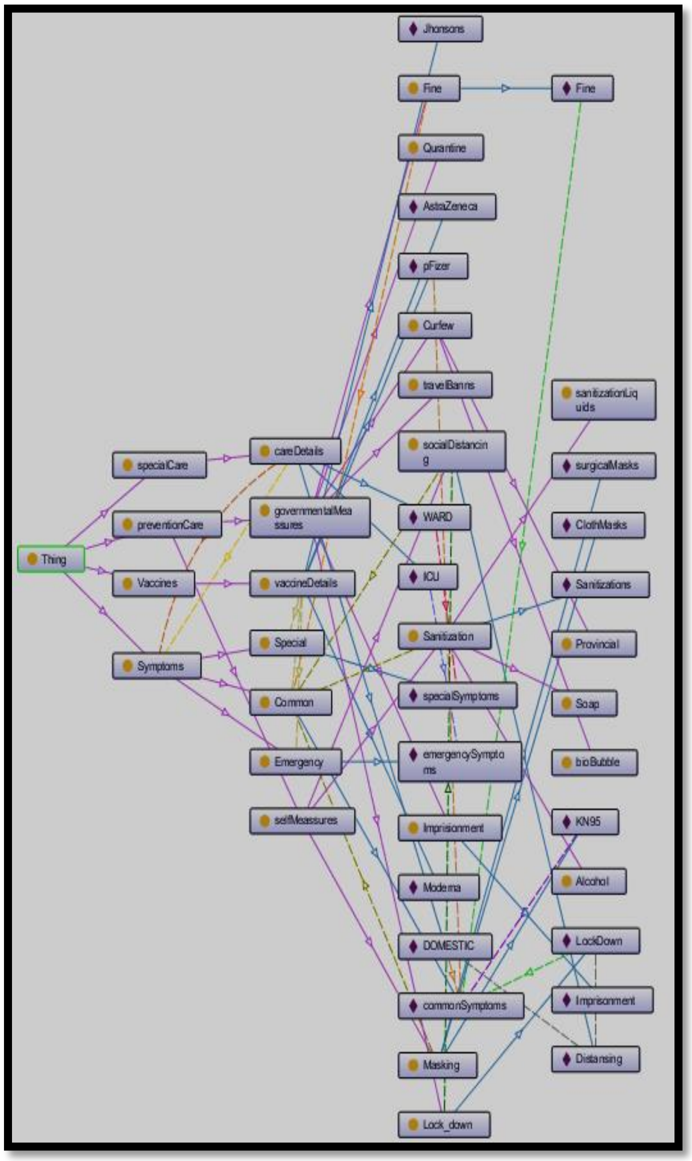


Fig. 13. Ontology Increment for COVID-19.