ECTI Transactions on Computer and Information Technology
Journal homepage: https://ph01.tci-thaijo.org/index.php/ecticit/
Published by the ECTI Association, Thailand, ISSN: 2286-9131

Automated Knowledge Integration from Heterogeneous Data Sources Using Text Analytics: A Case Study of COVID-19

Patipon Wiangnak[1] and Areerat Trongratsameethong[2]

ABSTRACT

Gathering information from multiple data sources takes a long time to collect, analyze, and classify. Furthermore, if the data sources have different data structures, the merged data structure must support such heterogeneity. In addition, semantic of data must also be considered. This paper proposes automated knowledge integration from heterogeneous data sources using ontology engineering combined with text analytics. Text stemming is used to preprocess data. Part-of-speech (POS) tagging, Universal Dependencies (UD), and text similarity measurement called cosine similarity are used to analyze and integrate data. Our work focuses on five COVID-19 knowledge scopes: COVID-19, coronaviruses, diseases, pandemics, and vaccines. For evaluation, six ontologies were constructed with six different cosine similarity values ranging from 0.5 to 1.0. Each constructed ontology has COVID-19 related and non-COVID-19 data in a ratio of 70 to 30. The six constructed ontologies were evaluated for consistency with the original data. Using cosine similarity with 0.6, precision, recall, and F1-score are 0.82, 0.71, and 0.76, respectively, and the constructed ontology is optimal, containing the highest amount of relevant COVID-19 information for this case study.

Article information:

Keywords: Cosine Similarity, Heterogeneous Data Structures, Natural Language Processing, Ontology Engineering, Ontology Integration

Article history:
Received: August 11, 2023
Revised: September 21, 2023
Accepted: November 9, 2023
Published: December 9, 2023 (Online)

DOI: 10.37936/ecti-cit.2023174.253785

1. INTRODUCTION

Coronavirus Disease 2019 (COVID-19) is a worldwide pandemic caused by severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2), which is rapidly spreading worldwide. For the past few years, a large number of people have been infected and died from the infection. Hence, the information about COVID-19 is of interest to many people. There are many interesting perspectives on COVID-19 information to be collected and studied, such as vaccines, transmission, prevention, etc. This information is published by many organizations, such as World Health Organization (WHO) and Health ministries of various countries. The information is also hosted on many repositories, such as crowdsourced DBpedia and Wikipedia, among many others. However, the provided information is in various file formats such as CSV, JSON, ontology, and HTML, with different data structures. Manually collecting and linking data from heterogeneous data sources is cumbersome. If there is a system that can support knowledge integration and store this knowledge automatically, the time for collecting, analyzing, filtering and storing will be reduced. Additionally, the structure for integrating data from various sources must support different data formats and be scalable.

Ontology is a knowledge base used to store concepts. It can support a variety of relationships such as superclass/subclass, is-part-of, and association. Moreover, ontology data are stored in a semantic graph that can support a variety of data structures. Automated ontology construction from heterogeneous data sources requires text analytic techniques. Natural Language Processing (NLP) is a text analytical technique providing many functionalities, such as text preprocessing and text analysis, which can support data cleansing and domain classification, respectively.

In this research, data from heterogeneous sources are automatically merged to construct COVID-19 knowledge by using ontology engineering and text analytics. Web-based services, such as HyperText

[1,2] The authors are with Department of Computer Science, Faculty of Science, Chiang Mai University, Chiang Mai, Thailand 50200, E-mail: patipon_wi@cmu.ac.th and areerat.t@cmu.ac.th
[2] The corresponding author: areerat.t@cmu.ac.th

Transfer Protocol (HTTP) request and web Application Program Interfaces (APIs), are used to collect the structured COVID-19 data. In contrast, web scraping is used for managing the unstructured data. Gathering and storing relevant COVID-19 knowledge will be helpful to people in various fields for future study and research.

2. RELATED WORK

2.1 Heterogeneous Data Source Integration

The challenge of data integration [1] is how to deal with the following heterogeneity: data model, data relationship, and data semantic. It is not easy to prepare and store the heterogeneous data formats into one relational data model, because the relational models typically reside on specific relationships predefined in the relational database schema. On the other hand, ontology schema is flexible and can support various data structures.

Dynamic integration [2] is a principle of weakly structured data integration from web-system. The weakly structured data are transformed in the integration process. The transformation may relate to the presence of following conflicts: heterogeneity, name, semantic, and structure. Ontologies and metamodels are used to create a system of dynamic integration of weakly structured data.

Web scraping [3] is a technique to extract specific information from an HTML tags. Python BeautifulSoup [4] is a library used to inspect the contents of all HTML tags.

This article focuses on creating an ontology schema and integrating relevant ontology information from heterogeneous structured and unstructured data.

2.2 Text Analytics

One of the text analytical techniques is text preprocessing. This method is used to make the original text processable and analyzable. Text preprocessing in [5] is used to prepare text data for sentiment analysis, an analysis of digital text to determine the emotional tone of the message, which includes the following steps: tokenization, text cleansing, part-of-speech (POS) tagging, and text stemming.

Text analysis relies on linguistic theory to analyze the syntax of a language. Universal Dependency (UD) [6] is a framework used to create treebank structures for more than one hundred languages. It utilizes grammatical relations between words in sentences to display morphosyntactic structures with morphological features and POS tags to provide word properties. Morphosyntactic structures are grammatical categories or linguistic units with morphological and syntactic properties in a sentence structure. Morphological features in linguistics are stem or root words, suffixes, and prefixes.

The cosine similarity [7] is used to match classes between two transport ontologies. The similarity between classes from two ontologies are measured. If the cosine similarity is greater than or equal to 0.5, the two classes are assumed to be similar. The cosine similarity is defined in (1).

$$Cosine\ Similarity = \frac{v_A \cdot v_B}{\|v_A\| \cdot \|v_B\|} \tag{1}$$

where, $v_A \cdot v_B$ is the dot product between vectors A and B, while $\|v_A\| \cdot \|v_B\|$ represents the cross product between lengths of two vectors.

Text analytic techniques used in this paper are as follows. POS tagging is used to extract noun phrases from sentences of data sources. Text stemming is used in text preprocessing to remove affixes from noun phrases. UD is used to eliminate the noun phases containing specific irrelevant terms, such as polio vaccine, influenza disease, etc. Cosine similarity is used to measure the relevance of preprocessed data acquired from heterogeneous data sources.

2.3 Ontology Engineering

Ontology [8] is a formal specification of knowledge concepts consisting of classes, properties, and instances. It is used to describe knowledge instances and relationships among them. Six steps of ontology engineering [9] are used to design ontology.

1) *Determine Scope:* Scopes or concepts to be covered by an ontology are determined in this step.

2) *Consider Reuse:* This step is used to find other existing ontologies that are suitable for the current task.

3) *Enumerate Terms:* Terminologies related to ontology concepts are defined in this step.

4) *Identify Classes and Properties:* This step is used to identify classes and their properties related to an ontology. There are two types of properties: data properties and object properties. Data property is used to connect an instance to literal value(s). Object property is used to relate instances of two classes.

5) *Define Constraints:* This step is used to define ontology constraints. There are two types of constraints: class restriction and property restriction. Three are three main class restrictions: quantifier restriction, cardinality restrictions, and hasValue restriction. Property restrictions are cardinality restriction, value-type restriction, and domain-range restriction.

6) *Create Instances:* This step is used to create instances of ontology classes.

In this research, processes used for constructing ontology schema and merging ontology data, are based on ontology engineering.

2.4 Ontology Evaluation

An information retrieval metric [10] is used to evaluate an ontology. The evaluation performances are

determined by precision, recall, and F1-score, displayed in (2), (3), and (4), respectively. The precision is the fraction of the documents retrieved relevant to the information needed. The recall is the fraction of the documents relevant to document retrieved. The F1-score provides a weighted harmonic mean of the precision and recall.

$$Precision = \frac{RelevantRetrieved}{Relevant} \tag{2}$$

$$Recall = \frac{RelevantRetrieved}{Retrieved} \tag{3}$$

$$F1_{score} = \frac{2 * Precision * Recall}{Precision + Recall} \tag{4}$$

The constructed ontologies will be evaluated using precision, recall, and F1-scores.

3. METHODOLOGY

COVID-19 ontology is constructed automatically based on ontology engineering process and text analytics. Text analytic is applied to integrate data from heterogeneous data sources. There are five steps in COVID-19 ontology integration displayed in Fig. 1. Python has various libraries to support Extract Transform Load (ETL) processes and data analytics. The Python libraries are therefore used in the automated ontology construction process.

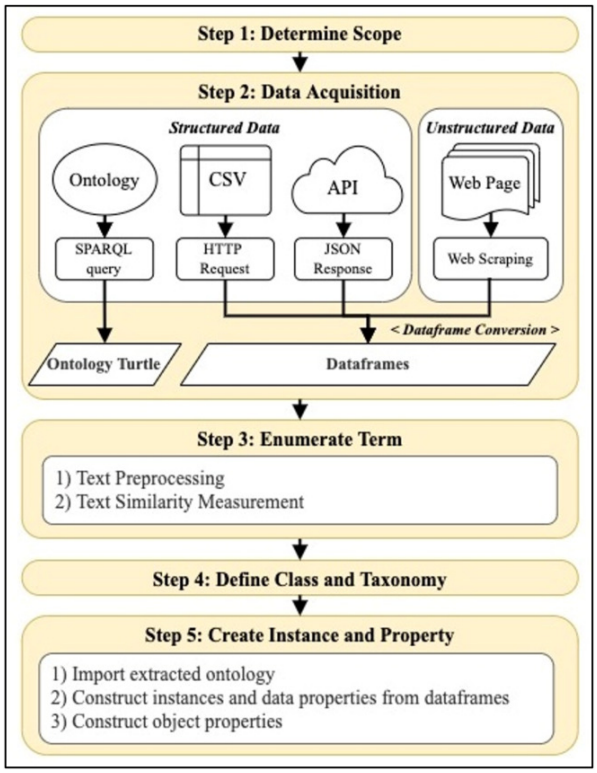

Fig.1: *Automated ontology construction using ontology engineering and text analytics.*

3.1 Determine Scope

Knowledge constructed in this case study is to raise COVID-19 awareness. Scopes of the ontology are focused on COVID-19, Coronavirus, Disease, Pandemic, and Vaccine.

3.2 Data Acquisition

COVID-19 information is available from various sources. The information is provided in many web-based services, such as HTTP request, web APIs, and web pages. The information gathered from these sources is about COVID-19 knowledge and statistics on infections and deaths, excluding sensitive information, such as patient and their medical records. This information is classified into two groups: structured data and unstructured data obtained from trusted sources, as follows:

1) Structured Data: The structured data are typically cleansed, and have clear structure. The following techniques are used for data acquisition:

a. *DBpedia:* DBpedia [11] is a Web API service for extracting knowledge data in many domains from various sources, and making them available on the web using Semantic Web and Linked Data technologies. Data provided by the DBpedia are in ontology formats, and SPARQL query tools are allowed for accessing the ontology data. Python SPARQLWrapper [12] is used as an ontology query tool to obtain only data related to the determined scopes in ontology Terse RDF Triple Language (Turtle). The ontology Turtle represents knowledge in triples containing subject, predicate, and object. It is an appropriate format for later steps. An algorithm for gathering data from DBpedia is displayed in Fig. 2.

b. *HTTP requests to World Health Organization:* WHO provides HTTP request service to search and download vaccine-related information. The determined scopes were used to search for information provided by WHO. The results returned from this source are CSV files. Python pandas is used to convert the CSV files to dataframes, which are then analyzed. The names of the CSV file are set as the name of dataframes.

c. *disease.sh:* The disease.sh [13] is an open disease-related statistics web API in JSON format. It provides summaries of cases and deaths of COVID-19 patients. Python urllib [14] is used to establish a connection and read the data in the JSON format provided by this API service. The results returned from this source are in JSON format covering COVID-19. These JSON files were converted into dataframes using Python pandas. The root element in the JSON file is defined as the name of the dataframe, while the child elements are defined as the column names.

```
Input:  determinedScopes (the scopes defined in step 1 of ontology engineering)
Output: ontologyTurtle
1  Function acquire_ontology_data_source(determinedScopes):
2      ontologyTurtle = EMPTY_LIST
3      sparqlQueries = SPARQLWrapper("http://dbpedia.org/sparql")
4      For each scope in determinedScopes:
5          Execute following sparqlQueries statement:
6              ''' SELECT DISTINCT ?inst ?prop ?thing WHERE {
                   ?inst ?prop ?thing ; rdfs:label ?label .
                   FILTER(regex(?label, scope, "i") && lang(?label)="en") } '''
7          Append the triple results to ontologyTurtle
8      Return ontologyTurtle in turtle (.ttl) file format
```

Fig.2: *An algorithm for requesting COVID-19 information from DBpedia using SPARQLWrapper.*

2) Unstructured Data: The following techniques are used for data acquisition: Wikipedia is a publicly available online encyclopedia and contains a significant amount of information presented on web pages in HTML format. The data, which are articles, are unstructured text. Web scraping technique is used for extracting information residing in HTML articles. Important information is often summarized in tables or graphs. Therefore, the information within HTML table tags is extracted. In our work, Python BeautifulSoup is applied for scraping web. The returned data must be cleansed, and their data structures must be created. There are three subprocesses to acquire COVID-19 information from Wikipedia: web crawling, web scraping, and HTML table-to-dataframe conversion. The algorithm of these subprocesses is displayed in Fig. 3.

a. *Web Crawling:* This process is to browse web pages to collect URLs matching the determined scopes. The first set of URLs is obtained from keyword searching on Wikipedia. Subsequently, if the web pages of the matched URLs contain <a> tags or links connected to other web pages, all URLs specified after href attributes of <a> tags are also collected.
b. *Web Scraping:* The web scraping techniques are applied to extract HTML tags for each URL obtained in the previous step. The HTML tables and text in the following tags in the web pages are extracted. The document title, <title>, describes what the document is about. HTML heading, <h1> to <h6>, explains what the information under the heading is about. The <h1> defines the most general title and <h6> defines the most specific title. The table caption, <caption> describes what the information contained in the table is about. Therefore, the table caption, heading, and title tags are also extracted in this step. For the tables, the <th> tag defines a header column in an HTML table. The <td> tag defines the data within the column of each row.
c. *HTML to dataframe conversion:* The data in HTML tables from the previous step are converted to dataframes. If HTML table contains caption, the table caption is set as dataframe name. If table caption is not presented, the data of the nearest heading tag is designated as a dataframe name. The nearest heading tag reflects the topic name more closely than heading tags farther away. If both the table caption and heading tags are missing, the HTML title tag will be designated as the dataframe name. Fig. 4 shows example results of unstructured data acquisition.

```
Input:  determinedScopes
Output: dataframes, dataframeNames
1  Function acquire_web_pages_data_source(determinedScopes)
2      dataframes = EMPTY_LIST
3      dataframeNames = EMPTY_LIST
4      crawledUrls = web_crawling(determineScopes)
5      cleansedHtml = web_crawling(crawledUrls)
6      For each tableName, cleansedTable in cleansedHtml
7          df = pandas.read_html(cleansedTable)
8          Append df to dataframes
9          Append tableName to dataframeNames
10     Return dataframes, dataframeNames

11 Function web_crawling(determineScopes):
12     crawledUrls = EMPTY_LIST
13     For each scope in determineScopes:
14         url = "http://en.wikipedia.org/wiki/" + scope
15         response = requests.get(url)
16         If response is not NULL:
17             Append url to crawledUrls
18             soup = BeautifulSoup(response, "html.parser")
19             For each articleTag in soup.find_all("a"):
20                 If articleTag.get("href") is not NULL:
21                     linkedUrl = articleTag.get("href")
22                     If requests.get(linkedUrl) is not NULL:
23                         Append linkedUrl
24     Return crawledUrls

25 Function web_scraping(crawledUrls):
26     htmlTables = EMPTY_LIST
27     For each url in deduplicated crawledUrls:
28         soup = BeautifulSoup(requests.get(url), "html.parser")
29         For each headingTag in soup.find_all(re.compile("^h[1-6]$")):
30             For each tableTags in headingTag.find_next_siblings("table"):
31                 tableName = headingTag.text
32                 For each tableTag in tableTags:
33                     If tableTag.find("caption") is not NULL:
34                         tableName = tableTag.find("caption").text
35                         cleansedTable = soup.select("tr[rowspan], th[colspan]").
                                           parent.decompose()
36                         Append tableName and cleansedTable to cleansedHtml
37     Return cleansedHtml
```

Fig.3: *Web scraping algorithm for collecting HTML data from Wikipedia using BeautifulSoup.*

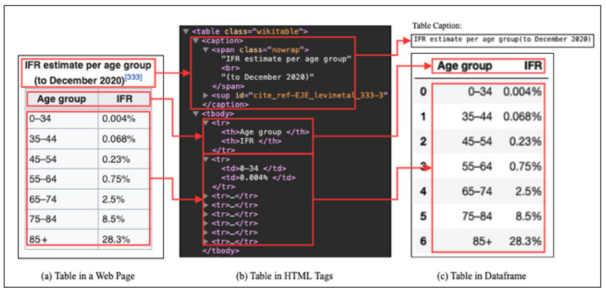

Fig.4: *Example results of web scraping algorithm.*

3.3 Enumerated Term

In this step, classes in the extracted Turtle ontology and dataframes from the previous step are analyzed to enumerate terms for ontology using text analytics. If the extracted ontology and dataframes

are relevant to COVID-19, such class names and dataframe names are defined as terms in this work's COVID-19 ontology. There are two subprocesses in this step: text preprocessing and text similarity measurement.

1) Text Preprocessing: The extracted ontology class names follow the proper naming convention. Therefore, text preprocessing for ontology class names is not required. Only the dataframe names are performed text preprocessing as follows:

a. *Noun Phrase Extraction:* The dataframe names obtained in the previous step may be compound nouns or sentences. If the dataframe is a sentence, only its compound nouns are analyzed. spaCy [15] is a library for natural language processing in Python trained from large text corpus. It is designed for text analytics using Convolutional Neural Network (CNN). The feature provided by the spaCy are tokenization, Parts of Speech (POS) tagging, and text classification. The spaCy is used in this step to extract the noun(s) from each dataframe name. There are three subprocesses in this step: POS tagging, noun chunking, and dependency parsing. The POS tagging is used to identify subject-verb-object of the dataframe names. Later, compound nouns are extracted from the POS tagging. Finally, the UD is applied to remove preposition, verb, and adjective from the compound nouns. An algorithm for noun phrase extraction is displayed in Fig. 5.

b. *Text Stemming:* This step is used to extract the base form of the nouns in the previous step by removing their affixes. For example, the stem of the words vaccines and vaccinated is a vaccine.

```
Input:  dataframeNames
Output: nouns (nouns of dataframes)
1  Function noun_phrase_extraction(dataframeNames):
2      nouns = EMPTY_LIST
3      nlp = spacy.load('en_core_web_lg')
4      For each dataframeName in dataframeNames:
5          doc = nlp(dataframeName)
6          For nounPhrase in doc.noun_chunks:
7              If any nounPhrase.dep_ is in the order of prioritizing the SUBJ or OBJ:
8                  Append text_stemming(nounPhrase.root.text) to nouns
9      Return nouns
```

Fig. 5: *Noun phrase extraction algorithm using spaCy.*

2) Text Similarity Measurement: In this step, the cosine similarity is used to measure the two similarities: (1) similarities of ontology class names and determined scopes, and (2) similarities of nouns and determined scopes. An algorithm of text similarity measurement is displayed in Fig. 6 and described as follows:

a. *Similarity Measurement of Class Names:* Each ontology Turtle class name is measured cosine similarity to the five scopes defined in the step 1. If the classes contain is-a relationships, only the superclasses are measured because their subclasses are determined as terms according to inheritance property. If either of the similarity values is greater than the threshold value, such a class name is determined as a term in the COVID-19 ontology. The minimum threshold is set to 0.5. The closer the class is related to the scope, the higher similarity value to the scope the class has.

b. *Similarity Measurement of nouns:* Each noun is measured cosine similarity to the five scopes. If either of the similarity values is greater than the threshold value, such a noun is determined a term in the COVID-19 ontology. Otherwise, the column names are measured instead and the column with the highest similarity value is determined as term, and the value in each row of such a column is an instance name. The columns used to perform similarity measurement must contain texts and must not empty in any rows because instance name must convey its meaning. For compound nouns containing specific words, such as polio vaccine and monkeypox, the specific words are compared with synonyms of the five scopes. The specific words that do not match any of the five scopes are eliminated.

```
Input:  threshold, determinedScopes, nouns (nouns are from class names,
        dataframe name, and column names)
Output: termLists
1  Function cosine_similarity(threshold, determinedScopes, nouns)
2      termLists = EMPTY_LIST
3      cosSims = EMPTY_LIST
4      matchScopes = EMPTY_LIST
5      For each noun in nouns:
6          For each scope in determinedScopes:
7              vector1 = word2vec(scope)
8              vector2 = word2vec(noun)
9              cosSim = dot(vector1, vector2) / (norm(vector1) * norm(vector2))
10             If cosSim >= threshold:
11                 Append cosSim to cosSims
12                 Append scope of cosSim to matchScopes
13         If cosSims is not EMPTY:
14             maxCosine = max(cosSims)
15             maxMatchScope = max(matchScopes)
16         Append maxMatchScope, maxCosine, noun to termLists
17     Return termLists
```

Fig. 6: *Text similarity algorithm using spaCy.*

3.4 Define Class and Taxonomy

This step involves defining class and taxonomy. Class taxonomy can be derived from the is-a relationship specified in the extracted ontology. There are two subprocesses in this step:

1)Define Class: Terms obtained from the previous step are defined as classes. The class names obtained from the previous step, hereafter, called dataframe class names. Instances of the defined classes are divided into two formats:

a. The instances of classes in the extracted ontology, hereafter, called extracted ontology instances.

b. The instances of classes obtained from the previous step, hereafter, called dataframe instances. Each dataframe is defined as a class. Each row in the dataframe is defined as an instance. Columns of the dataframe are defined as data properties.

2) Define Taxonomy: Class taxonomy is derived from class hierarchies of the extracted ontology.

3.5 Define Instance and Property

First, the COVID-19 ontology is constructed by importing ontology schema and ontology data from the extracted ontology. Then, instances and data properties from dataframes are merged to COVID-19 ontology. Finally, the object properties are constructed on the COVID-19 ontology. There are three subprocesses described as follows:

1) Import extracted ontology: SPARQLWrapper is used to import schema, along with instances and their properties from extracted ontology, into COVID-19 ontology. The SPARQL queries are generated to query the extracted ontology and insert the query results into the COVID-19 ontology. An algorithm for importing extracted ontology is displayed in Fig. 7.

2) Construct instances and data properties from dataframes: The dataframe class names are checked for redundancy with the classes in the same scope on the COVID-19 ontology. If the class names are duplicated, instances of such duplicated classes are merged. Otherwise, instances and data properties from dataframes are inserted. If the duplicate classes have the same data property name, the data values of the identical data property are also merged.

3) Construct object properties: The data property names of each class from dataframes are checked for redundancy with the data property names of the other classes. If property names are redundant, the new object property will be constructed as the object property link. The new property is named by concatenating "has_related_by_" and the redundant property name. For example, "has_related_by_symptom". The data property names of each class from dataframes are later checked for redundancy with data property names of the extracted ontology classes, using the same criterion.

Input: ***extractedOntology***, ***dataframes***
Output: ***COVID-19_Ontology***

```
1  Function import_extracted_ontology(extractedOntology):
2      COVID-19_Ontology = EMPTY
3      sparqlQueries = SPARQLWrapper(extractedOntology)
4      For each class in extractedOntology:
5          Execute following sparqlQueries statement:
               ''' SELECT ?inst ?prop ?thing WHERE {
                   ?inst rdf:type class ; ?prop ?thing .
                   FILTER(!isLiteral(?thing) || (isLiteral(?thing)
                   && lang(?thing)="en")) }'''
6          Append the queried results to COVID-19_Ontology
7      Return COVID-19_Ontology
```

Fig.7: *An algorithm for importing extracted ontology using SPARQLWrapper.*

4. EXPERIMENTAL RESULTS AND EVALUATION

4.1 Experimental Results

Python with SPARQLWrapper, BeautifulSoup, and spaCy library were used to develop the experiment, relying on ontology engineering processes and text analytics. The experimental results are as follows:

1) Determine Scope: COVID-19, Coronavirus, Disease, Pandemic, and Vaccine are defined as scopes of COVID-19 ontology.

2) Data Acquisition: The structured and unstructured data collected from four heterogeneous data sources are transformed into two main formats: ontology Turtle and dataframe. The examples of these formats are displayed in Fig. 8 and 9.

```
@prefix rdf:    <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix dbr:    <http://dbpedia.org/resource/> .
@prefix owl:    <http://www.w3.org/2002/07/owl#> .
dbr:COVID-19    rdf:type     owl:Thing ,
@prefix dbo:    <http://dbpedia.org/ontology/> .
dbr:COVID-19    rdf:type     dbo:Disease .
@prefix rdfs:   <http://www.w3.org/2000/01/rdf-schema#> .
dbr:COVID-19    rdfs:label  "COVID-19"@en ;
    rdfs:comment    "Coronavirus disease 2019 (COVID-19) is a contagious disease caused b
    coronavirus 2 (SARS-CoV-2). The first known case was identified in Wuhan, China, in D
    spread worldwide, leading to an ongoing pandemic."@en .
@prefix dbp:    <http://dbpedia.org/property/> .
dbr:COVID-19    dbp:name    "Coronavirus disease 2019"@en ,
        ""@en ;
    dbp:frequency   "confirmed cases"@en ;
    dbp:synonyms    "COVID,  coronavirus"@en ;
    dbp:prevention  "Face coverings, quarantine, physical/social distancing, ventilation,
    dbo:deathCause  dbr:Giovanni_Battista_Rabino ,
        dbr:Anatolii_Mokrousov ,
        dbr:Abdul_Ghafar_Atan ,
        dbr:Steven_Dick ,
        dbr:Suresh_Angadi ,
```

Fig.8: *Example results of data acquisition in the ontology Turtle format.*

	ISO3	VACCINE_NAME	PRODUCT_NAME	COMPANY_NAME	AUTHORIZATION_DATE
0	SHN	AstraZeneca - AZD1222	AZD1222	AstraZeneca	NaN
1	GRL	Moderna - mRNA-1273	mRNA-1273	Moderna	NaN
2	FRO	Moderna - mRNA-1273	mRNA-1273	Moderna	NaN
3	FRO	Pfizer BioNTech - Comirnaty	Comirnaty	Pfizer BioNTech	NaN
4	BIH	AstraZeneca - AZD1222	AZD1222	AstraZeneca	NaN
...	...	...	...	...	...
1050	TKM	Unknown Vaccine	Unknown Vaccine	NaN	NaN
1051	UZB	Unknown Vaccine	Unknown Vaccine	NaN	NaN
1052	MCO	Novavax - Covavax	Covavax	NaN	NaN

Fig.9: *Example result of data acquisition in the dataframe format.*

3) Enumerate Terms: The Turtle ontology and dataframes from a previous step are enumerated using text preprocessing and cosine similarity measurement. The results are displayed in Table 1, which shows the number of enumerated terms using text analytics in many cosine similarity values. The higher the cosine similarity value, the lower the amount of the enumerated terms.

4) Define Class and Taxonomy: The terms from the previous step are defined as classes. The class taxonomies are derived from the extracted ontology Turtle. Classes and taxonomies defined in this step are displayed in Fig. 10 using protégé [16]. Protégé is a knowledge management tool, whose ontology schemas and instances can be visualized as ontology graphs.

Table 1: *Number of Enumerated Terms of COVID-19 Concepts Acquired from Heterogeneous Data Sources.*

Data Format	Original Data		Number of Enumerated Terms					
			Cosine Similarity Value					
	Relevant	Irrelevant	0.5	0.6	0.7	0.8	0.9	1.0
Ontology Turtle	122	36	116	50	2	2	2	2
Dataframe	24	7	24	24	21	13	2	0
Total of distinct terms	141	43	135	70	21	13	3	2

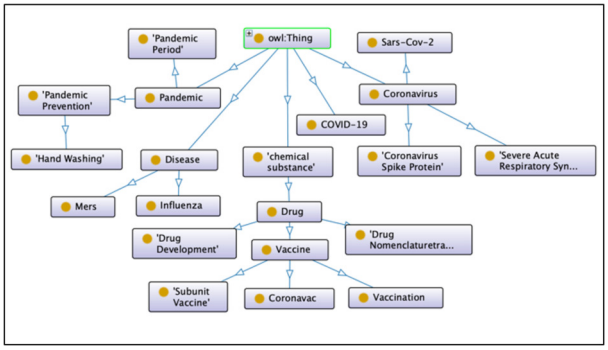

Fig.10: *Example classes and taxonomies of COVID-19 ontology.*

5) Define Instance and Property: Instances from the extracted ontology Turtle and dataframes are merged, as displayed in Fig. 11.

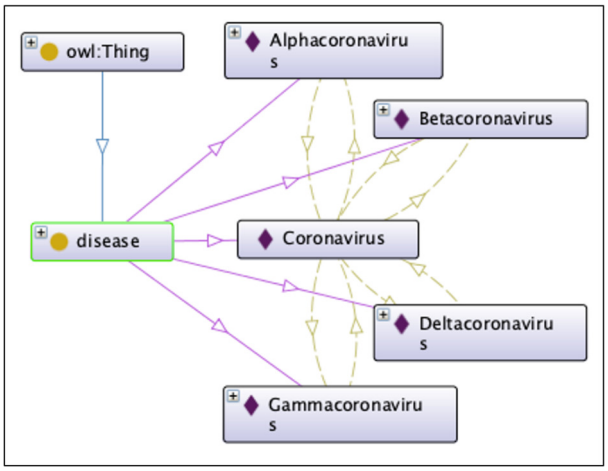

Fig.11: *Examples of instances and properties of COVID-19 ontology.*

4.2 Evaluation

The work's COVID-19 ontologies were constructed using text analytics with cosine similarity varying from 0.5 to 1.0. The data used in the ontology construction contain data related and unrelated to COVID-19 in a ratio of 70 to 30. The six constructed ontologies were evaluated for data consistency with the original data, and the following scores were measured: precision, recall, and F1-score. The precision refers to the fraction of the constructed instances relevant to COVID-19 and constructed instances. The recall refers to the fraction of the constructed instances relevant to COVID-19 and constructed instances plus constructed instances that are irrelevant to COVID-19. The work's ontology with 0.6 cosine similarity has better performance than the others as shown in Fig. 12. The elements of the COVID-19 ontologies were counted and displayed in Table 2.

For the ontology of COVID-19 that was generated with a cosine similarity of 0.5, it still contained information not related to COVID-19. However, the constructed COVID-19 ontology with cosine similarity ranging from 0.7 to 1.0; some relevant COVID-19 information is eliminated. The higher the cosine similarity value, the more relevant classes are eliminated.

The high cosine similarity leads to the loss of relevant information. On the other hand, if the cosine similarity is set too low, the irrelevant information will be included. Thus, the cosine similarity threshold must be set appropriately. The ontology with 0.6 cosine similarity is the best value for integrating COVID-19 ontology in this case study.

In conclusion, integrating knowledge does not need the highest cosine similarity value, because not only the exactly matched information but also relevant information is to be merged.

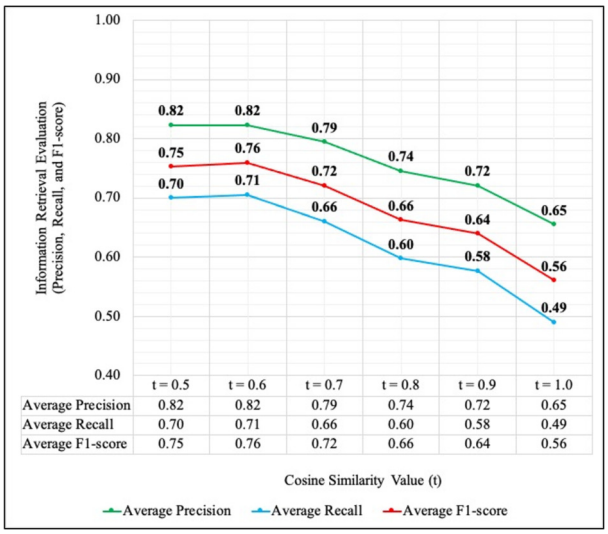

Fig.12: *Evaluation of the work's COVID-19 ontologies.*

Example classes that are eliminated from the constructed COVID-19 ontologies with text analytics are displayed in Fig. 13. Example classes irrelevant to determined scopes of the constructed COVID-19 ontology with 0.5 cosine similarity are displayed in Fig. 14.

4.3 Implementation Guideline

Web application for semantic search was developed. The work's COVID-19 ontology with 0.6 cosine

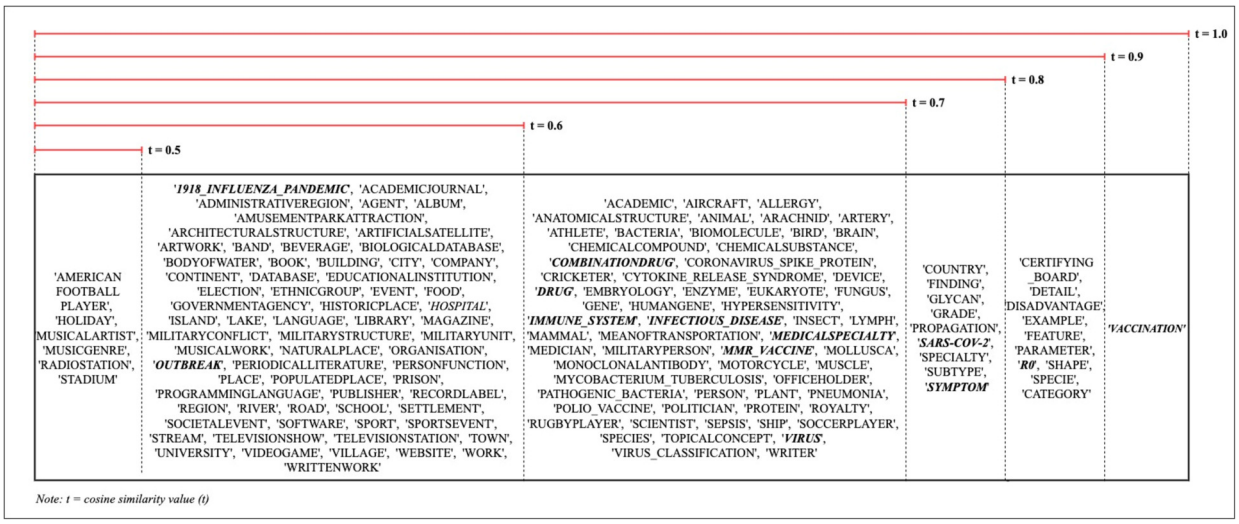

Fig.13: *Example classes that are eliminated from the constructed COVID-19 ontologies with text analytics.*

Table 2: *Number of Ontology Schema and Ontology Data of Constructed COVID-19 Ontologies.*

COVID-19 Ontologies	Determined Scopes	Ontology Schema and Ontology Data				
		#class	#instance		#data property	#object property
			r[a]	ir[b]		
Original Data	COVID-19	20	248	74	488	18
	Coronavirus	5	169	50	384	7
	Disease	61	309	92	315	16
	Pandemic	4	133	39	257	12
	Vaccine	55	512	153	471	10
Using text analytics with cosine similarity = 0.5	COVID-19	7	204	29	253	1
	Coronavirus	2	99	6	114	1
	Disease	18	141	50	283	13
	Pandemic	9	68	14	102	7
	Vaccine	40	230	100	370	1
Using text analytics with cosine similarity = 0.6	COVID-19	4	198	26	88	1
	Coronavirus	1	94	10	14	1
	Disease	18	131	53	174	13
	Pandemic	8	63	14	52	7
	Vaccine	18	182	46	151	1
Using text analytics with cosine similarity = 0.7	COVID-19	0	0	0	0	0
	Coronavirus	0	0	0	0	0
	Disease	1	131	28	152	11
	Pandemic	8	54	12	102	7
	Vaccine	1	111	39	30	1
Using text analytics with cosine similarity = 0.8	COVID-19	0	0	0	0	0
	Coronavirus	0	0	0	0	0
	Disease	1	88	22	123	11
	Pandemic	3	21	11	54	1
	Vaccine	1	102	29	30	1
Using text analytics with cosine similarity = 0.9	COVID-19	0	0	0	0	0
	Coronavirus	0	0	0	0	0
	Disease	1	68	17	90	11
	Pandemic	1	11	9	44	0
	Vaccine	1	99	23	30	1
Using text analytics with cosine similarity = 1.0	COVID-19	0	0	0	0	0
	Coronavirus	0	0	0	0	0
	Disease	1	44	17	24	11
	Pandemic	1	10	7	13	0
	Vaccine	0	0	0	0	0

[a] Number of relevant instances of constructed ontology

[b] Number of irrelevant instances of constructed ontology

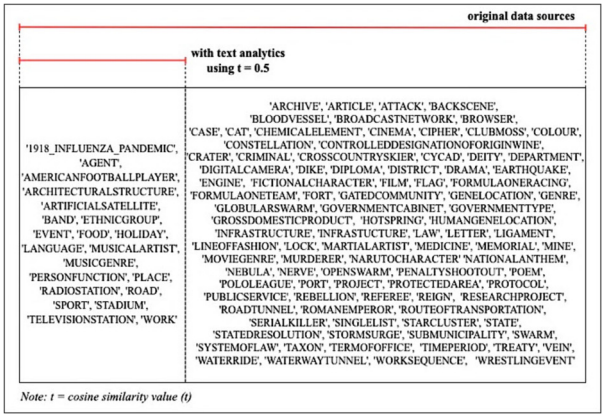

Fig.14: *Example classes that are irrelevant to determined scopes of the constructed COVID-19 ontologies with 0.5 cosine similarity.*

similarity is used as a knowledge base for the semantic search. Tools used for web application development are Apache Jena Fuseki [17] and Python Django [18]. Apache Jena Fuseki is a SPARQL server package for a web application. Django is used to develop User Interface (UI) for submitting keyword search. They work together as follows. First, the keyword submitted from UI is transformed into SPARQL query statement using SPARQLWrapper to query COVID-19 knowledge stored in the SPARQL server. Then, the query results are returned to the UI. The system architecture of the semantic search is displayed in Fig. 15. An example of semantic search using "COVID-19" keyword is displayed in Fig. 16 and the search results relevant to the COVID-19 scopes are displayed in Figs. 17 and 18. The example of searching vaccine data is displayed in Fig. 19.

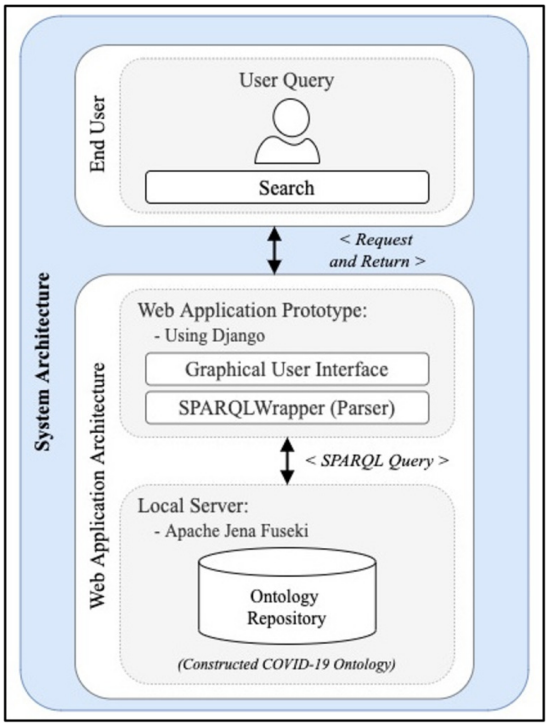

Fig.15: *System architecture for semantic search.*

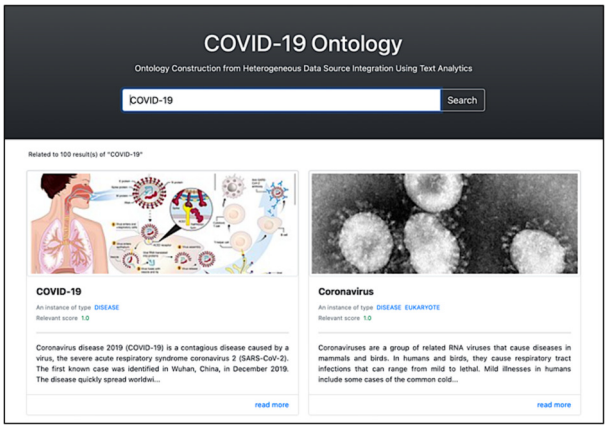

Fig.16: *A example of semantic search with the COVID-19 keyword.*

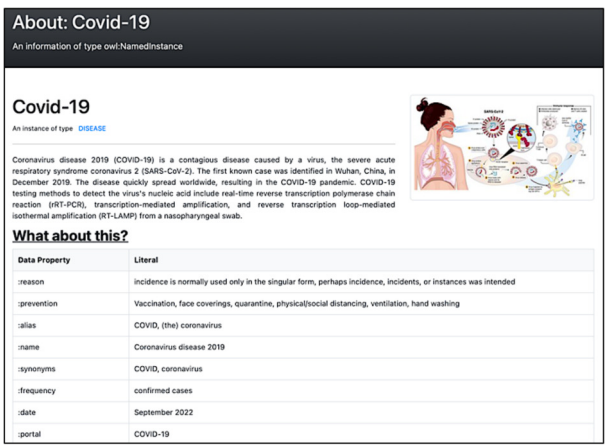

Fig.17: *Example results of COVID-19 scope.*

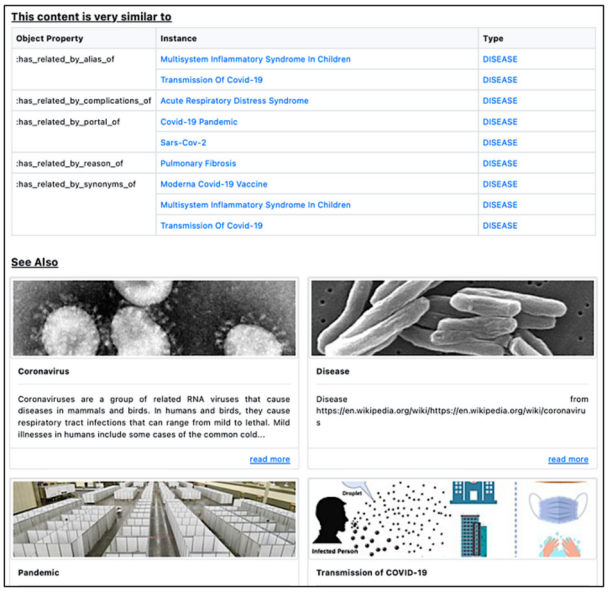

Fig.18: *Example results relevant to COVID-19 scope.*

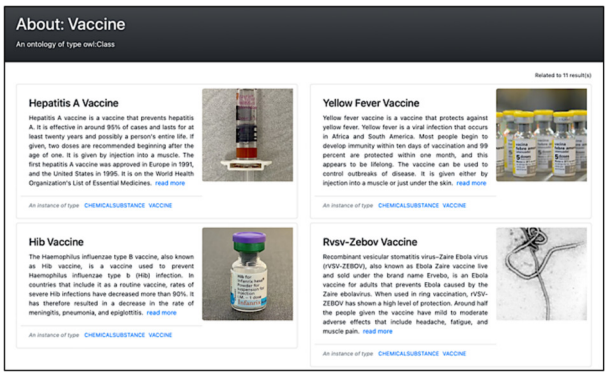

Fig.19: *Example results relevant to vaccine scope.*

5. CONCLUSIONS

The COVID-19 ontology was constructed automatically from heterogeneous data sources with different data structures using ontology engineering and text analytics. COVID-19 information to be created as instances in the COVID-19 were collected from DBpedia, WHO web API, disease.sh web API, and Wikipedia, each with a different structure and format. The information gathered from these sources is divided into two types: structured and unstructured data. Web scraping was used to collect unstructured COVID-19 data from Wikipedia web pages. Web-based services were used to collect the COVID-19 information in structured formats: CSV files, JSON files, and an ontology Turtle file. POS tagging, UD and text stemming in natural language processing, and text similarity measurement were used to integrate the COVID-19 information from heterogeneous data sources. The work's COVID-19 ontology with 0.6 cosine similarity is the optimal value, obtains high precision and recall, and the constructed ontology has

the highest amount of relevant COVID-19 information.

This research may be used to perform automated knowledge integration for other domains. The steps given in the methodology section can be used; only scopes in the first step must be redefined. In addition to knowledge integration, this research's methodology can also be applied to migrate data from heterogeneous data sources.

The challenges of this research are (1) Multiple sentences included in the same paragraph and (2) many emerging terminologies. It is difficult to find the relationship between sentences in the paragraph. Automatic ontology construction is required to tackle new terminologies. Several data analytic techniques must be used to define classes, taxonomies, instances and properties.

The COVID-19 information from CSV files, JSON files, and web pages have not been assigned the class taxonomy. Text classification in NLP could be applied for the task.

The COVID-19 information from web pages does not include contents in paragraph tags (<p>). These contents could be included in the COVID-19 ontology schema and instances using Named Entity Recognition (NER) [19] and Hearst Patterns [20]. NER and Hearst Patterns are combined to define taxonomy or superclass/subclass relationship residing in the contents of paragraph tags. The NER is used to identify what class the instances belong to and Hearst Patterns is later used to identify taxonomy.

ACKNOWLEDGEMENT

The researchers are grateful to Assistant Professor Dr. Prakarn Unachak for his suggestions and comments.

References

[1] N. K. Soe, T. T. Yee and E. C. Htoon, "Semantic Layer Construction for Big Data Integration," *2020 International Conference on Advanced Information Technologies (ICAIT)*, Yangon, Myanmar, pp. 24-29, 2020.

[2] A. Berko *et al.*, "Application of Ontologies And Meta-Models for Dynamic Integration of Weakly Structured Data," *2020 IEEE Third International Conference on Data Stream Mining & Processing (DSMP)*, Lviv, Ukraine, pp. 432-437, 2020.

[3] A. V. Saurkar, K. G. Pathare and S. A. Gode, "An Overview on Web Scraping Techniques and Tools," *2018 International Journal on Future Revolution in Computer Science & Communication Engineering (ijfrcsce)*, vol. 4, pp. 363–367, Apr. 2018.

[4] D. M. Thomas and S. Mathur, "Data Analysis by Web Scraping using Python," *2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA)*, Coimbatore, India, pp. 450-454, 2019.

[5] M. M. Fouad, T. F. Gharib and A. S. Mashat, "Efficient Twitter Sentiment Analysis System with FeatureSelection and Classifier Ensemble," *2018 The International Conference on Advanced Machine Learning Technologies and Applications (AMLTA)*, Cairo, Egypt, Jan. 2018.

[6] M.-C. de Marneffe, C. D. Manning, J. Nivre, and D. Zeman, "Universal Dependencies," in *Computational Linguistics*, Cambridge, MA: MIT Press, pp. 255–308, Jun. 2021.

[7] X. Xue, H. Wang, J. Zhang and Y. Huang, "Matching Transportation Ontologies with Word2Vec and Alignment Extraction Algorithm," in *2021 Journal of Advanced Transportation*, Hindawi, May 2021.

[8] G. Antoniou, P. Groth, F. van Harmelen and R. Hoekstra, *A Semantic Web Primer*, Third Edition. London, England: The MIT Press, 2012.

[9] C. M. Keet, *An Introduction to Ontology Engineering*, College Publications, 2018.

[10] N. Rastogi, P. Verma and P. Kumar, "Evaluation of Information Retrieval Performance Metrics using Real Estate Ontology," *2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT)*, Tirunelveli, India, pp. 102-106, 2020.

[11] E. R. Swedia, A. B. Mutiara, M. Subali, and Ernastuti, "Deep Learning Long-Short Term Memory (LSTM) for Indonesian Speech Digit Recognition using LPC and MFCC Feature," in *2018 Third International Conference on Informatics and Computing (ICIC)*, pp. 1–5, Oct. 2018.

[12] The SPARQLWrapper Development Team, *SPARQLWrapper Documentation*, 2022. [Online]. Available: `https://sparqlwrapper.readthedocs.io/en/latest/`

[13] E. Winters *et al.*, *disease.sh - An open API for disease-related statistics*, 2022. [Online]. Available: `https://disease.sh/docs/`

[14] The urllib3 Development Team, *urllib3 Documentation*, 2023. [Online]. Available: `https://urllib3.readthedocs.io/en/stable/`

[15] H. Matthew and I. Montani, *spaCy: Industrial-strength Natural Language Processing in Python*, 2022. [Online]. Available: `https://spacy.io/usage`

[16] M. Horridge, "A Practical Guide To Building OWL Ontologies Using Protégé 4 and CO-ODE Tools Edition 1.3," *The University of Manchester*, Mar. 2011.

[17] The Apache Jena Project Team, *Apache Jena Fuseki*, The Apache Software Foundation, 2022. [Online]. Available: `https://jena.apache.org/documentation/fuseki2/`

[18] The Django Software Foundation, *Django*, The Django Software Foundation, 2022. [Online]. Available: `https://www.djangoproject.com/foundation/`

[19] N. Kanya and T. Ravi, "Modelings and techniques in Named Entity Recognition-an Information Extraction task," *IET Chennai 3rd International on Sustainable Energy and Intelligent Systems (SEISCON 2012)*, Tiruchengode, pp. 1-5, 2012.

[20] A. I. A. Aldine, M. Harzallah, B. Giuseppe, N. Béchet and A. Faour, "Redefining Hearst Patterns by using Dependency Relations," in *Proceedings of the 10th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management(IC3K2018)*, vol.2, pp. 148-155, Jan. 2018.

Patipon Wiangnak received his Master of Science and Bachelor of Science degrees in Computer Science from the Department of Computer Science, Faculty of Science, Chiang Mai University, Thailand, in 2023 and 2019, respectively. He worked for a healthcare organization for almost three years, gaining experience in database management, data analysis, and software development. His research interests include knowledge engineering, natural language processing, ontology, and the semantic web.

Areerat Trongratsameethong is a lecturer of the Computer Science Department, Faculty of Science, Chiang Mai University. She has her Ph.D. in computer science from Mahidol University, Bangkok, Thailand. She worked for business companies and software houses for many years while gaining experience in software development, system analysis and design, and software requirement analysis. Her current expertise and teaching focus on fundamentals of programming, fundamentals of database system, object-oriented design, organization of programming language, and ontology design and development.